Latent Representation of CFD Meshes and Application to 2D Airfoil Aerodynamics

Zhen Wei [*] , Benoît Guillard [†] , and Pascal Fua [‡]
EPFL, Lausanne, 1015, Switzerland

Vincent Chapin [§] and Michaël Bauerheim [¶]
ISAE-SUPAERO, Toulouse, 31055, France

Mesh manipulation is central to Computational Fluid Dynamics (CFD). However, creating appropriate computational meshes often involves substantial manual intervention that has to be repeated each time the target shape changes. To address this problem, we propose an auto-decoder-based latent representation approach. Human prior knowledge is embedded into learned geometric patterns, which eliminates the need for further handcrafting. Furthermore, the resulting computational meshes are differentiable with respect to the model parameters, which makes it suitable for inclusion in end-to-end trainable pipelines. We apply the model on 2D airfoils to demonstrate its ability to handle various tasks.

Nomenclature

C_d	=	the drag coefficient
$g_\Theta()$	=	the auto-decoder model parameterized by Θ
$h()$	=	the surrogate model
$\mathcal{L}$	=	the loss function
$\hat{M} = \{\hat{V}, \hat{R}\}$	=	the template CFD mesh and its vertices, edges
M, M^S, M^{IO}	=	the decoded CFD mesh, its object surface submesh and inlet/outlet submesh
X_{ib}, $\mathbf{r}_{ib}$, $\mathbf{t}_{ib}$	=	the skewed coordinate system and its bases defined at the i-th vertex and the b-th pair of mesh edges
S	=	the target airfoil geometry to be encoded
$\mathbf{z}$	=	the latent vector

[*]Doctoral Student, Computer Vision Lab, School of Computer and Communication Sciences, AIAA Member
[†]Doctoral Student, Computer Vision Lab, School of Computer and Communication Sciences
[‡]Professor, Computer Vision Lab, School of Computer and Communication Sciences. Corresponding author, email: pascal.fua@epfl.ch
[§]Associate Professor, Department of Aerodynamics, Energies and Propulsion
[¶]Associate Professor, Department of Aerodynamics, Energies and Propulsion

I. Introduction

Mesh manipulation is at the heart of Computational Fluid Dynamics (CFD) applications. However, generating appropriate computational meshes often involves much manual intervention that has to be repeated for each new shape under consideration. This is time-consuming for several reasons. First, the parameterization and its hyper-parameters depend heavily on the target object, for which strong geometric prior knowledge and a case-by-case analysis are always required. Second, the resulting model is specialized and rarely generalizes to different geometries or boundary conditions. Hence, re-meshing is required when the shape changes and much time is wasted in industrial practice. Finally, none of the existing approaches to automated mesh representation [1–9] are designed to be differentiable so they can be integrated seamlessly into gradient-based frameworks, such as those that involve deep learning.

In this work, we propose a fully automated approach to redesigning a computational mesh as the target shape changes. It relies on a latent representation, i.e. a vector in a low-dimensional space, of both the object surface and the corresponding computational mesh. Through deep learning, the model learns geometric priors for the target objects while guaranteeing that the resulting meshes can be used to produce accurate simulations. To this end, it encodes an unstructured point cloud sampled from the object surface into a low-dimensional latent vector and then decodes it into an appropriately deformed CFD mesh. To preserve its quality, we introduce a regularization loss and a differentiable Active Model layer. The proposed model eliminates most manual steps, and avoids re-meshing without relying on any specific mesh parameterization. The method can also output various mesh types, such as structured, unstructrued or hybrid meshes. Additionally, the resulting computational meshes are fully differentiable with respect to the latent vector, which allows integration into deep learning pipelines or any gradient-based optimization frameworks.

We develop and validate our approach on 2D airfoils. The paper is organized as follows. In Sec. II, we provide a literature review of related mesh representation methods. In Sec. III, we formulate and discuss the technical details of the proposed model. In Sec. IV, we conduct comprehensive experiments to prove the effectiveness of the proposed representation. The model is then integrated into an end-to-end shape optimization pipeline to demonstrate the benefits of full differentiation. In Appendix, we explore the properties of learned latent space. Unelaborated mathematical and experimental details are also provided.

II. Related Work

In the CFD literature, the mesh representation has been widely applied in various tasks, to name a few recent works, including mesh deformation [10, 11], aerodynamic shape optimization [12, 13], multidisciplinary design [14, 15], etc. Typical examples arise in simulations with fluid-structure interactions, since the geometry and thus the associated mesh are changing over time as a result of the balance between the aerodynamic load and the mechanical structure. For such problems, difficulties to guarantee the mesh quality appear when large deformations occur [16]. Similarly, in the contexts of aerodynamic optimization [17], uncertainty quantification (UQ, [18]) or sensitivity studies, numerous

evaluations of various shapes and/or meshes are required, which again is difficult to automatize while guaranteeing a proper mesh quality for CFD. Specifically, both geometrical quality (aspect ratio, skewness, etc.) and physical requirements (boundary layers, shocks, etc.) have to be maintained when deforming the mesh, a property named here as Mesh Quality Preservation (MQP).

For these tasks, mesh representations are usually derived from explicit handcrafted parameterizations instead of directly from the mesh. Common options include fixed sampling schemes [2], nonuniform rational B-splines (NURBS) [19], control points for Free-Form Deformations (FFD) [20–22] or for Radial Basis Functions (RBF) [23]. Manual hyperparameter tuning is needed to adapt to different geometries, such as the number and the positions of control points, the values of supporting radius, etc. In all cases, the mesh ends up being described by a vector.

As for the linear dimension reduction methods, a popular way to produce a compact representation is to use the Proper Orthogonal Decomposition (POD) that mathematically derives a set of orthogonal modes. This can involve finding basis functions via Gram-Schmitt orthogonalization on a few geometries [1] or applying an SVD to create optimal orthogonal shape modes given a training dataset [2, 3, 5, 24]. However, reconstructing the geometry from a latent vector remains difficult and requires dedicated handcraft engineering for a specific task. The Active Subspace Model (ASM) [4, 7, 25–27] and Active Subspace Identification (ASI)[28, 29] reduce the dimension by analyzing the gradient of surrogate models. It is intended to limit the curse of dimensionality [30] for surrogate-based optimization or uncertainty quantification. It uses random sampling methods to estimate the real active subspace. Even though it requires handcrafted rules to generate valid samples, ASM's approximation error is upper bounded by the Poincaré constant that increases with dimensionality given a limited number of samples [31, 32]. The Class/Shape function Transformation (CST) [33] describes 2D airfoil shapes by the summation of Bernstein polynomial basis. Higher dimensional CST representation works for more complicated geometries but has a slower convergence rate when applied in the shape optimization task [34]. Non-linear approaches have also been proposed. For example, Generative Topographic Mapping (GTM) [6] is used to project a 30-dimensional design variable to a two dimensional latent vector. However, because GTM involves a Bayesian generative model, its latent representation cannot easily be integrated into a gradient-based pipeline. Furthermore the complexity of tuning hyperparameter for the radius basis grows exponentially with the latent space's dimensionality. Finally, all the models mentioned above are either linear projections or single nonlinear projections with Gaussian kernels. By relying on a nonlinear deep neural network, our proposed model can learn more complex representations.

More recently, Generative Adversarial Network (GAN) has been used for novel geometry generation [35–37]. They can be difficult to train and efforts have been made to stabilize their training and to filter out invalid results [8, 9]. They are typically used for data sampling to generate novel shapes and augment training data for the subsequent parameterization. Hence, they serve as preconditioning modules in step-by-step frameworks and cannot contribute to gradient propagation in end-to-end pipelines.

We took our inspiration from computer vision work that has convincingly demonstrated the ability of auto-encoders [38, 39], variational auto-encoders [40] and auto-decoders [41, 42] to learn latent geometric representations and eliminate the explicit shape parameterization. The deep learning based models are able to learn accurate object surface representation automatically with the signed distance field. Recent progress makes the auto-decoder models fully differentiable [43]. However, unlike in these approaches, our model is mesh-based and represents both the object surface and corresponding computational mesh. The Mesh Quality Preservation for CFD purposes is a major consideration, which was not part of computer vision research. The deep learning algorithms for computer vision therefore have to be thoroughly adapted to the specificity of fluid simulations so that they become useful for the CFD community. Consequently, the main objective of our work is to propose a novel mesh representation and deformation framework, which (i) is handcrafted-parameter-free, (ii) encapsulates MQP, (iii) is fully differentiable for a direct implementation in gradient-based frameworks needed in optimization or uncertainty quantification studies, and additionally (iv) is flexible to the mesh type employed so that structured, unstructured or hybrid meshes can be used. This framework is developed here for 2D meshes and is validated on various CFD tasks, including the aerodynamic optimization of a 2D airfoil with several geometrical constraints.

III. Method

The proposed model is based on deep geometric learning to provide a flexible tool for mesh representation. It is designed to encode a reference airfoil shape and then deform a fixed CFD mesh to reconstruct the geometry. In particular, only points sampled on the airfoil's surface are fed into the pipeline and an entire CFD mesh is generated after encoding and decoding the latent vector. The pipeline is shown in Fig.1. In Section IV, demonstration is in a CFD context where the deformed mesh is employed to perform further CFD computations, including the optimization of the aerodynamic performances of a 2D airfoil. In the remainder of this section, we first formalize this process in Section III.A, and then we describe the neural network and training details used to implement it.

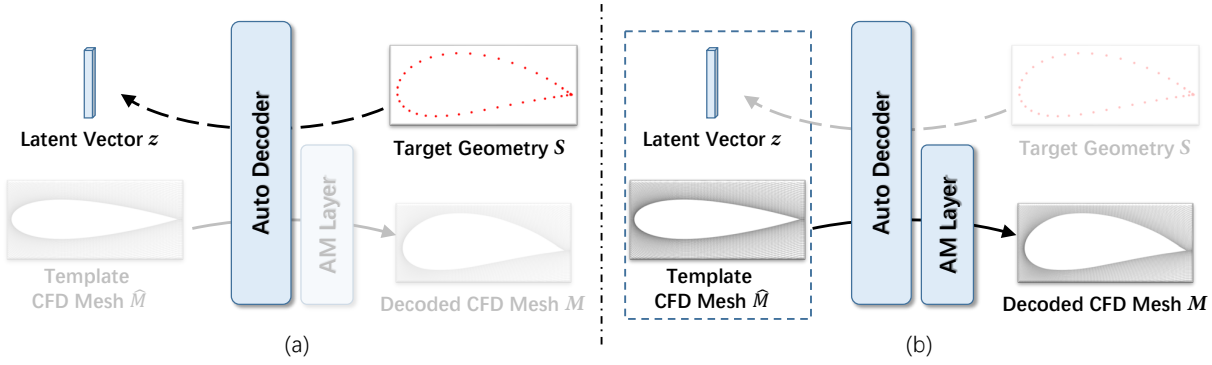

Fig. 1 Pipeline of the proposed model, including (a) the encoding step and (b) the decoding step.

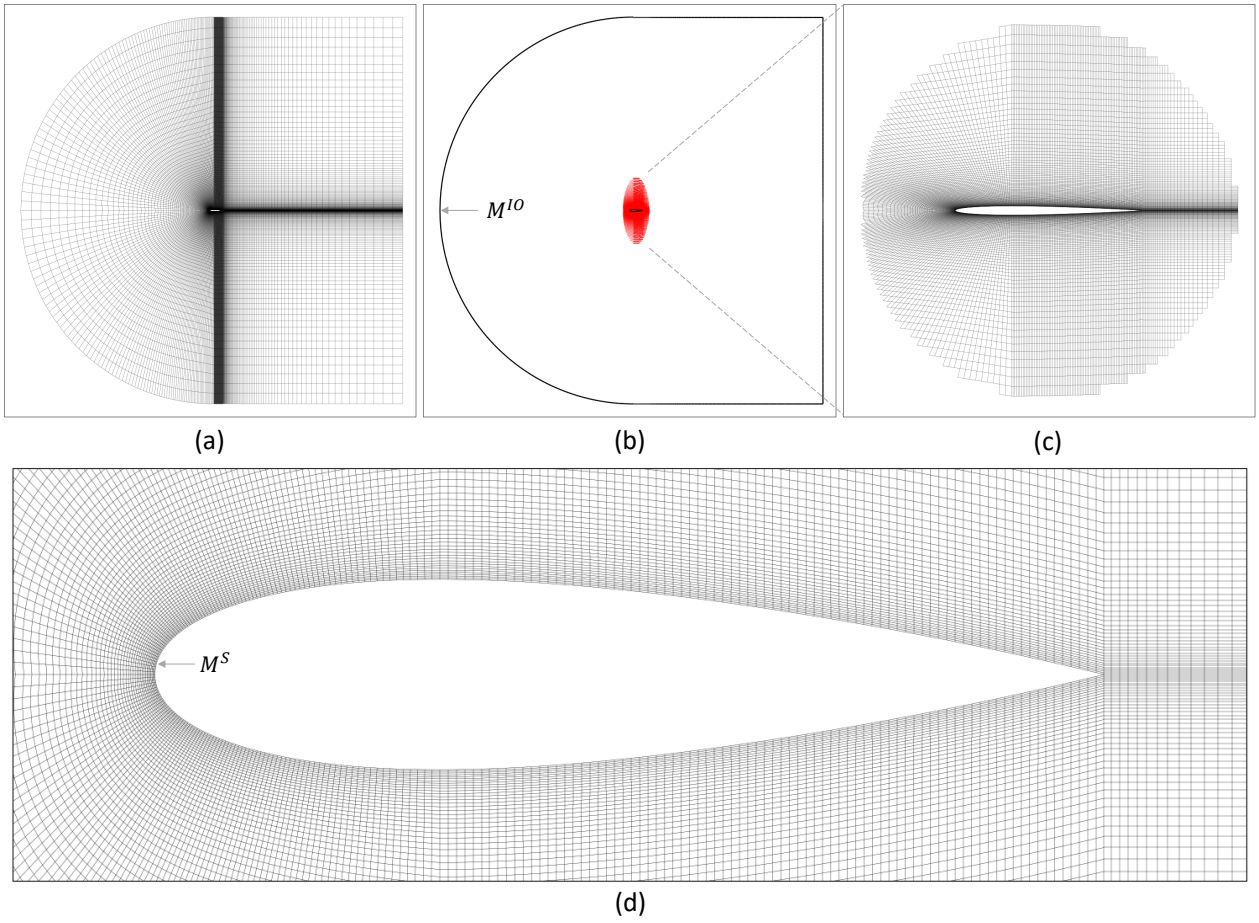

Fig. 2 The default template mesh (TM-A) and its (a) overall mesh, (b) boundary, (c) deformation area and (d) surface.

A. Formalization

Consider an two dimensional airfoil shape represented by a CFD mesh $\hat{M} = \{\hat{V}, \hat{E}\}$, where $\hat{V} = \{\hat{\mathbf{v}}_1, \hat{\mathbf{v}}_2, ..., \hat{\mathbf{v}}_N\}$ stands for the mesh's N 2D vertices $\hat{\mathbf{v}}_i \in \mathbb{R}^2$ for $1 \leq i \leq N$ and $\hat{E}$ for the edges connecting them. N can be arbitrary. Let $\hat{V}^S$ and $\hat{E}^S$ be the subsets of vertices and edges in the surface mesh $\hat{M}^S$ that define the airfoil's profile, and $\hat{M}^{IO}$ be the inflow and outflow boundaries, also known as the inlet and outlet mesh. They define the computational boundary, as shown in Fig. 2. A deformed mesh of the same topology can be represented as $M = \{\hat{V} + \Delta V, \hat{E}\}$, where ΔV is a set of translation vectors, one for each vertex. Note that the proposed framework is intended to encode and decode a geometry by mesh deformation so that the connectivity remains the same.

Let $D = 2N$, $\mathbf{v}$ be the D-vector obtained by stacking all the 2D coordinates of the N vertices of M, and $\delta\mathbf{v}$ the vector of vertex translations represented by ΔV. This enables us to write that $\mathbf{v} = \hat{\mathbf{v}} + \delta\mathbf{v}$, where $\hat{\mathbf{v}}$ describes the vertices of $\hat{M}$. Finally, let $M(\mathbf{v})$ be the mesh with edges $\hat{E}$. Given these definitions, learning a low-dimensional representation of the

mesh means finding a mapping parametrized by Θ

$$g_\Theta : \mathbb{R}^d \rightarrow \mathbb{R}^D \ , \tag{1}$$
$$\delta \mathbf{v} = g_\Theta(\mathbf{z}; \hat{M}) \ ,$$

where $d \ll D$, and $\mathbf{z} \in \mathbb{R}^d$ is the so-called low-dimensional latent vector. Θ represents the weights that control the behavior of the deep neural network that implements g. We denote the decoded mesh deformation as $g_\Theta(\mathbf{z})$ when $\hat{M}$ is fixed in the following discussions for simplicity.

The pipeline encodes and then decodes to fit a target airfoil represented by a geometry S, which is a set of surface sampling points. No topology information is required in S, which removes the need for complicated data pre-processing and facilitate the use of various data formats. Despite S being only an unstructured point cloud, the decoded output consists in a full computational mesh M which is a smooth deformation of $\hat{M}$.

To learn the weights Θ, we use an auto-decoding approach [41, 42]. Given a set of T training geometries that are only composed of sampled surface points, denoted $S_1, \ldots, S_T$, we seek

$$\Theta^*, \mathbf{Z}^* = \underset{\Theta, \mathbf{z}_1, \ldots, \mathbf{z}_T}{\operatorname{argmin}} \sum_{t=1}^{T} \mathcal{L}(M(\hat{\mathbf{v}} + g_\Theta(\mathbf{z}_t)), S_t) \ , \tag{2}$$

where $\mathcal{L}$ is a loss function that is small when $g_\Theta(\mathbf{z}_t)$ yields a deformed mesh that is regular and whose profile defined by $\hat{V}^S$ and $\hat{E}^S$ is close to S_t after deformation. Here, the optimal $\mathbf{z}_t$ corresponds to a low-dimensional representation of the complete airfoil shape S_t.

At inference time, given a target airfoil profile S and frozen weights Θ^*, we solve

$$\mathbf{z}^* = \underset{\mathbf{z}}{\operatorname{argmin}} \, \mathcal{L}(M(\hat{\mathbf{v}} + g_{\Theta^*}(\mathbf{z})), S) \ . \tag{3}$$

In other words, the latent vector $\mathbf{z}$ parameterizes the possible profiles, and their corresponding computational mesh. We adjust this code to be as close as possible to the target $\mathbf{z}^*$ so that the overall mesh M corresponds to the target profile S.

B. Loss Function

The mesh deformation pipeline relies on the latent code $\mathbf{z}$ that provides a low-dimensional representation of the airfoil, and on the function g_{Θ^*} that returns the vector of vertex deformation $\delta\mathbf{v}$ of the mesh. They are learned by minimizing the loss function $\mathcal{L}$ of Eq. 2. Given a decoded CFD mesh $M = M(\hat{\mathbf{v}} + g_\theta(\mathbf{z}))$ and a target airfoil profile S_t, $\mathcal{L}(M, S_t)$ of Eq. 2 should be small if, and only if,

1) the airfoil profile defined by M, namely M^S, is very similar to the target profile S_t,

2) the computational mesh quality of M is adequate for CFD simulations.

Furthermore, unnecessary deformations should be penalized and latent space representations whose dimensionality is too high should be discouraged to guarantee $d \ll D$. Hence we write

$$\mathcal{L} = w_{dist}\mathcal{L}_{dist} + w_{reg}\mathcal{L}_{reg} + w_{dec}\mathcal{L}_{dec} + w_z\mathcal{L}_z \ , \tag{4}$$

where $\mathcal{L}_{dist}$ is small when the airfoil profile defined by M^S is close to the target, $\mathcal{L}_{reg}$ is small when the surface mesh is of high quality, $\mathcal{L}_{dec}$ is a standard decay term on $\delta\mathbf{v}$, and $\mathcal{L}_z$ is a decay term on the latent vector norm. w_{dist}, w_{reg}, w_{dec}, and w_z are loss balancing weights. These loss components are detailed below.

Distance Loss $\mathcal{L}_{dist}$. $\mathcal{L}_{dist}$ is intended to penalize geometric differences between the decoded profile and the target one. As illustrated by Fig.3, we take it to be the sum of two terms

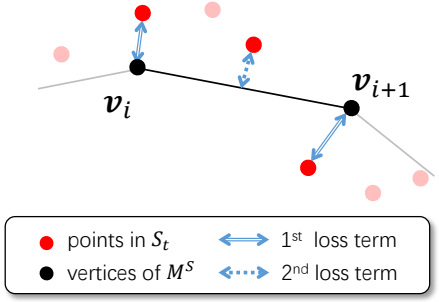

Fig. 3 The visualization of the distance loss $\mathcal{L}_{dist}$.

$$L_{dist} = \frac{1}{|V^S|}\sum_{\mathbf{v}_i^S \in V^S} \min_{s\in S_t} ||s - \mathbf{v}_i^S||^2 + \frac{1}{|E^S|}\sum_{e_j^S \in E^S} \min_{s\in S_t} dist(s, e_j^S) \ , \tag{5}$$

where s is a sampling point in S_t, and $dist()$ stands for the distance of a point to a line segment. We denote V^S the subset of vertices on the deformed airfoil profile, and E^S the corresponding edges. The first term is the sum of the distances of each point in V^S to the closest point in S_t. The second term is the sum of the minimum distance of each edge along the airfoil profile to the closest point in S_t.

Regularization Loss $\mathcal{L}_{reg}$. Minimizing L_{dist} only guarantees that the decoded profile matches the target profile. However, this only involves moving the vertices in $\hat{V}^S$. Doing so without moving the others accordingly would produce extremely irregular and possibly overlapping computational meshes that will make CFD computations inaccurate or numerically unstable. We must therefore ensure that all vertices move accordingly to the surface vertices in order to preserve the deformed mesh quality—cells' skewness, orthogonality, aspect ratio—as well as its ability to represent the underlying physics [44].

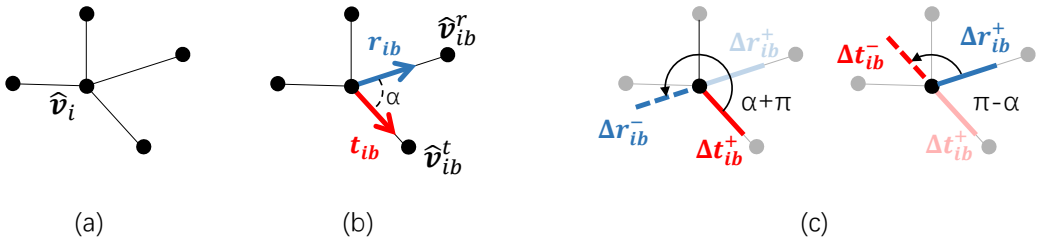

Fig. 4 The skewed coordinate systems. (a) $\hat{\mathbf{v}}_i$ and its neighbors. (b) The two axes. (c) The finite difference discretization.

In practice, we can assume that the template mesh $\hat{M}$ has been designed to satisfy these requirements, and we want the deformed mesh M to keep on satisfying them. To this end, we introduce an energy function that quantifies the degree of distortion of M with respect to $\hat{M}$. Defining $\hat{M}$ as a stationary point and then solving an extreme point finding problem yield a mesh M whose local mesh structure is close to that of $\hat{M}$ so that the mesh quality priors embedded in $\hat{M}$ are preserved. To formulate this energy function, we first parameterize $\hat{M}$ using the skewed coordinate systems relying on the template mesh, as depicted by Fig. 4(a-b). For each vertex $\hat{\mathbf{v}}_i$, we pair its neighboring vertices which results in B_i different two-tuples, such as $(\hat{\mathbf{v}}^r_{i1}, \hat{\mathbf{v}}_{i1})^t$, $(\hat{\mathbf{v}}^r_{i2}, \hat{\mathbf{v}}^t_{i2})$, ..., $(\hat{\mathbf{v}}^r_{iB_i}, \hat{\mathbf{v}}^t_{iB_i})$. We pick one tuple $(\hat{\mathbf{v}}^r_{ib}, \hat{\mathbf{v}}^t_{ib})$ and the base vectors of the coordinate system X_{ib} are defined as the normalized vectors

$$\mathbf{r}_{ib} := \frac{\hat{\mathbf{v}}^r_{ib} - \hat{\mathbf{v}}_i}{||\hat{\mathbf{v}}^r_{ib} - \hat{\mathbf{v}}_i||_2} \quad \text{and} \quad \mathbf{t}_{ib} := \frac{\hat{\mathbf{v}}^t_{ib} - \hat{\mathbf{v}}_i}{||\hat{\mathbf{v}}^t_{ib} - \hat{\mathbf{v}}_i||_2} \,. \tag{6}$$

In this base, the vertex is a function of coordinates as $\mathbf{v}(r_{ib}, t_{ib})$, and vertices from the template and deformed mesh can be written as

$$\begin{cases} \hat{\mathbf{v}}_i(\hat{r}_{ib}, \hat{t}_{ib}) = \hat{r}_{ib}\mathbf{r}_{ib} + \hat{t}_{ib}\mathbf{t}_{ib} \,, \text{ for } \hat{\mathbf{v}}_i \in \hat{V} \\ \mathbf{v}_i(\hat{r}_{ib}, \hat{t}_{ib}) = \hat{\mathbf{v}}_i(\hat{r}_{ib}, \hat{t}_{ib}) + \delta\hat{\mathbf{v}}_i \\ \qquad\qquad\quad = \hat{r}_{ib}\mathbf{r}_{ib} + \hat{t}_{ib}\mathbf{t}_{ib} + \delta\hat{\mathbf{v}}_i \end{cases} , \text{ for } \mathbf{v}_i \in V. \tag{7}$$

with respect to the Cartesian origin. We attach the origin to $\hat{\mathbf{v}}_i$ in Fig.4 for visualization clarity. Since we assume the template mesh is of good quality and has no zero-area cells so the collinear $(\mathbf{r}_{ib}, \mathbf{t}_{ib})$ is not considered. Then by considering M as a discretization of an elastic material that is framed by fixed M^S and M^{IO}, we define the energy function at the vertex $\mathbf{v}_i$ to measure the mesh distortion by summing up the squared Frobenius norm of the strain and its spatial derivative, which writes

$$\mathbb{E}_{ib} = \left\|\nabla_{X_{ib}}\, \Delta V\right\|_F^2 + \left\|\nabla_{X_{ib}}(\nabla_{X_{ib}}\, \Delta V)\right\|_F^2 = \left\|\frac{\partial \delta\mathbf{v}_i}{\partial r_{ib}}\right\|^2 + \left\|\frac{\partial \delta\mathbf{v}_i}{\partial t_{ib}}\right\|^2 + 2\left\|\frac{\partial^2 \delta\mathbf{v}_i}{\partial r_{ib}\partial t_{ib}}\right\|^2 + \left\|\frac{\partial^2 \delta\mathbf{v}_i}{\partial r_{ib}^2}\right\|^2 + \left\|\frac{\partial^2 \delta\mathbf{v}_i}{\partial t_{ib}^2}\right\|^2 . \tag{8}$$

This formulation is in the same spirit as the one used Active Surface Models (ASMs) that has been extensively studied in computer vision and computer graphics [45–48].

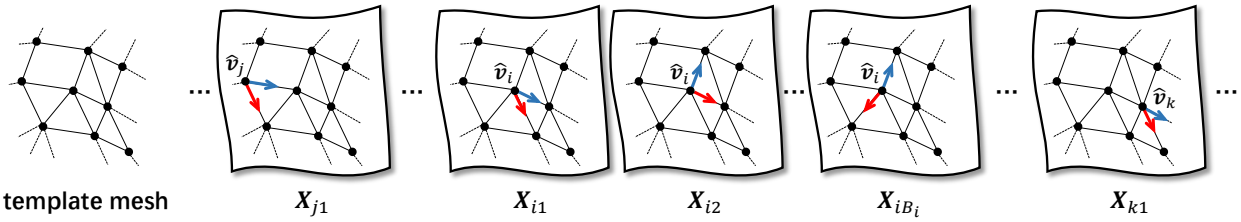

Fig. 5 The multiple skewed coordinate systems build on different vertex and pairs of neighbors.

To prove $\hat{M}$ is a stationary point of $\mathbb{E}_{ib}$, we look at the following formulation

$$-\frac{\partial}{\partial r_{ib}}\left(\frac{\partial \delta \mathbf{v}_i}{\partial r_{ib}}\right)-\frac{\partial}{\partial t_{ib}}\left(\frac{\partial \delta \mathbf{v}_i}{\partial t_{ib}}\right)+2\frac{\partial^2}{\partial r_{ib}\partial t_{ib}}\left(\frac{\partial^2 \delta \mathbf{v}_i}{\partial r_{ib}\partial t_{ib}}\right)+\frac{\partial}{\partial r_{ib}^2}\left(\frac{\partial^2 \delta \mathbf{v}_i}{\partial r_{ib}^2}\right)+\frac{\partial}{\partial t_{ib}^2}\left(\frac{\partial^2 \delta \mathbf{v}_i}{\partial t_{ib}^2}\right). \tag{9}$$

Since $\delta \mathbf{v}_i = \mathbf{v}_i - \hat{\mathbf{v}}_i$ and $\partial^2 \hat{\mathbf{v}}_i / \partial r_{ib}^2 = \partial^2 \hat{\mathbf{v}}_i / \partial t_{ib}^2 = 0$ with the definition in Eq. 7, it can be rewritten as

$$F_{ib} := -\frac{\partial}{\partial r_{ib}}\left(\frac{\partial \mathbf{v}_i}{\partial r_{ib}}\right)-\frac{\partial}{\partial t_{ib}}\left(\frac{\partial \mathbf{v}_i}{\partial t_{ib}}\right)+2\frac{\partial^2}{\partial r_{ib}\partial t_{ib}}\left(\frac{\partial^2 \mathbf{v}_i}{\partial r_{ib}\partial t_{ib}}\right)+\frac{\partial}{\partial r_{ib}^2}\left(\frac{\partial^2 \mathbf{v}_i}{\partial r_{ib}^2}\right)+\frac{\partial}{\partial t_{ib}^2}\left(\frac{\partial^2 \mathbf{v}_i}{\partial t_{ib}^2}\right). \tag{10}$$

$F_{ib} = 0$ for $\forall\, i,\ b$ when $\mathbf{v}_i = \hat{\mathbf{v}}_i$ according to the definition of $\mathbf{r}_{ib}$ and $\mathbf{t}_{ib}$ in Eq.7, which is the Euler-Lagrange equation of $\mathbb{E}_{ib}$ and is the sufficient and necessary condition of our claim. To regularize the computational mesh, the magnitude of F_{ib} should be as small as possible to reach the stationary point of $\mathbb{E}_{ib}$. Since a single $\mathbb{E}_{ib}$ only describes the energy locally and at a specific orientation, we define multiple coordinates systems similarly on all vertices and with all combinations of outgoing edges , as shown in Fig.5, as well as their corresponding energy functions and Euler-Lagrange equations. This applies to all vertices in the computational mesh V^{Com} that are not on the airfoil profile (V^S) nor on the in/outflow boundaries (V^{IO}), namely $V^{Com} = V \setminus (V^S \cup V^{IO})$. We take the regularization term $\mathcal{L}_{reg}$ to be the averaged squared F_{ib} as

$$\mathcal{L}_{reg} := \frac{1}{|V^{Com}|}\sum_{i=1}^{|V^{Com}|}\frac{1}{B_i}\sum_{b=1}^{B_i}||F_{ib}||_2^2\ ,\ \text{where}\ \ V^{Com} = V \setminus (V^S \cup V^{IO})\ . \tag{11}$$

Minimizing $\mathcal{L}_{reg}$ yields a deformed computational mesh whose local cell structure is similar to that of $\hat{M}$. This eliminates serious issues such as overlapped facets, and desirable mesh properties embedded in $\hat{M}$, such as the cell's aspect ratio, skewness and orthogonality, are preserved as much as possible. Using a dedicated coordinate system for every vertex and every combination of its neighbors makes $\mathcal{L}_{reg}$ to work with arbitrary cell types.

The derivatives of Eq. 10 are estimated by finite differences. This implies computing the perturbation of vertices. As shown in Fig.4(c), positive perturbations are sampled directly along the edges as

$$\Delta \mathbf{r}_{ib}^{+} = \epsilon \frac{\mathbf{v}_{ib}^{r} - \mathbf{v}_i}{||\hat{\mathbf{v}}_{ib}^{r} - \hat{\mathbf{v}}_i||_2}\ \ \text{and}\ \ \Delta \mathbf{t}_{ib}^{+} = \epsilon \frac{\mathbf{v}_{ib}^{t} - \mathbf{v}_i}{||\hat{\mathbf{v}}_{ib}^{t} - \hat{\mathbf{v}}_i||_2}\ , \tag{12}$$

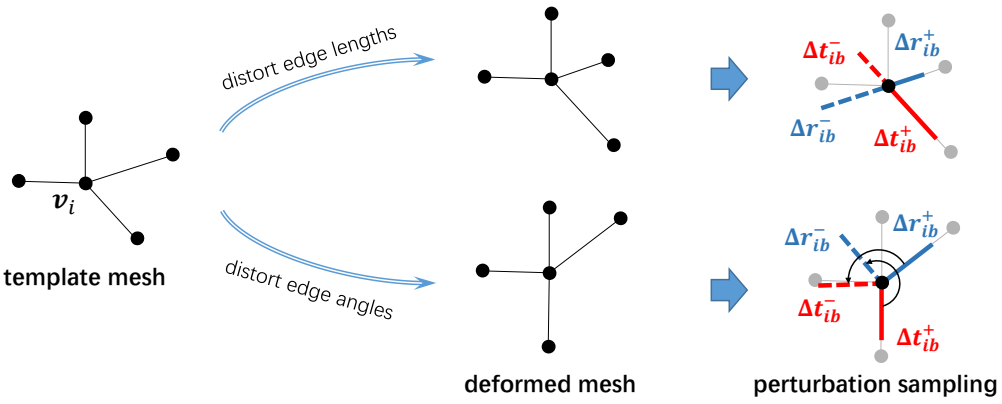

Fig. 6 **Mesh distortion changes cell structures and results in nonzero $\mathcal{L}_{reg}$.**

where $\mathbf{v}_{ib}^r$ and $\mathbf{v}_{ib}^t$ are $\hat{\mathbf{v}}_{ib}^t$ and $\hat{\mathbf{v}}_{ib}^t$ after the deformation. Negative perturbations $\Delta\mathbf{r}_{ib}^-$ and $\Delta\mathbf{t}_{ib}^-$ are rotated from $\Delta\mathbf{t}_{ib}^+$ by $\pi - \alpha$ and $\Delta\mathbf{s}_{ib}^+$ by $\pi + \alpha$, respectively, where α is the angle between the base vectors $(\mathbf{r}_{ib}, \mathbf{t}_{ib})$, i.e. $\alpha = atan2(|\mathbf{r}_{ib} \times \mathbf{t}_{ib}|, (\mathbf{r}_{ib} \cdot \mathbf{t}_{ib}))$. We keep the positive scalar ϵ small to correctly approximate the derivatives. The edge normalizers (denominators in Eq. 6 and 12) and the rotating angle α are calculated on the template mesh, and remain constant even when the mesh is distorted. Using the proposed coordinate system and the sampling strategy avoids complicated interpolation inside cells. We provide additional details about the finite difference derivative computation in Appendix.B and show that they yield a valid approximation of the derivatives in Appendix.C. As shown in Fig.6, any type of deformations from the template mesh will leave the stationary point $\hat{M}$ since changes of edge length ratios and angles make Eq.10 nonzero.

With this approach of approximating derivatives, each F_{ib} can be written as a linear function of the position of $\mathbf{v}_i$ and its neighboring vertices. Eq.11 can thus be rewritten as a numerical approximation with a matrix form as

$$\mathcal{L}_{reg} := ||K\mathbf{v}||_2^2 , \tag{13}$$

where $\mathbf{v}$ is the vectorized form of V_{Com}, where $V^{Com} = V \setminus (V^S \cup V^{IO})$, and K is the discretization matrix to implement the finite difference approximation. K is only dependent on the template mesh and is only generated once for the given template mesh $\hat{M}$. Thanks to the linearity in Eq.11, building multiple coordinate systems results in a single sparse K matrix and does not increase the computational cost significantly.

Deformation Decay Loss $\mathcal{L}_{dec}$. We decay the vertices' motion and constrain the model to generate the minimal necessary deformation

$$\mathcal{L}_{dec} = \frac{1}{|V|} \sum_i \|\delta\mathbf{v}_i\|_2^2 . \tag{14}$$

Regularization Loss on Latent Space $\mathcal{L}_z$. Additionally minimizing

$$\mathcal{L}_z = \frac{1}{T}\sum_t \|\mathbf{z}_t\|_2^2 \tag{15}$$

amounts to assuming that the prior distribution of latent vectors is a gaussian centered around zero and helps regularize the problem. In other words, $\mathcal{L}_z$ promotes smoothness of the latent space.

C. Network

The Structure of Auto-Decoder. We take the latent code $\mathbf{z}$ of Eq. 1 to be a 256-dimensional vector. The auto-decoder comprises 4 graph convolution layers [49] and a fully connected output layer. Each graph convolution is applied with weight normalization [50] and relies on the ReLU nonlinear activation [51]. By using the term $g_\Theta(\mathbf{z}; \hat{M})$, the auto-decoder concatenates the latent code at the first layer and the vertex feature in other layers with coordinates of template vertices as input at every graph convolution, and uses edge information for message passing. Using the template vertex coordinates as inputs provide inductive bias of spatial continuity that helps the model generates smooth deformations. The fully connected layer is applied on each vertex feature vector and predicts a 2D vertex displacement.

The Active Model Layer. At inference time, an Active Model Layer (AM layer) is used to further refine the mesh quality at a finer granularity. The AM layer minimizes the same Eq.13 and in an implicit post-processing approach. Given that K is not invertible, the equation can be solved iteratively as

$$\lambda(\mathbf{v}^t - \mathbf{v}^{t-1}) + K\mathbf{v}^t = 0 \Rightarrow (K + \lambda I)\mathbf{v}^t = \lambda \mathbf{v}^{t-1}\,, \tag{16}$$

where I is the identity, the superscript t is the iteration step and λ is the iteration step size. The initial state comes from the auto-decoder's direct output as $V^0 = \hat{V} + \Delta V$. Due to the large size and the sparsity of K, inverting $(K + \lambda I)$ precisely is impractical. Instead, we use the Neumann series with

$$(K + \lambda I)^{-1} \approx \sum\nolimits_{n=0}^{K} (-1)^n \left(\frac{1}{\lambda}\right)^{n+1} K^n\,. \tag{17}$$

The AM layer is necessary to eliminate minor mesh quality issue occured in auto-decoder's direct output. We demonstrate in Sec.IV.C that combining the minimization of $\mathcal{L}_{reg}$ and the AM layer works most effectively and efficiently.

D. Training Details

Training Set. For training purposes, we collected a set of 1100 airfoil profiles, $T = 1000$ for training and 100 for testing. We use NACA's 4-digit and 5-digit airfoils and sample random digits. We rely on the MATLAB formulations*†

*http://www.mathworks.com/matlabcentral/fileexchange/19915-naca-4-digit-airfoil-generator
†http://www.mathworks.com/matlabcentral/fileexchange/23241-naca-5-digit-airfoil-generator

to generate the profiles. Chord lengths are normalized to 1, and the leading and trailing edges are fixed at $(0, 0)$ and $(1, 0)$, respectively. Each training profile S_t is a collection of 600 2D points sampled on the surface of the airfoil. Note that topology no information is in S_t, which enables the model to work on various type of data format, and reduces the demand for complicated data pre-processing.

Template Mesh. To obtain the template mesh $\hat{M}$ of Section. III.A, we start from the NACA-0012 profile because it is the most commonly studied airfoil. We use a manually generated quadrilateral mesh [‡] as the default template mesh. It contains $116,240$ vertices and $230,920$ faces. The computational boundaries are at least 14 chords away from the airfoil. The mesh is extruded and there is only one cell along the z axis. As shown in Fig. 2, we reduce the required amount of computation by fixing some of the exterior vertices and only allowing those closer to the profile to move.

Unless otherwise specified, the quadrilateral mesh in mention is used in experiments by default, denoted as TM-A. The proposed model also works well on other types of meshes, even without any adaptation on the network after training, which will be further discussed in Sec.IV.B.

Implementation Details. We use the Pytorch [52] library for implementation. During training, we used the Adam optimizer [53] to solve Eq. 2 with a learning rate of 5×10^{-4} for the decoder weights and 10^{-3} for the latent codes. The training batch size was 5 and we ran 20 training epochs. We took the weights of Eq. 4 to be $w_{dist} = 3 \times 10^3, w_{reg} = 10^3, w_{dec} = 10^{-3}, w_z = 10^{-4}$. At inference time, the decoder remains fixed while the latent code for a reference geometry is initialized with Gaussian noise. It is then optimized with 800 gradient descent steps and a learning rate of 5×10^{-4}.

IV. Experiments

We validate our proposed model on 2D airfoil geometries. We show it outperforms traditional mesh representations in (i) reducing the amount of handcraft designs, (ii) discovering geometric patterns automatically, (iii) representing both surfaces and entire CFD meshes and (iv) providing full differentiability to downstream applications.

A. Mesh Representation and Reconstruction

In this experiment, we evaluate the quality of our mesh representations. We encode multiple airfoils into latent codes using Eq.3. No manual adjustments are required when working with different airfoils. The embedding quality is measured by the accuracy of surface reconstruction decoded from the latent codes.

To do this, we randomly chose 50 NACA airfoils outside the training set. We also use another 50 non-NACA airfoils, to investigate the model's generalization ability. These non-NACA airfoils belongs to the AG and RG series, plus the low-Reynolds-number Eppler airfoils, which are collected from the UIUC airfoil dataset [54]. Since most airfoil

[‡]http://www.wolfdynamics.com/tutorials.html?id=148

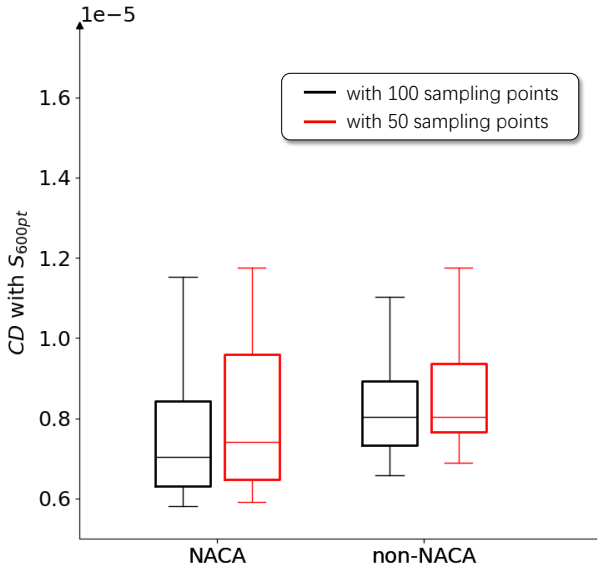

Fig. 7 The *Chamfer Distance* (CD) between the reconstructed airfoils and the 600-point reference geometries.

geometric data publicly available samples a surface with 50 to 100 points, we sample 100 and 50 surface points on each airfoil as the target geometries to test under different sampling qualities. We use 600 sampling points S_{600pt} on an airfoil as a lossless geometry. The reconstruction error is calculated by the Chamfer Distance [55] (CD) between the deformed airfoil surface and S_{600pt}, which is defined as

$$CD(V^S, S_{600pt}) = \frac{1}{|V^S|} \sum_{v_i \in V^S} \min_{s_i \in S_{600pt}} ||v_i - s_i||^2 + \frac{1}{|S_{600pt}|} \sum_{s_i \in S_{600pt}} \min_{v_i \in V^S} ||v_i - s_i||^2 . \tag{18}$$

Since V^S and S_{600pt} are discrete sampling of a continuous airfoil profile, CD remains positive even when two shapes are exactly registered.

Fig.7 shows the reconstruction accuracy over the 50 test airfoils. The mean CDs are 8.9×10^{-6} and 9.3×10^{-6} for 100-point and 50-point S over NACA airfoils, and those are 8.3×10^{-6} and 8.7×10^{-6} for non-NACA airfoils. The errors are very small. We visualize some reconstruction results in Fig.8. The figure shows that the reported mean CDs indicate high-fidelity reconstructions of both airfoil types and under both sampling qualities. At the same time, the statistics and the qualitative results show little performances difference between NACA and non-NACA airfoils. The proposed model can work well with other shapes that are similar to the training geometries without additional adaptations. For all test settings, the average time cost to encode and reconstruct a given airfoil is $30.1s$ on a standard PC with an NVIDIA V100 GPU card.

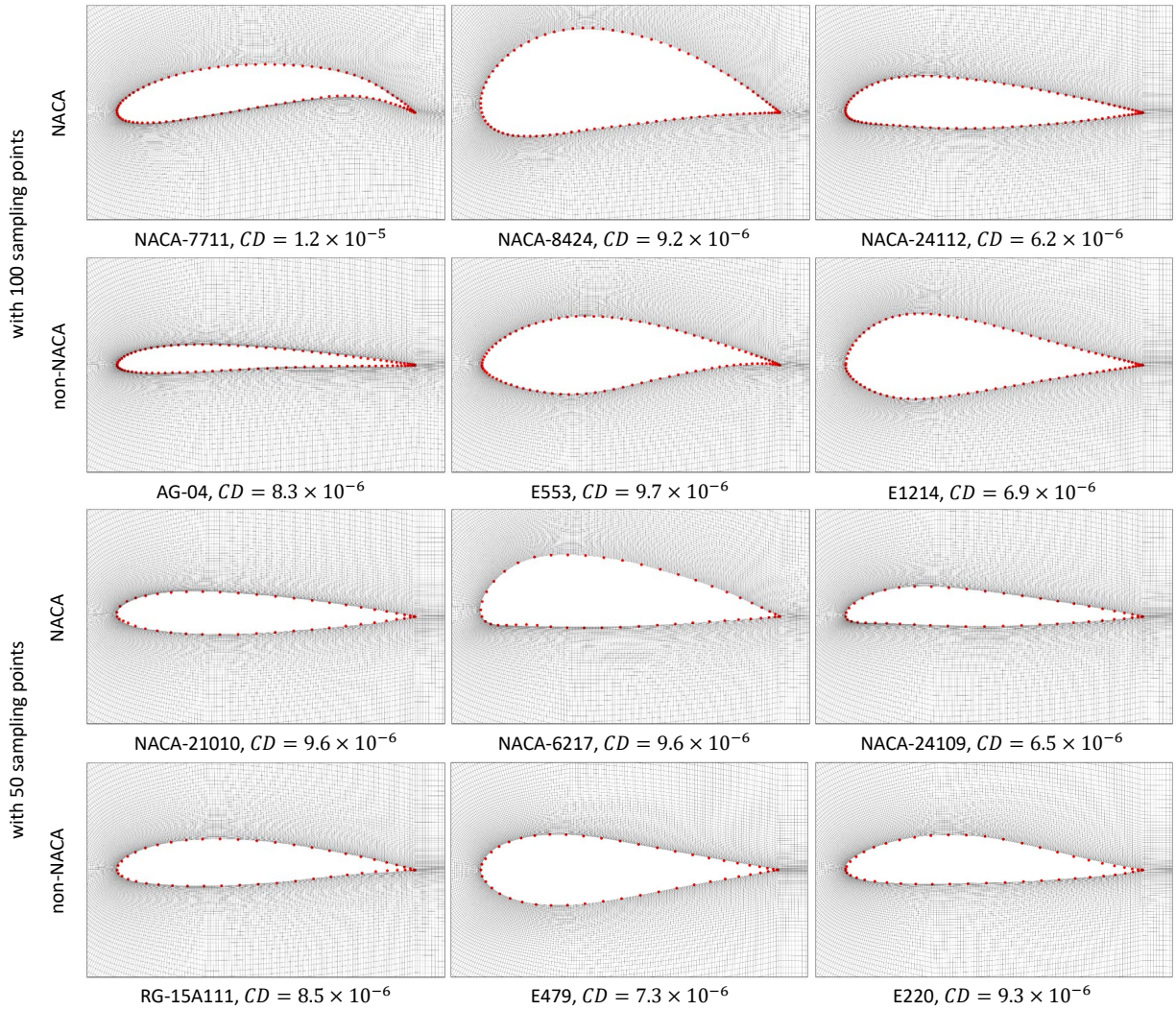

Fig. 8 The reconstructions of various airfoil series with different numbers of sampled points, i.e. the red dots. CDs are shown.

B. Using Different Template Meshes

We found that the auto-decoder trained with TM-A works well on other types of template mesh, even without additional adaptations after training. It means that, given a different template mesh $\tilde{M} = \{\tilde{V}, \tilde{E}\}$ and the auto-decoder $g_{\Theta*}$ trained with template mesh $\hat{M}$, we can still well solve

$$\mathbf{z}^* = \underset{\mathbf{z}}{\operatorname{argmin}} \mathcal{L}(\tilde{M}(\tilde{\mathbf{v}} + g_{\Theta^*}(\mathbf{z}, \tilde{M})), S) \ . \tag{19}$$

To test he model's generalization ability, we use four different template meshes.

- **The hybrid parametric mesh (denoted as TM-B)** is generated by VLab [56, 57], as shown in Fig.9. It comprises a triangulated part that divides the far field and well refined wall layers that fully resolves the boundary region. It

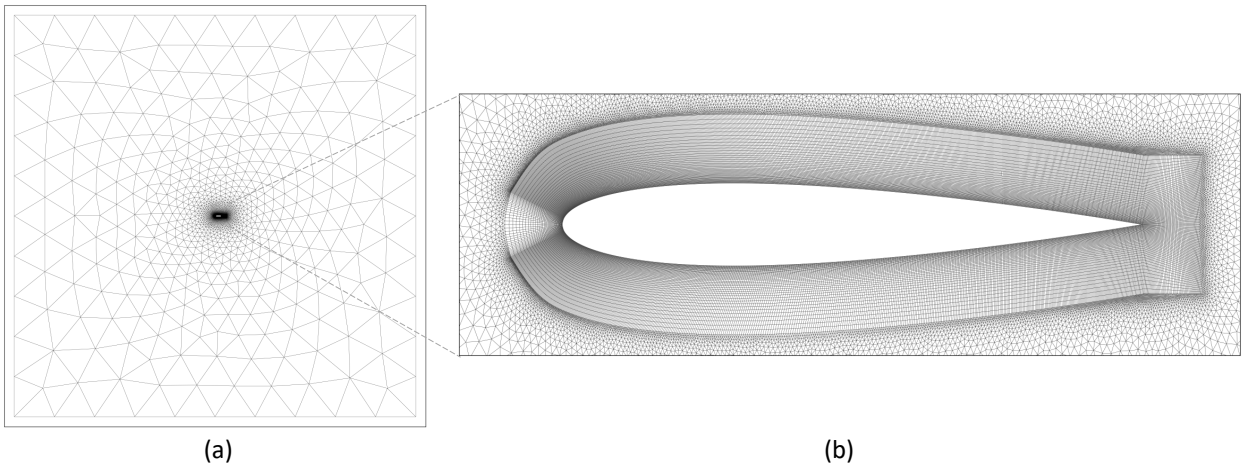

Fig. 9 The hybrid parametric template mesh (TM-B) with (a) an overall and (b) a zoomed-in views.

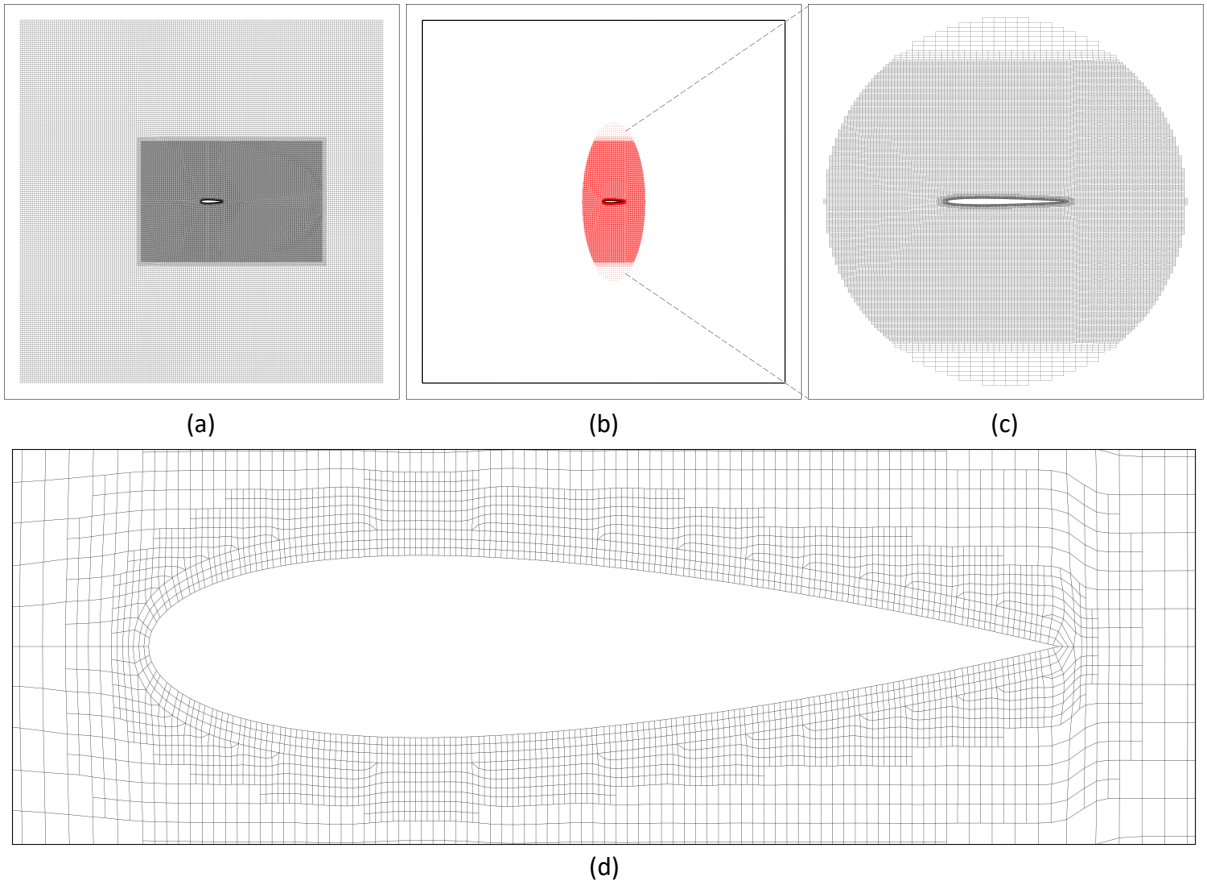

Fig. 10 The template blockmesh TM-C with views of its (a) overall mesh, (b) boundary, (c) deformation area and (d) surface.

contains 64, 917 vertices and 216, 793 edges belonging to triangles and rectangles.

- **The block mesh (denoted as TM-C)** is generated by the OpenFoam's *blockMesh* command on the NACA-0012

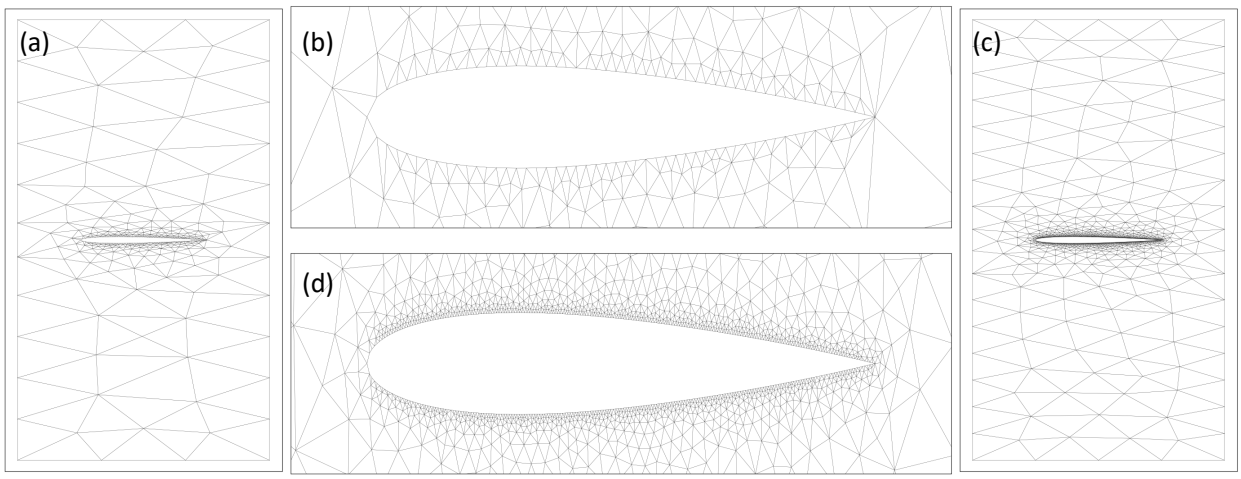

Fig. 11 The (a-b) coarse and (c-d) fine triangulated template meshes generated by Gmsh.

airfoil, as shown in Fig.10. The *snappyHexMesh* options are used during meshing so that there are four mesh granularities and viscous layers, which leads to a mixture of cell types and various numbers of neighbors for vertices. Similar to TM-A, we also select a subarea surrounding the airfoil to deform. The deformation area contains 30,536 vertices and 29,848 polygon cells with 3, 4 or 5 edges.

- **The triangulation meshes** are generated by the Gmsh library [58]. The cells are all triangulated. We use this meshing to studies the effect of different mesh densities. To do this, we create a coarse mesh (denoted as TM-D1) with 347 vertices and 564 faces (see Fig.11(a)), and a fine mesh (denoted as TM-D2) with 1,642 vertices and 2,842 faces (see Fig.11(b)). The overall mesh is relatively small so no subarea extraction is needed. The template airfoil is also NACA-0012.

Given these template meshes, TM-B is used for simulations in the following section, while TM-C, TM-D1 and TM-D2 are only used to investigating our model's generalization ability to different geometries.

Four random NACA airfoils and four random non-NACA airfoils are reconstructed using these different template meshes and Eq.3. No severe mesh quality issues are found in any of these results. A qualitative comparison near the airfoil is presented in Fig.12 where the special structures created by the meshing algorithms in the template mesh are well preserved in all cases. The simulation results of Sec.IV.C.2 also indicate that meshes based on TM-B are of good quality.

C. Computational Mesh Quality

We now investigate the effectiveness of representing the entire computational mesh in the context of CFD. To do this, we first use the OpenFOAM's mesh checking tool to evaluate the mesh quality after deformation. Then simulations are performed on the generated meshes to analyze the effect of mesh reconstruction on aerodynamic performance.

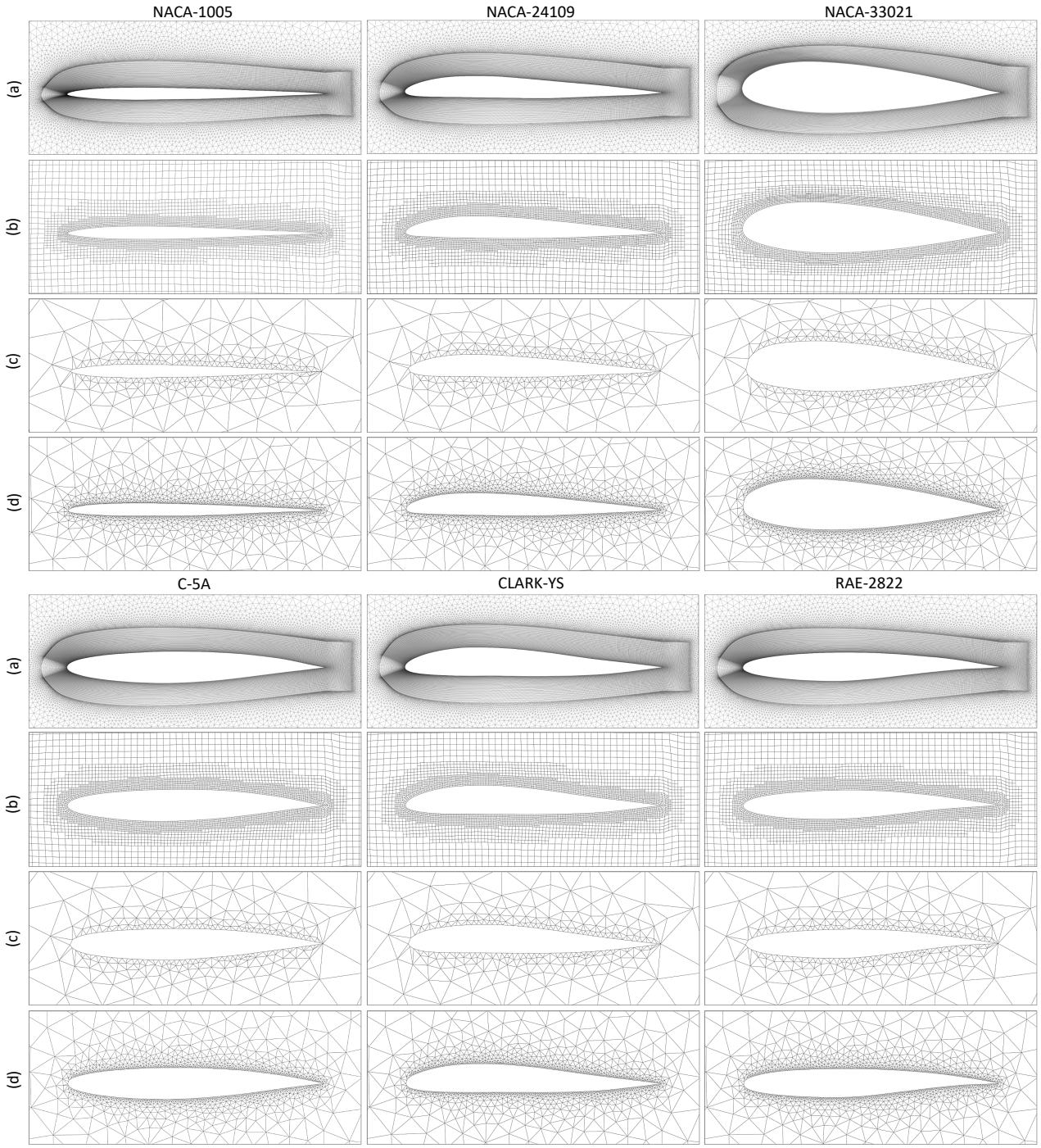

Fig. 12 Mesh reconstructions inferred with (a) TM-B, (b) TM-C, (c) TM-D1 and (c) TM-D2. Best viewed with zoom-in.

1. Quantitative Evaluations on Mesh Quality

We use the 50 random NACA airfoils and 50 random non-NACA airfoils of any series from the UIUC airfoil dataset. To study the role of $\mathcal{L}_{reg}$ and AM Layer in preserving the mesh quality, all meshes are reconstructed from TM-A with the proposed model (Model #4) and the three other variants (Model #1-3):

- **Model #1** is trained without $\mathcal{L}_{reg}$ and the AM Layer is not used. It serves as a baseline model.
- **Model #2** is trained without $\mathcal{L}_{reg}$, but the AM layer is used as post-processing. In this case, it takes more than $8,000$ iterations for Eq.17 to converge on a 2D airfoil mesh.
- **Model #3** uses $\mathcal{L}_{reg}$ for training but the AM layer is not used.
- **Model #4** uses both $\mathcal{L}_{reg}$ for training and the AM layer for inference. The number of iteration can be largely reduced to 240. Model #4 is the default setting of the proposed model.

Table 1 Metrics used in the volumetric mesh quality check.

issues		***errors***		***qualities***
mesh	E1	number of negative volume cells		
overlapping	E2	number of incorrectly oriented faces		
skewness	E3	number of highly skewed faces	Q1	max skewness
orthogonality	E4	number of non-orthogonality errors	Q2	max non-orthogonality
			Q3	mean non-orthogonality
			Q4	number of severely non-orthogonal faces (> 70 degrees)

Table 2 Average mesh quality evaluation results over 100 random airfoils.

airfoils	models	E1	E2	E3	E4	Q1	Q2	Q3	Q4
NACA	#1	364.74	2245.54	535.40	476.82	12348.725	178.861	12.607	277.66
	#2	0.00	0.08	0.02	0.06	0.771	49.424	9.017	2.40
	#3	0.00	534.12	3.86	109.96	480.298	137.272	9.852	44.42
	#4	0.00	0.00	0.00	0.00	0.543	42.668	9.020	6.30
non-NACA	#1	435.98	2657.66	513.80	572.02	2031.345	179.113	13.675	353.48
	#2	0.00	0.30	0.06	0.08	0.803	51.442	9.100	10.26
	#3	0.00	65.32	0.40	10.92	11.222	101.035	8.646	16.52
	#4	0.00	0.00	0.00	0.00	0.486	36.073	8.292	0.00
overall	#1	400.36	2451.60	524.60	524.42	7190.035	178.987	13.141	315.57
	#2	0.00	0.19	0.04	0.07	0.787	50.433	9.059	6.33
	#3	0.00	299.72	2.13	60.44	245.760	119.154	9.249	30.47
	#4	0.00	0.00	0.00	0.00	0.514	39.371	8.656	3.15
NACA-0012	TM-A	0.00	0.00	0.00	0.00	0.486	30.144	7.520	0.00

OpenFoam's *checkMesh* command is used for evaluation. This tool generates a short report that contains several quantitative results. We divide these statistics into two categories, i.e. *errors* and *qualities*. Any occurrence of *errors* in the report indicate existed fatal issues that impair the simulation's correctness. The *quality* results are geometric criteria of the mesh that affect the stability, convergence speed and residual control of simulations. We use 8 measurements in total for a comprehensive evaluation. Tab.1 explains the meanings of all metrics and lower values indicate better mesh qualities. The evaluation results are compared with the template mesh quality in Tab.2. Several qualitative comparisons are shown in Fig.13.

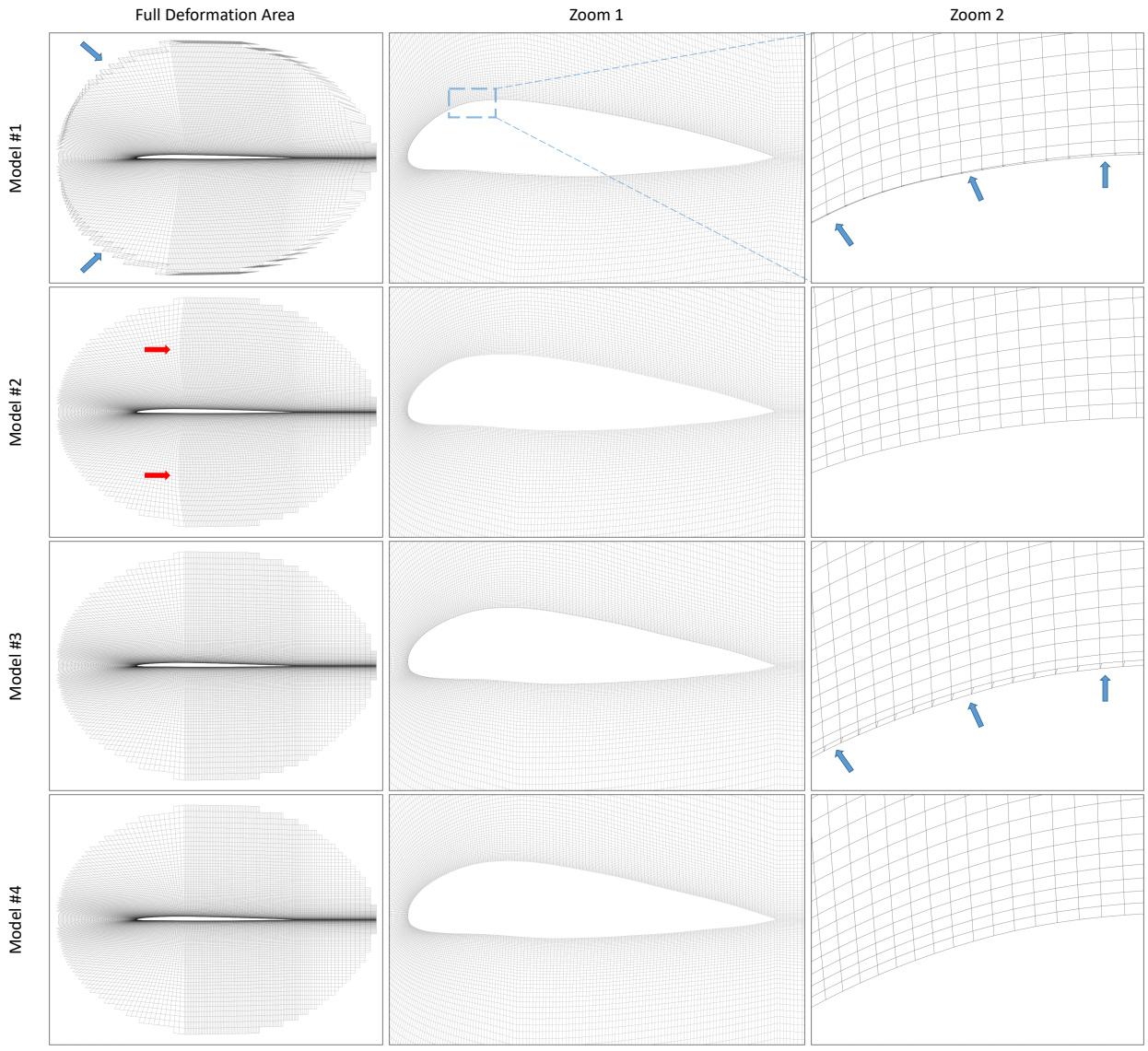

Fig. 13 Illustrations of NACA-24109's meshes generated by Model #1-4. Blue and red arrows highlight the overlapping and non-orthogonal issues.

Model #1 has no guarantee on the mesh quality without any explicit constrains or post-processing. The results of Model #2 show that the AM layer solves almost all the overlapping issues and other severe errors, but some global non-orthogonality remains. Using $\mathcal{L}_{reg}$ for training while removing the AM layer as in Model #3 removes the large mesh distortion near the boundary of the deformation area, as shown in Fig.13, but the overlapping issue near the object surface is not totally solved. However, the combination of $\mathcal{L}_{reg}$ and AM layer as in Model #4 is most effective. Meanwhile, Model #4 works consistently well on NACA and non-NACA airfoils, which demonstrates its robust generalization ability. Compared with the NACA-0012 template mesh TM-A, Model #4 produces meshes without errors and with the same order of magnitude for quality metrics Q1-Q4. It explains the proposed model with toh $\mathcal{L}_{reg}$ and the AM layer is suitable for CFD, as it produces meshes of similar quality as a handcrafted one, in an automated manner.

2. Case Studies on Simulations

In this section, we validate the quality of our generated meshes when used to simulate 2D airfoil dynamics. To this end, we compare simulation results obtained using our meshes with experimental data from the UIUC Low Speed Airfoil Data [59–62] and the CFD data generated automatically by the VLab computational framework [56, 57].

The UIUC Low Speed Airfoil Data features airfoil tests at low-Reynold numbers in the UIUC wind tunnel. This dataset contains testing results on various airfoils. We select the data of lift coefficients at $Re = 10^5$ of three NACA airfoils (i.e. NACA-0009 / 2414 / 6409) and three non-NACA airfoils (i.e. AG-24, ClarkY and SD-6060) as references.

The VLab data is composed of high-fidelity numerical simulations created by the computational framework VLab [56, 57]. It uses parametric hybrid meshes and ANSYS® Fluent® to provide CFD of various shapes in an automatic way.We refer to NACA-2412 and NACA-8412.

For the UIUC Low Speed dataset comparisons, the simulations were performed by running OpenFoam on computational meshes generated from TM-A for the reference airfoils. We use the RANS solver coupled with the k-omega-SST turbulence model [63] at $Re = 10^5$. For the VLab data comparison, meshes for both airfoils are generated based on TM-B. The CFD meshes are then used in VLab's solver for RANS simulations at $Re = 10^6$. Both the Spalart–Allmaras [64] and the k-omega-SST turbulence models are used for the respective simulations, allowing for the study of mesh effects by comparing discrepancies caused by different meshes and turbulence models.

Fig.14 shows our simulation data at AoAs of $-5°$, $-2.5°$, $0°$, $2.5°$, $5°$ and $7.5°$. The simulated lift coefficients fit well with the wind tunnel experimental data. The averaged y+ values of these cases range from 1.94 to 2.49, and the mean y+ when simulating on TM-A is 2.19. In Fig.15, the simulated results resolve the trend of lift-drag ratio with the changes of lift coefficients. The averaged y+, for example the one of NACA-2412, is 1.47 while the y+ of simulating on TM-B is 1.42. These results confirm that the deformed meshes decoded by the proposed model have adequate quality for CFD simulations. The proposed mesh model well preserve the boundary layer qualities given both template meshes, and the errors have minimal effects on simulation results.

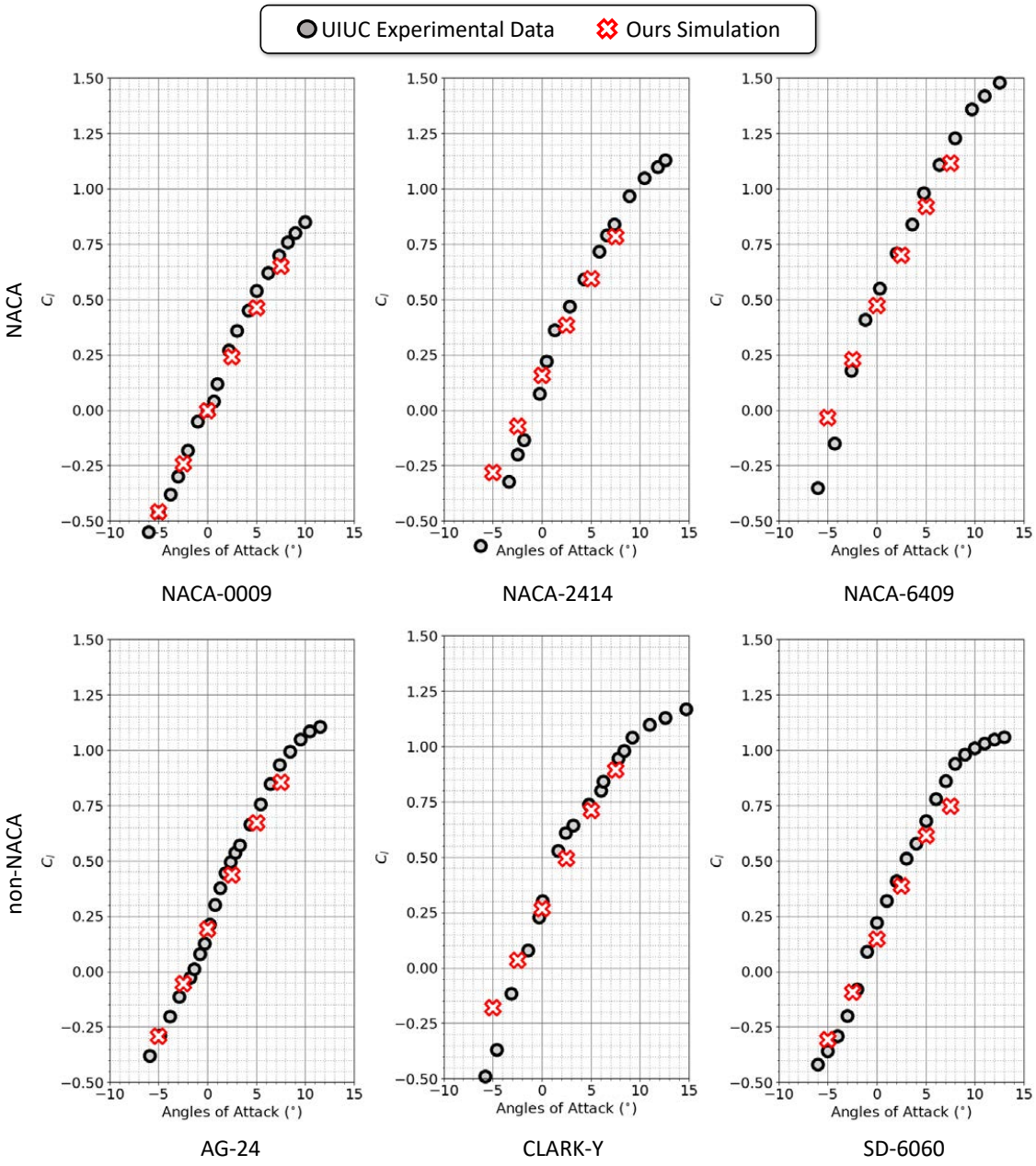

Fig. 14 Comparisons between the simulation results on deformed meshes and the UIUC's experimental data.

D. Full Differentiability for Downstream Applications

The proposed method provides full differentiability of the generated mesh with respect to its latent representation. It integrates end-to-end pipelines for downstream applications. Users can not only directly decode the latent code and obtain the mesh instantly, but also back-propagate gradients via the mesh to the latent vector so as to manipulate the geometry and computational mesh. Compared to step-by-step pipelines, the proposed model requires no hyper-parameter tuning in order to adapt to different applications or targeted geometries. This enables fast prototyping for downstream

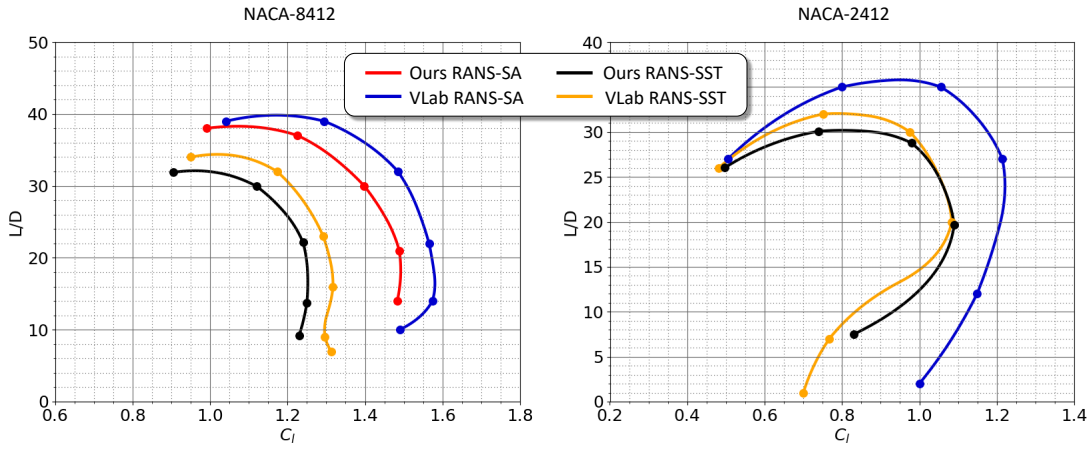

Fig. 15 Comparisons between the simulation results on deformed meshes and VLab reference data.

applications on novel and less studied objects.

Here, we demonstrate this idea and implement a fully differentiable pipeline for the 2D airfoil shape optimization task: given an initial 2D airfoil geometry S_{init}, obtain an optimized geometry V_{opt}^{S} so as to minimize the airfoil's drag. The role of our model is twofold. First is serves as a fully differentiable mesh representation, with a prior. Gradients from the surrogate model are passed directly to the latent code for shape manipulation. Second, it yields readily available CFD meshes of novel shapes for the surrogate model during optimization.

1. Shape Optimization for Various Geometries

Problem Statement. The objective of the optimization task in this study is to minimize the total drag of given initial airfoils. The optimization is conducted with an inviscid free stream at 0.85 Mach and all airfoils are at zero angles of attack. The governing equation is the 2D compressible Euler with a constant ratio of specific heats of 1.4.

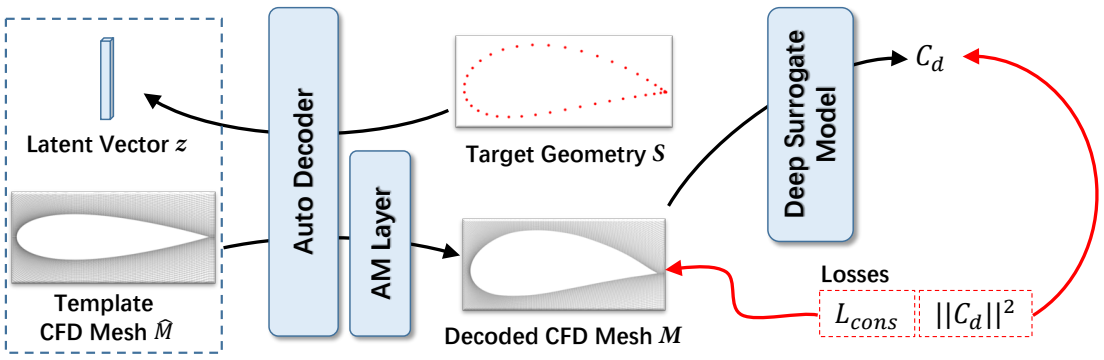

Fig. 16 The pipeline of shape optimization to minimize drag.

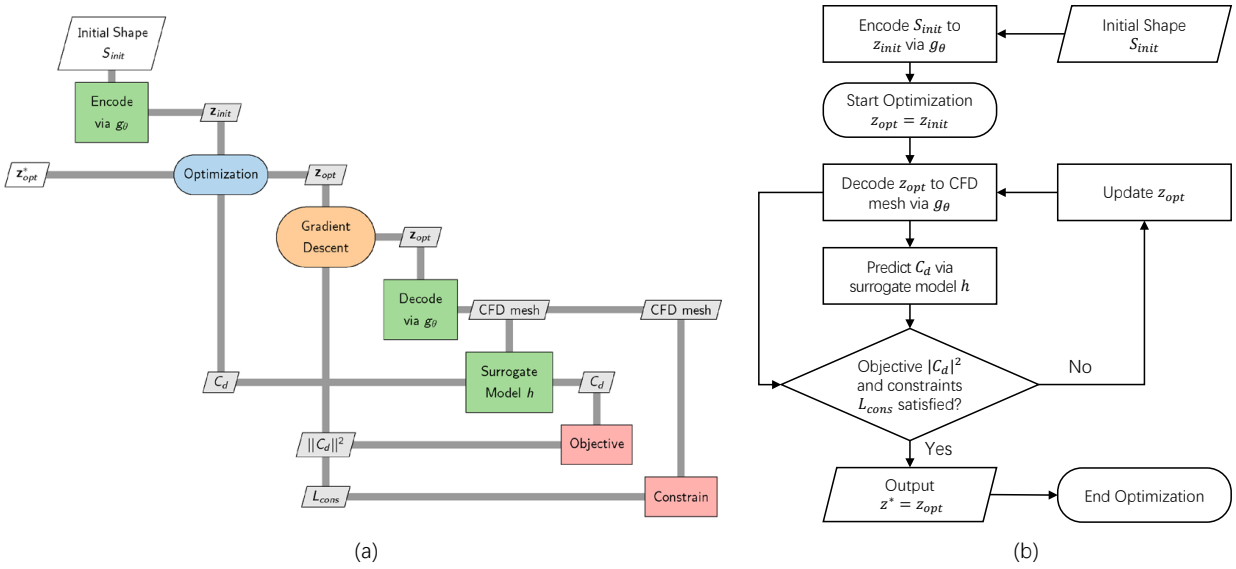

Fig. 17 The workflow of shape optimization shown in (a) the XDSM and (b) the flow chart.

Pipeline Design. To implement a shape optimization pipeline, a surrogate model is required to evaluation the CFD properties of an airfoil. There are multiple options for the surrogate model. For example, the user can utilize an adjoint CFD solver (e.g. ADflow[65], SU2[66], etc.) to generate gradients by directly simulating on the reconstructed CFD mesh. The user can also choose a deep learning model as the surrogate model. Here we follow [67] and use a graph convolutional neural network (GCNN) to predict the air pressure on the airfoil's surface. The pipeline is depicted by Fig.16. More details of the surrogate model can be found in *Appendix.D*. We follow the surrogate-based optimization (SBO) scheme and the GCNN surrogate model needs to be re-trained by adding newly optimized results into the training set. Our model reconstructs CFD meshes of new samples for CFD simulations.

Optimization. The workflow of optimization is shown in Fig.17. Given an initial airfoil, we first encode S_{init} into a latent vector $\mathbf{z}_{init}$ by Eq.3. The latent code to be optimized $\mathbf{z}_{opt}$ is initialized by $\mathbf{v}_{init}$. Then we fix the auto-decoder g_Θ as well as the trained GCNN surrogate model h, and optimize $\mathbf{z}_{opt}$ by minimizing the predicted drag coefficient C_d as

$$\mathbf{z}^*_{opt} = \underset{\mathbf{z}_{init}}{argmin}\, ||C_d||^2, \text{ where } C_d = h(\hat{V} + g_\Theta(\mathbf{z}_{init}), \hat{E}). \tag{20}$$

The optimized geometry can be extracted from the decoded mesh as

$$V^S_{opt} \in V_{opt} = \hat{V} + g_\theta(\mathbf{z}_{opt}). \tag{21}$$

The Adam optimizer is used. The maximum number of iteration is 300 to ensure the convergence.

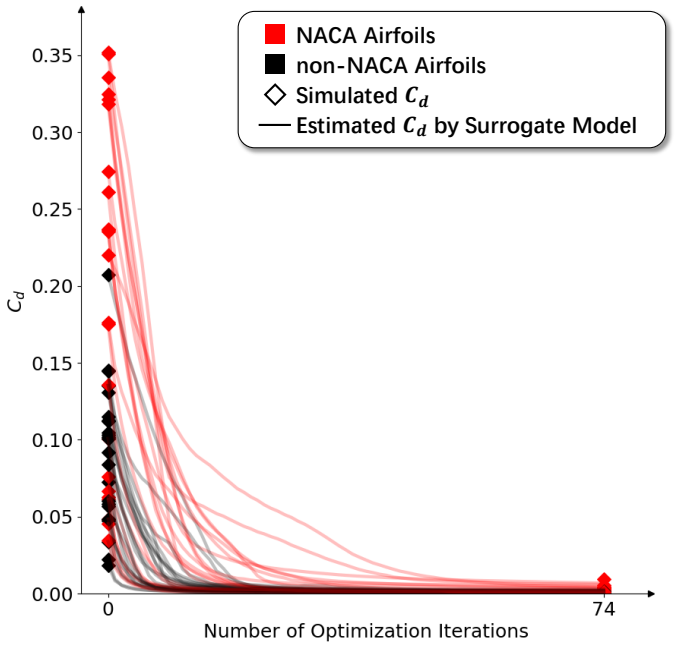

Fig. 18 The initial and optimized C_d of 25 NACA and 25 non-NACA test airfoils.

Results. We randomly collect 25 NACA and 25 non-NACA airfoils as initial shapes. Optimizations are run on all airfoils and we simulate with OpenFoam on the deformed shapes before the first iteration and after the last iteration to get their initial and optimized C_d values. Fig.18 shows that the C_ds have been significantly reduced in all cases. The model has consistent performances on NACA and non-NACA airfoils even when the mesh representation model is trained with NACA airfoils only. On both airfoil series, the optimization on a single case takes 3.7s – 4.2s.

Fig.19 depicts the evolution of the airfoils during unconstrained shape optimization in two different cases, one initialized with NACA-4219 and the other with E-169. The shapes evolve from the initial black airfoils and progress as the shapes change from pink to red in color.

2. Case Studies on Shape Optimization with Geometric Constraints

In practice, the shape optimization problem is often coupled with certain geometric constraints. These constraints are easy to integrate in our pipeline. As shown in Fig.16, the constraints can be written as differentiable formula and regarded as an extra loss function $\mathcal{L}_{cons}$. $\mathcal{L}_{cons}$ is applied directly on the deformed geometry. Eq.20 can be rewritten as

$$\mathbf{z}^*_{opt} = \underset{\mathbf{z}_{opt}}{argmin}(||C_d||^2 + w_{cons}\mathcal{L}_{cons}) \,. \tag{22}$$

In this section, we demonstrate optimization results with three different constraints, namely the bounding constraint,

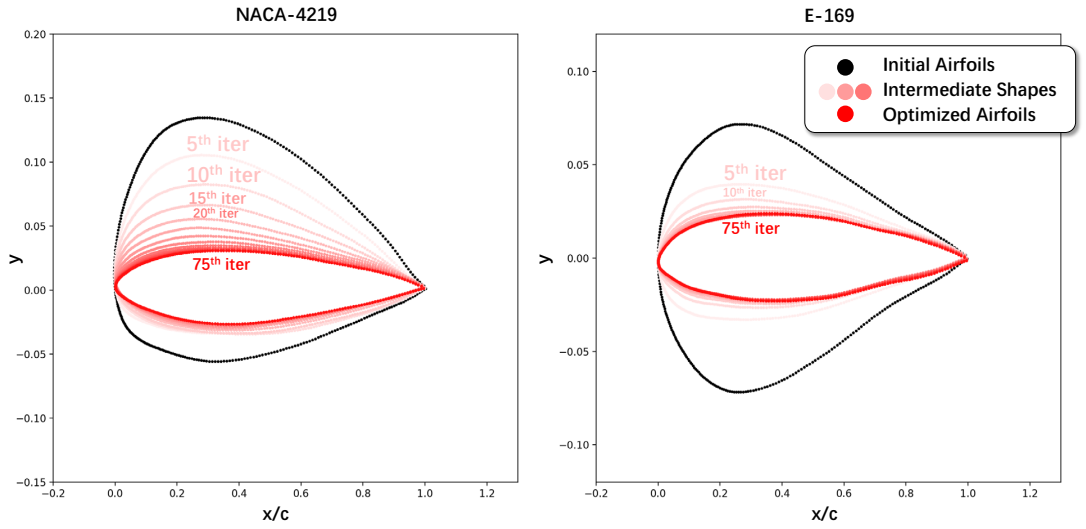

Fig. 19 The evolution of airfoils during unconstrained shape optimization. Shapes are plotted at every five iterations.

the maximum thickness constraint and the area constraint. The bounding constraint limits the airfoil's thickness to be no less than the initial one, which is defined as

$$\mathcal{L}_{cons} = \frac{1}{|V^S|}\sum_{i=1}^{|V^S|}\left\|\max(|y_i^{opt}| - |y_i^{init}|, 0)\right\|^2 \quad , \quad \text{where} \quad \begin{cases} v_i^{opt} = (x_i^{opt}, y_i^{opt}) \in V_{opt}^S \\ v_i^{init} = (x_i^{init}, y_i^{init}) \in V_{init}^S \end{cases} . \tag{23}$$

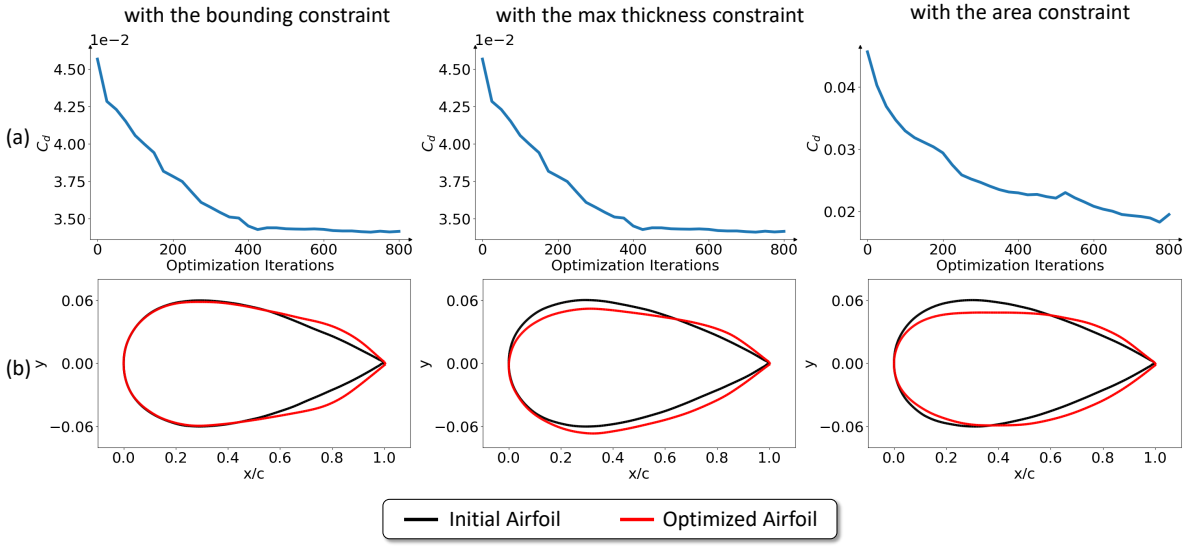

Fig. 20 Optimized NACA-0012 with different geometric constraints. (a) Changes of C_d. (b) Changes of the airfoils.

The maximum thickness constraint is a relaxed bounding constraint, which only limits the airfoil's maximal thickness.

To implement it, we use cubic splines, $spline^{+}(\cdot)$ and $spline^{-}(\cdot)$, to interpolate the surface positions on the upper airfoil and lower airfoil given a horizontal coordinate x, and then calculate the thickness by subtracting the vertical coordinates. Considering that airfoils' chord lengths are 1 and the leading edges are placed at the origin, then the constraint can be written as

$$\mathcal{L}_{cons} = \max_{x\in(0,1)} \left(\left\| \max((spline^{+}_{init}(x) - spline^{-}_{init}(x)) - (spline^{+}_{opt}(x) - spline^{-}_{opt}(x)), 0) \right\|^2 \right) . \tag{24}$$

The area constraint is to limit the optimized airfoil's area to be no less than the initial one. We use the Shoelace formula to compute the airfoil's area A. By sorting V^S in clockwise order, the constraint is

$$\mathcal{L}_{cons} = \left\| \max(A_{init}(V^S_{init}) - A_{opt}(V^S_{opt}), 0) \right\|^2 , \text{ where } A(V^S) = \frac{1}{2} \sum_{i=1}^{|V^S|} y_i(x_i - x_{i+1}) . \tag{25}$$

We use NACA-0012 as the initial airfoil and perform three optimizations with the geometric constraints in mention. The optimized results are demonstrated in Fig.20. In all cases the drag coefficients are reduced by a noticeable proportion and the geometric changes are explainable under the inviscid flow condition. The optimized airfoils move the shock waves towards its trailing edge under all constraints. They change the directions of surface normal near the trailing edge so as to counteract the drag forces generated at the leading edge. Meanwhile, the area constraint relaxes the limitation on airfoil's thickness so that the airfoil's leading edge is narrowed to reduce drag.

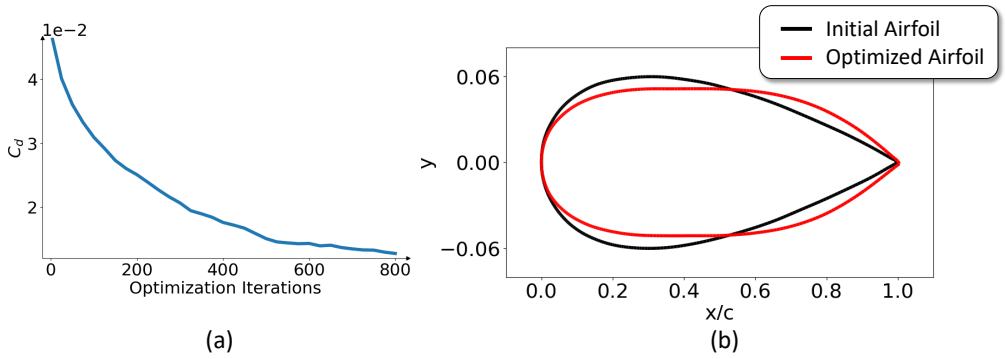

Fig. 21 Optimized NACA-0012 with the area and symmetry constraints. (a) Changes of C_d. (b) Changes of the airfoil.

The full differentiation of our mesh model, along with our end-to-end trainable pipeline, makes it possible to easily handle multiple constraints. For example, we combine the area constraint defined by Eq.25 and a symmetry constraint that ensures the consistency between airfoil's upper and lower curves. Then L_{cons} is the simple addition of both constraints, which writes

$$\mathcal{L}_{cons} = \left\|\max(A_{init}(V^S_{init}) - A_{opt}(V^S_{opt}), 0)\right\|^2 + \left\| \left|spline^+_{opt}(x)\right| - \left|spline^-_{opt}(x)\right| \right\|^2 , \quad \text{where } A(V^S) = \frac{1}{2}\sum_{i=1}^{|V^S|} y_i(x_i - x_{i+1}) \text{ and } x \in (0,1) . \tag{26}$$

The result of Fig.21 shows that the optimized shape satisfies both constraints without compromising the performance on reduced drag.

In summary, the results show that our model is effective for the fast prototyping of object shape designs and is able to work in a fully automatic fashion.

V. Conclusions

In summary, a latent representation model of CFD mesh is proposed in this study. The main advantages of the proposed model are three folds. First, the model learns geometric prior during training and eliminates most handcrafts in object shape parameterization. Second, the model represents both the surface mesh and the corresponding computational mesh of the object. Third, the model is fully differentiable and works well with gradient-based downstream applications.

We develop an auto-decoder model to encode a given geometry that only contains unstructured points sampled on the surface into a low-dimensional latent vector. The latent vector is then decoded as the deformation of a fixed template mesh to reconstruct the target geometric surface. We propose a regularization loss applied during training and a differentiable Active Model layer applied during inference to regularize the quality of deformed computational meshes.

Extensive experiments have been conducted to validate the effectiveness of the proposed model. The accurate reconstruction results demonstrate that the latent vectors contain rich geometric information. The ablation studies on the quality of computational mesh show that the decoded CFD meshes can replace re-meshing when conducting numerical simulations. By integrating the proposed model into an end-to-end shape optimization pipeline, one can perform fast prototyping by minimizing manual interventions for parameterization and can produce reasonable optimized shapes. We've also discovered that the trained model is insensitive to different types of template meshes. Other latent space properties, such as smoothness and principal components, are investigated and visualized in the Appendix.

Acknowledgement

This work is supported in part by the Swiss National Science Foundation and the French "Programme d'Investissements D'avenir": ANR-17-EURE-0005. Z. Wei is supported by the TSAE scholarship funded by Toulouse Graduate School in Aerospace Engineering. M. Bauerheim is also supported by the French Direction Générale de l'Armement through the Agence de l'Innovation de Défense (AID) DECAP project.

Appendix

A. The Properties of the Learned Latent Space

1. The Smoothness of Latent Space

We analyze the latent space's smoothness by interpolating the latent codes of two different airfoils $\mathbf{z}_1$ and $\mathbf{z}_2$. The interpolated M^S is extracted from the reconstructed CFD mesh decoded from a weighted sum of z_1 and z_2, namely $M(\hat{\mathbf{v}} + g_\Theta(w_1\mathbf{z}_1 + w_2\mathbf{z}_2))$, where $w_1 \in (0, 1)$ and $w_2 = 1 - w_1$ are interpolation weights. We select some visually distinguishable airfoils and show the interpolated results in Fig.22, which included interpolations within NACA airfoils, non-NACA airfoils and between different airfoil series. Each row in Fig.22 shows a gradual transformation on shape, indicating the change of $w_1\mathbf{z}_1 + w_2\mathbf{z}_2$ in the latent space is also smooth.

2. The Learned Geometric Patterns

Here we visualize the learned geometric patterns by analysing the latent space's principle components (PC). Instead of collecting many latent codes, performing SVD and then computing the PCs, we follow a sampling-free approach [68] to decompose the latent space. We modify it to apply it on the auto-decoder model. The first layer of auto-decoder is a graph convolution where the node features are the coordinate of $\hat{\mathbf{v}}_i$ and latent code $\mathbf{z}$. The feature it extracts f_1 is

$$f_1 = \sigma(\sum_i^N \Theta_1[\hat{\mathbf{v}}_i, \mathbf{z}]) \ , \tag{27}$$

where $\sigma(.)$ is the activation function, Θ_1 is the first layer parameter in g_Θ, N means N neighboring vertices connected to $\hat{\mathbf{v}}_i$ and $[.,.]$ means vector concatenation. This process can be equally written as

$$f_1 = \sigma(\sum_i^N (\Theta_{1;v}\hat{\mathbf{v}}_i + \Theta_{1;z}\mathbf{z})); , \tag{28}$$

where $\Theta_{1,v}$ and $\Theta_{1,z}$ are split from Θ_1. Then the principle components are calculated from the matrix decomposition of $(\Theta_{1,z}^T\Theta_{1,z})$.

In Fig.23, we demonstrate the explored novel shapes by perturbing the latent codes of existing airfoils along the directions of top 3 PCs. All novel shapes have noticeable changes. We can observe common patterns for each PC. For example, when perturbing along the first PC, the shapes extend upward at the leading edges while the shapes bulge downward with the negative perturbations. Fig.24 shows the learned patterns of the top three PCs. These observations indicate that the latent space has discovered and acquired geometric prior knowledge via learning.

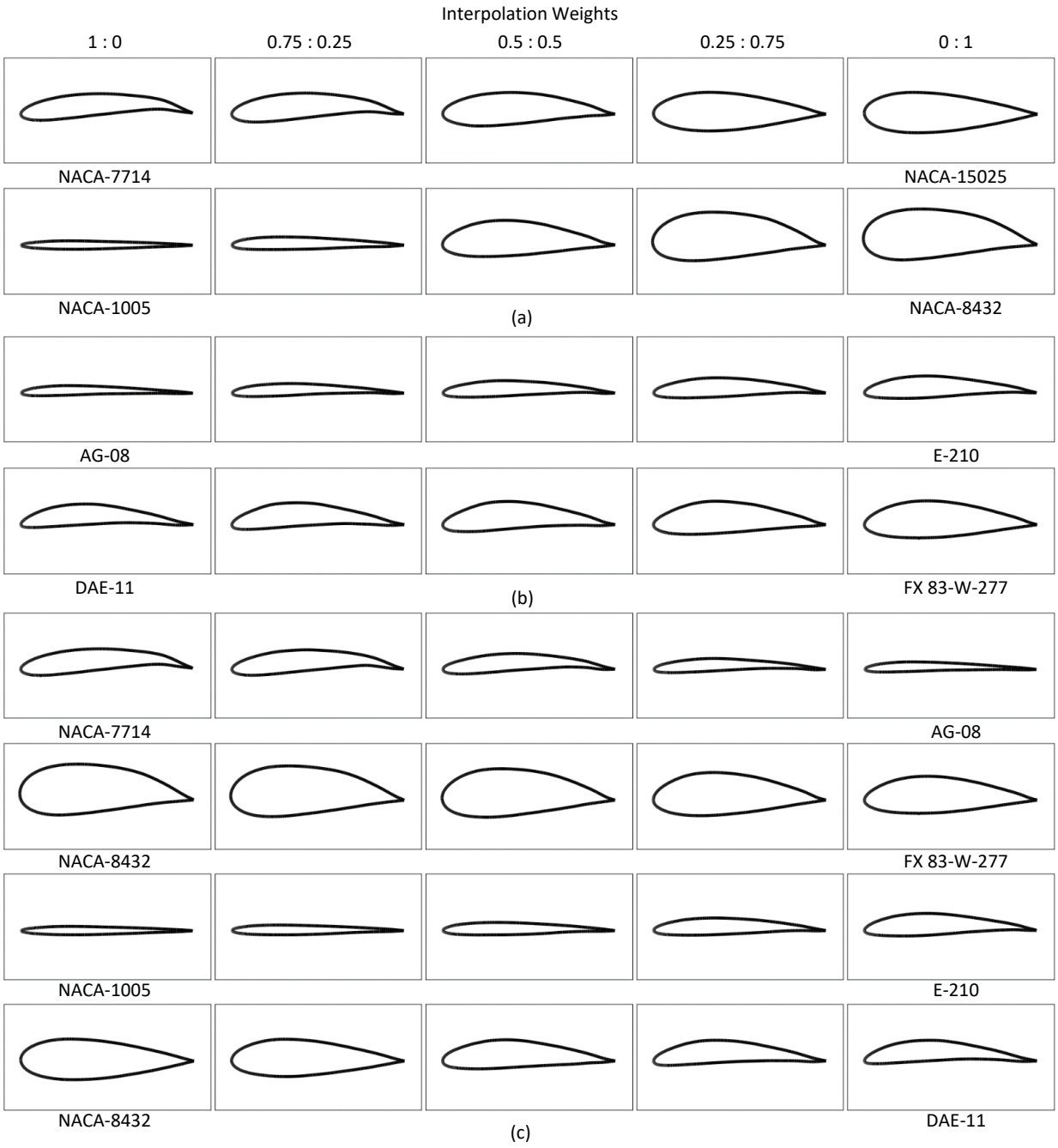

Fig. 22 Decoded airfoils by interpolating between latent codes of (a) NACA, (b) non-NACA and (c) mixed airfoils.

B. Calculating Finite Difference Approximation

Since the following explanation is the same for all vertices and combinations of neighbors, we omit the subscripts i and b for simplicity.

The definition of $\mathbf{r}$, $\mathbf{t}$ and the sampling strategy of $\Delta\mathbf{r}^+$, $\Delta\mathbf{r}^-$, $\delta\mathbf{t}^+$ and $\Delta\mathbf{t}^-$ are described in Sec.III.B and illustrated in

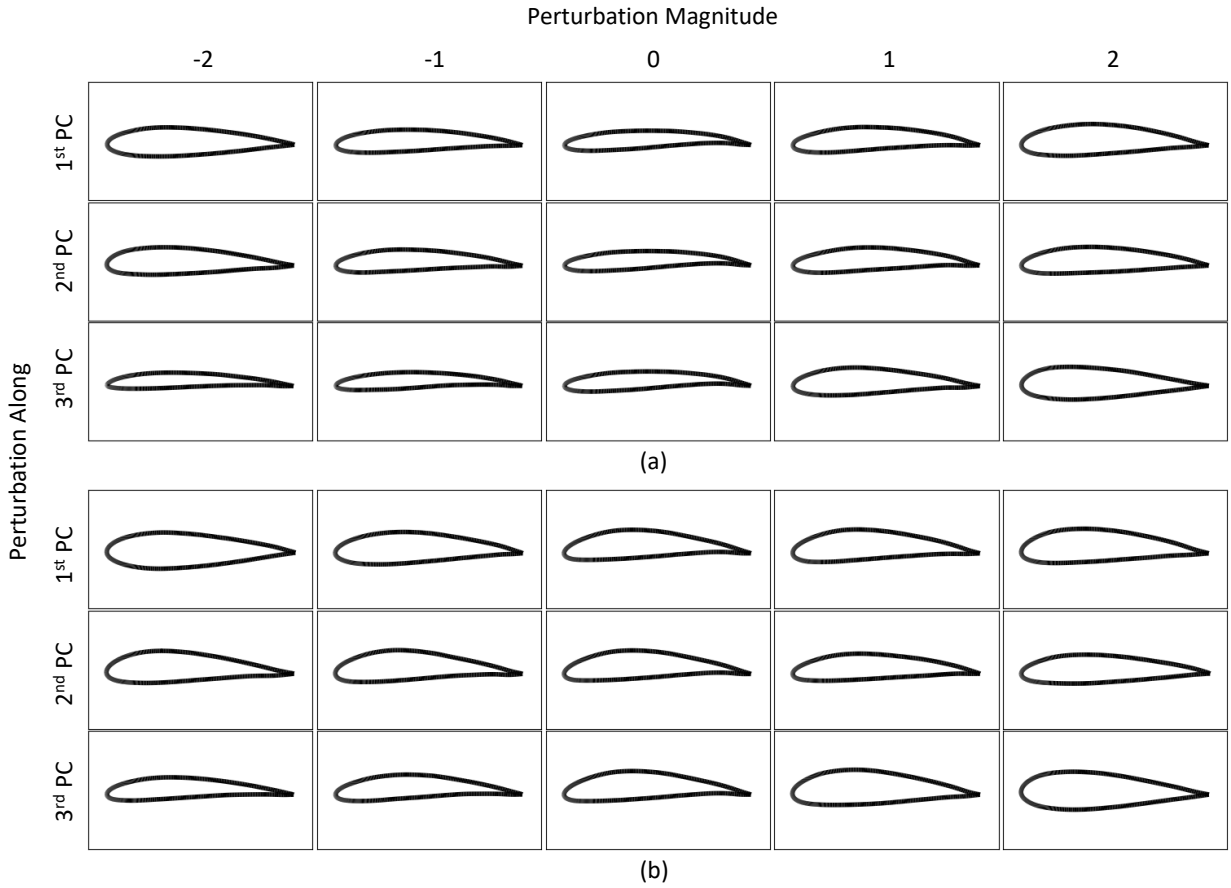

Fig. 23 **Decoded airfoils by perturbing the top 3 PCs from (a) NACA-4710's and (b) LA203A's latent codes.**

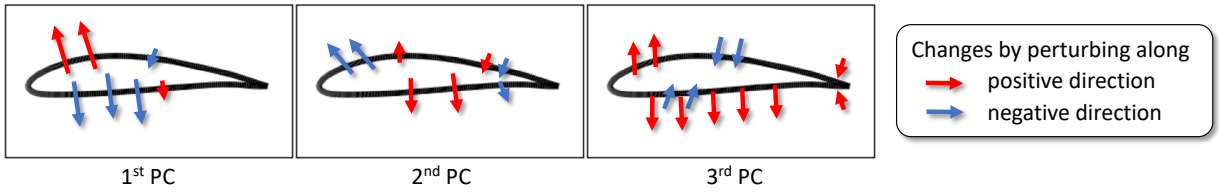

Fig. 24 **Visualized top 3 PCs. Arrows represent changes by perturbing PCs along different directions.**

Fig.4. The subscript i is omitted for simplicity. Let's denote the finite difference sampling, e.g. on axis $\mathbf{r}$, as

$$\mathbf{v}(r + \Delta r^{+}, t) = r\mathbf{r} + \Delta\mathbf{r}^{+} + t\mathbf{t} \ ,$$

$$\mathbf{v}(r - \Delta r^{-}, t) = r\mathbf{r} + \Delta\mathbf{r}^{-} + t\mathbf{t} \ .$$

Then the finite difference approximations for the derivatives in Eq.10 can be written as

$$\begin{aligned}
\frac{\partial \mathbf{v}}{\partial r} &\approx \frac{1}{\Delta r}[\mathbf{v}(r+\Delta r^+, t) - \mathbf{v}(r,t)] \ , \\
\frac{\partial^2 \mathbf{v}}{\partial r^2} &\approx \frac{1}{\Delta r^2}[\mathbf{v}(r+\Delta r^+, t) - 2\mathbf{v}(r,t) + \mathbf{v}(r-\Delta r^-, t)] \ , \\
\frac{\partial^4 \mathbf{v}}{\partial r^4} &\approx \frac{1}{\Delta r^4}[\mathbf{v}(r+2\Delta r^+, t) - 4\mathbf{v}(r+\Delta r^+, t) + 6\mathbf{v}(r,t) - 4\mathbf{v}(r-\Delta r^-, t) + \mathbf{v}(r-2\Delta r^-, t)] \ .
\end{aligned}$$

Similarly, we can write the finite difference approximation w.r.t. t as well.

C. Proof of Validity of Sampling Strategy

Similar as in Appendix.B, we omit the subscript b for simplicity.

In Sec.III.B and Eq.12, we propose how to sample perturbations to approximate the derivatives in Eq.10. This is different from the standard definition to compute a derivative and we give a proof to show why such a sampling method can be an approximation. First, we use a term $l(r_i, t_i)$, a function of vertex $\mathbf{v}_i(r_i, t_i)$, as a concise substitute of $g_\Theta(\mathbf{z}, M)$ when the latent code, edges and vertices except for $\mathbf{v}_i$ in M are fixed. In this case, we can write $\mathbf{dv}_i = l(r_i, t_i)$ and $\mathbf{v}_i = \hat{\mathbf{v}}_i + l(r_i, t_i)$.

In theory, the finite difference of $\mathbf{v}_i$ should be written as

$$\begin{aligned}
\mathbf{v}_i(r_i + \Delta r, t_i) &= (r_i + \Delta r)\mathbf{r}_i + t_i \mathbf{t}_i + l(r_i + \Delta r, t_i) \\
&= r_i \mathbf{r}_i + t_i \mathbf{t}_i + \Delta r \mathbf{r}_i + l(r_i + \Delta r, t_i) \\
&= \hat{\mathbf{v}}_i + \epsilon \mathbf{r}_i + l(r_i + \Delta r, t_i) \ ,
\end{aligned}$$

where the magnitude of perturbation $|\Delta r| = \epsilon$. By expanding $l(r_i + \Delta r, t_i)$ with the first order Taylor expansion at $\hat{\mathbf{v}}_i$, we have

$$\begin{aligned}
\mathbf{v}_i(r_i + \Delta r, t_i) &= \hat{\mathbf{v}}_i + \epsilon \mathbf{r}_i + l(r_i + \Delta r, t_i) \\
&= \hat{\mathbf{v}}_i + \epsilon \mathbf{r}_i + l(r_i, t_i) + \frac{\partial l(r_i, t_i)}{\partial r}(\hat{\mathbf{v}}_i + \Delta r \mathbf{r}_i - \hat{\mathbf{v}}_i) + O(\frac{\partial^2 l}{\partial r^2}) \\
&= \hat{\mathbf{v}}_i + \epsilon \mathbf{r}_i + l(r_i, t_i) + \frac{\partial l(r_i, t_i)}{\partial r}(\epsilon \mathbf{r}_i) + O(\frac{\partial^2 l}{\partial r^2}) \\
&\approx \hat{\mathbf{v}}_i + \epsilon \mathbf{r}_i + l(r_i, t_i) + \frac{\partial l(r_i, t_i)}{\partial r}(\epsilon \mathbf{r}_i) \ .
\end{aligned}$$

In practice, we use a different finite difference instead

$$\begin{aligned}
\mathbf{v}_i(r_i,t_i) + \Delta\mathbf{r}_i{}^+ &= \hat{\mathbf{v}}_i + l(r_i,t_i) + \epsilon\frac{(\mathbf{v}_{ir}-\mathbf{v}_i)}{||\hat{\mathbf{v}}_{ir}-\hat{\mathbf{v}}_i||^2} \\
&= \hat{\mathbf{v}}_i + l(r_i,t_i) + \epsilon(\frac{\hat{\mathbf{v}}_{ir}-\hat{\mathbf{v}}_i}{||\hat{\mathbf{v}}_{ir}-\hat{\mathbf{v}}_i||^2} + \frac{l(r_{ir},t_{ir})-l(r_i,t_i)}{||\hat{\mathbf{v}}_{ir}-\hat{\mathbf{v}}_i||^2}) \\
&= \hat{\mathbf{v}}_i + l(r_i,t_i) + \epsilon\mathbf{r}_i + \epsilon\frac{l(r_{ir},t_{ir})-l(r_i,t_i)}{||\hat{\mathbf{v}}_{ir}-\hat{\mathbf{v}}_i||^2} \ .
\end{aligned}$$

Again, we use the first order Taylor expansion of $l(r_{ir},t_{ir})$ at $\hat{\mathbf{v}}_i$. Considering ϵ is a very small value, we have

$$\begin{aligned}
\mathbf{v}_i(r_i,t_i) + \Delta\mathbf{r}_i{}^+ &= \hat{\mathbf{v}}_i + l(r_i,t_i) + \epsilon\mathbf{r}_i + \epsilon\frac{l(r_{ir},t_{ir})-l(r_i,t_i)}{||\hat{\mathbf{v}}_{ir}-\hat{\mathbf{v}}_i||^2} \\
&= \hat{\mathbf{v}}_i + l(r_i,t_i) + \epsilon\mathbf{r}_i + \frac{\partial l(r_i,t_i)}{\partial r}(\epsilon\mathbf{r}_i) + O(\epsilon^2\frac{\partial^2 l}{\partial r^2}) \\
&\approx \hat{\mathbf{v}}_i + l(r_i,t_i) + \epsilon\mathbf{r}_i + \frac{\partial l(r_i,t_i)}{\partial r}(\epsilon\mathbf{r}_i) \\
&\approx \mathbf{v}_i(r_i+\Delta r,t_i) \ . \quad \square
\end{aligned}$$

Similarly, we can write down the negative perturbation in theory as

$$\mathbf{v}_i(r_i-\Delta r,t_i) \approx \hat{\mathbf{v}}_i - \epsilon\mathbf{r}_i + l(r_i,t_i) - \frac{\partial l(r_i,t_i)}{\partial r}(\epsilon\mathbf{r}_i) \ .$$

Let's define the affine transformation matrix $R(\alpha)$ that rotates $\mathbf{\Delta t}^+$ into $\mathbf{\Delta r}^-$ as $\mathbf{\Delta r}^- = R(\alpha)\mathbf{\Delta t}^+$. Then we can write down the finite difference used in practice as

$$\begin{aligned}
\mathbf{v}_i(r_i,t_i) + \Delta\mathbf{r}_i{}^- &= \mathbf{v}_i(r_i,t_i) + R(\alpha)\Delta\mathbf{t}_i{}^+ \\
&= \hat{\mathbf{v}}_i + l(r_i,t_i) + \epsilon R(\alpha)\frac{\mathbf{v}_{it}-\mathbf{v}_i}{||\hat{\mathbf{v}}_{it}-\hat{\mathbf{v}}_i||^2} \\
&= \hat{\mathbf{v}}_i + l(r_i,t_i) + \epsilon R(\alpha)\mathbf{t}_i + \epsilon R(\alpha)\frac{l(r_{it},t_{it})-l(r_i,t_i)}{||\hat{\mathbf{v}}_{it}-\hat{\mathbf{v}}_i||^2} \ .
\end{aligned}$$

By use the first order Taylor expansion of $l(r_{it},t_{it})$ at $\hat{\mathbf{v}}_i$, we have

$$\begin{aligned}
\mathbf{v}_i(r_i,t_i) + \Delta\mathbf{r}_i{}^- &= \hat{\mathbf{v}}_i + l(r_i,t_i) + \epsilon R(\alpha)\mathbf{t}_i + \epsilon R(\alpha)\frac{l(r_{it},t_{it})-l(r_i,t_i)}{||\hat{\mathbf{v}}_{it}-\hat{\mathbf{v}}_i||^2} \\
&\approx \hat{\mathbf{v}}_i + l(r_i,t_i) + \epsilon R(\alpha)\mathbf{t}_i + \epsilon R(\alpha)\frac{\partial l(r_i,t_i)}{\partial t}\mathbf{t}_i \\
&= \hat{\mathbf{v}}_i - \epsilon\mathbf{r}_i + l(r_i,t_i) - \frac{\partial l(r_i,t_i)}{\partial r}(\epsilon\mathbf{r}_i) \\
&\approx \mathbf{v}_i(r_i-\Delta r,t_i) \ . \quad \square
\end{aligned}$$

In conclusion, the vicinity of $\mathbf{v}_i$ in the tangent space of M at vertex $\mathbf{v}_i$ is an approximation of the real manifold with first order accuracy. $\mathbf{v}_i + \Delta\mathbf{r}_i^+$ and $\mathbf{v}_i + \Delta\mathbf{r}_i^-$ can substitute $\mathbf{v}_i(r_i + \Delta r, t_i)$ and $\mathbf{v}_i(r_i - \Delta r, t_i)$, respectively. Similar conclusions also apply for perturbations towards other directions.

D. Details of Surrogate Based Optimization and Surrogate Model

Our shape optimization followed the major procedures of the surrogate-based optimization (SBO) [69]. The main steps are listed below.

Algorithm 1 The workflow of our shape optimization.

Given: Airfoils from the UIUC database and their RANS simulation results.
1. Construct a GCNN surrogate model based on the RANS pressure data.
2. Minimize the optimization objectives as defined in Eq.22.
3. Sample optimized shapes generated from different initial airfoils, perform RANS simulations and update the surrogate model using new data.
4. Minimize the optimization objectives again with the updated GCNN.

More specifically, $1,000$ airfoils are collected at Step 1 and 300 shapes are sampled at Step 3.

As for the surrogate model, the same network architecture of the surrogate model proposed in [67] is used. The model is composed of 5 graph convolution blocks. Each block contains 3 graph convolutional layers. A batch normalization layer is used after each graph convolutional layers and the Exponential Linear Unit is used as the activation function. The surrogate model takes M^S as input. It predicts the pressure value for each vertex and the drag coefficient can be computed via an integral on the airfoil's surface. To train and update the surrogate model, the Adam optimizer is adopted with the learning rate as 5×10^{-4}. The training uses 900 epochs.

References

[1] Robinson, G. M., and Keane, A. J., "Concise Orthogonal Representation of Supercritical Airfoils," Journal of Aircraft, Vol. 38, No. 3, 2001, pp. 580–583.

[2] Poole, D. J., Allen, C. B., and Rendall, T. C. S., "Metric-Based Mathematical Derivation of Efficient Airfoil Design Variables," American Institute of Aeronautics and Astronautics Journal, Vol. 53, No. 5, 2015, pp. 1349–1361.

[3] Masters, D. A., Taylor, N. J., Rendall, T. C. S., Allen, C. B., and Poole, D. J., "Geometric Comparison of Aerofoil Shape Parameterization Methods," American Institute of Aeronautics and Astronautics Journal, Vol. 55, No. 5, 2017, pp. 1575–1589.

[4] Li, J., Bouhlel, M. A., and Martins, J. R. R. A., "Data-Based Approach for Fast Airfoil Analysis and Optimization," American Institute of Aeronautics and Astronautics Journal, Vol. 57, No. 2, 2019, pp. 581–596.

[5] Kedward, L., Allen, C. B., and Rendall, T., "Towards Generic Modal Design Variables for Aerodynamic Shape Optimisation," AIAA Scitech Forum, 2020.

[6] Viswanath, A., Forrester, A. I. J., and Keane, A. J., "Dimension Reduction for Aerodynamic Design Optimization," American Institute of Aeronautics and Astronautics Journal, Vol. 49, No. 6, 2011, pp. 1256–1266.

[7] Constantine, P. G., Dow, E., and Wang, Q., "Active Subspace Methods in Theory and Practice: Applications to Kriging Surfaces," SIAM Journal on Scientific Computing, Vol. 36, No. 4, 2014, pp. A1500–A1524.

[8] Li, J., Zhang, M., Martins, J. R. R. A., and Shu, C., "Efficient Aerodynamic Shape Optimization with Deep-Learning-Based Geometric Filtering," American Institute of Aeronautics and Astronautics Journal, Vol. 58, No. 10, 2020, pp. 4243–4259.

[9] Li, J., and Zhang, M., "On Deep-Learning-Based Geometric Filtering in Aerodynamic Shape Optimization," Aerospace Science and Technology, Vol. 112, 2021, p. 106603.

[10] Mi, B., Cheng, S., Luo, Y., and Fan, H., "A new many-objective aerodynamic optimization method for symmetrical elliptic airfoils by PSO and direct-manipulation-based parametric mesh deformation," Aerospace Science and Technology, Vol. 120, 2022. https://doi.org/https://doi.org/10.1016/j.ast.2021.107296.

[11] Stannard, A., and Qin, N., "Hybrid Mesh Deformation for Aerodynamic-Structural Coupled Adjoint Optimization," AIAA Journal, Vol. 60, No. 6, 2022, pp. 3438–3451. https://doi.org/10.2514/1.J061293.

[12] Li, J., He, S., Zhang, M., Martins, J. R. R. A., and Cheong Khoo, B., "Physics-Based Data-Driven Buffet-Onset Constraint for Aerodynamic Shape Optimization," American Institute of Aeronautics and Astronautics Journal, Vol. 60, No. 8, 2022, pp. 4775–4788. https://doi.org/10.2514/1.J061519.

[13] Kaya, H., and Tuncer, I. H., "Discrete Adjoint-Based Aerodynamic Shape Optimization Framework for Natural Laminar Flows," American Institute of Aeronautics and Astronautics Journal, Vol. 60, No. 1, 2022, pp. 197–212. https://doi.org/10.2514/1.J059923.

[14] Wu, N., Mader, C. A., and Martins, J. R., "A Gradient-based Sequential Multifidelity Approach to Multidisciplinary Design Optimization," Structural and Multidisciplinary Optimization, Vol. 65, No. 4, 2022, pp. 1–20. https://doi.org/10.1007/s00158-022-03204-1.

[15] Li, W., and Geiselhart, K., "Multi-objective, Multidisciplinary Optimization of Low-Boom Supersonic Transports Using Multifidelity Models," Journal of Aircraft, 2022. https://doi.org/10.2514/1.C036656.

[16] Tian, F.-B., Dai, H., Doyle, J. F., and Rousseau, B., "Fluid-Structure Interaction Involving Large Deformations: 3D Simulations and Applications to Biological Systems," Journal of Computational Physics, Vol. 258, 2014, pp. 451–469.

[17] Lyu, Z., Kenway, G., and Martins, J., "Aerodynamic Shape Optimization Investigations of the Common Research Model Wing Benchmark," American Institute of Aeronautics and Astronautics Journal, Vol. 53, No. 4, 2015, pp. 968–985.

[18] Roy, P. T., Segui, L. M., Jouhaud, J.-C., and Gicquel, L., "Resampling Strategies to Improve Surrogate ModelBased Uncertainty Quantification: Application to LES of LS89," International Journal for Numerical Methods in Fluids, Vol. 87, No. 12, 2018, pp. 607–627. https://doi.org/https://doi.org/10.1002/fld.4504.

[19] Toal, D. J. J., Bressloff, N. W., Keane, A. J., and Holden, C. M. E., "Geometric Filtration Using Proper Orthogonal Decomposition for Aerodynamic Design Optimization," American Institute of Aeronautics and Astronautics Journal, Vol. 48, No. 5, 2010, pp. 916–928.

[20] Sederberg, T., and Parry, S., "Free-Form Deformation of Solid Geometric Models," ACM SIGGRAPH, Vol. 20, No. 4, 1986.

[21] Lamousin, H. J., and Jr., W. N. W., "NURBS-based free-form deformations," Computer Graphics and Applications, Vol. 14, No. 6, 1994, pp. 59–65. https://doi.org/10.1109/38.329096.

[22] Kenway, G., Kennedy, G., and Martins, J. R. R. A., "A CAD-Free Approach to High-Fidelity Aerostructural Optimization," 13th AIAA/ISSMO Multidisciplinary Analysis Optimization Conference, Fort Worth, Texas, USA, September 13-15, 2010. https://doi.org/10.2514/6.2010-9231.

[23] de Boer, A., van der Schoot, M., and Bijl, H., "Mesh Deformation Based on Radial Basis Function Interpolation," Computers and Structures, Vol. 85, No. 11, 2007, pp. 784–795.

[24] Li, Z., Wang, X., Wang, F., and Jiang, P., "On Boosting Single-Frame 3D Human Pose Estimation via Monocular Videos," International Conference on Computer Vision, 2019.

[25] Lukaczyk, T. W., Constantine, P., Palacios, F., and Alonso, J. J., "Active Subspaces for Shape Optimization," Multidisciplinary Design Optimization Conference, 2014. https://doi.org/10.2514/6.2014-1171.

[26] Namura, N., Shimoyama, K., and Obayashi, S., "Kriging surrogate model with coordinate transformation based on likelihood and gradient," Journal of Global Optimization, Vol. 68, No. 3, 2017, pp. 827–849. https://doi.org/10.1007/s10898-017-0516-y.

[27] Grey, Z. J., and Constantine, P. G., "Active Subspaces of Airfoil Shape Parameterizations," American Institute of Aeronautics and Astronautics Journal, Vol. 56, No. 5, 2018, pp. 2003–2017. https://doi.org/10.2514/1.J056054.

[28] Bauerheim, M., Ndiaye, A., Constantine, P., Moreau, S., and Nicoud, F., "Symmetry breaking of azimuthal thermoacoustic modes: the UQ perspective," Journal of Fluid Mechanics, Vol. 789, 2016, p. 534–566. https://doi.org/10.1017/jfm.2015.730.

[29] Magri, L., and Fusiello, A., "Multiple Model Fitting as a Set Coverage Problem," Conference on Computer Vision and Pattern Recognition, 2016.

[30] Bellman, R., Curse of Dimensionality, Princeton university press, 1961.

[31] Payne, L. E., and Weinberger, H. F., "An optimal Poincaré inequality for convex domains," Archive for Rational Mechanics and Analysis, Vol. 5, 1960, pp. 286–292. https://doi.org/10.1007/BF00252910.

[32] Beyer, K., Goldstein, J., Ramakrishnan, R., and Shaft, U., "When is "nearest neighbor" meaningful?" International Conference on Database Theory, 1999, pp. 217–235. https://doi.org/10.1007/3-540-49257-7_15.

[33] Kulfan, B. M., "Universal Parametric Geometry Representation Method," Journal of Aircraft, Vol. 45, No. 1, 2008, pp. 142–158. https://doi.org/10.2514/1.29958.

[34] Ceze, M., Hayashi, M., and Volpe, E., A Study of the CST Parameterization Characteristics, 2009. https://doi.org/10.2514/6.2009-3767.

[35] Achour, G., Sung, W. J., Pinon-Fischer, O. J., and Mavris, D. N., "Development of a Conditional Generative Adversarial Network for Airfoil Shape Optimization," AIAA Scitech Forum, Orlando, FL, USA, January 6-10, 2020. https://doi.org/10.2514/6.2020-2261.

[36] Chen, W., Chiu, K., and Fuge, M. D., "Airfoil Design Parameterization and Optimization Using Bézier Generative Adversarial Networks," American Institute of Aeronautics and Astronautics Journal, Vol. 58, No. 11, 2020, pp. 4723–4735. https://doi.org/10.2514/1.J059317.

[37] Du, X., He, P., and Martins, J. R. R. A., "A B-Spline-based Generative Adversarial Network Model for Fast Interactive Airfoil Aerodynamic Optimization," AIAA Scitech Forum, Orlando, FL, USA, January 6-10, 2020. https://doi.org/10.2514/6.2020-2128.

[38] Dai, A., Qi, C., and Nießner, M., "Shape Completion Using 3D-Encoder-Predictor CNNs and Shape Synthesis," Conference on Computer Vision and Pattern Recognition, 2017.

[39] Wu, J., Zhang, C., Zhang, X., Zhang, Z., Freeman, W. T., and Tenenbaum, J. B., "Learning Shape Priors for Single-View 3D Completion And Reconstruction," European Conference on Computer Vision, Vol. 11215, 2018, pp. 673–691.

[40] Bagautdinov, T., Wu, C., Saragih, J., Fua, P., and Sheikh, Y., "Modeling Facial Geometry Using Compositional VAEs," Conference on Computer Vision and Pattern Recognition, 2018.

[41] Tan, S., and Mayrovouniotis, M. L., "Reducing data dimensionality through optimizing neural network inputs," AIChE Journal, Vol. 41, No. 6, 1995, pp. 1471–1480.

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

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

[44] Knupp, P., "Remarks on Mesh Quality," Tech. rep., Sandia National Lab.(SNL-NM), Albuquerque, NM (United States), 2007.

[45] Kass, M., Witkin, A., and Terzopoulos, D., "Snakes: Active Contour Models," International Journal of Computer Vision, Vol. 1, No. 4, 1988, pp. 321–331.

[46] Terzopoulos, D., and Vasilescu, M., "Sampling and Reconstruction with Adaptive Meshes," Conference on Computer Vision and Pattern Recognition, 1991, pp. 70–75.

[47] Fua, P., "Model-Based Optimization: Accurate and Consistent Site Modeling," International Society for Photogrammetry and Remote Sensing, 1996.

[48] Wickramasinghe, U., Knott, G., and Fua, P., "Deep Active Surface Models," Conference on Computer Vision and Pattern Recognition, 2021.

[49] Kipf, T., and Welling, M., "Semi-Supervised Classification with Graph Convolutional Networks," arXiv Preprint, 2016.

[50] Salimans, T., and Kingma, D., "Weight Normalization - A Simple Reparameterization to Accelerate Training of Deep Neural Networks," Advances in Neural Information Processing Systems, 2016.

[51] Nair, V., and Hinton, G. E., "Rectified Linear Units Improve Restricted Boltzmann Machines," International Conference on Machine Learning, 2010.

[52] Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S., "Pytorch: An Imperative Style, High-Performance Deep Learning Library," Advances in Neural Information Processing Systems, 2019.

[53] Kingma, D. P., and Ba, J., "Adam: A Method for Stochastic Optimisation," International Conference on Learning Representations, 2015.

[54] Selig, M., UIUC airfoil data site, Department of Aeronautical and Astronautical Engineering, University of Illinois at Urbana-Champaign, 1996.

[55] Barrow, H. G., Tenenbaum, J. M., Bolles, R. C., and Wolf, H. C., "Parametric Correspondence and Chamfer Matching: Two New Techniques for Image Matching," International Joint Conference on Artificial Intelligence, 1977.

[56] Chapin, V., de Carlan, N., and Heppel, P., "Performance Optimization of Interacting Sails through Fluid Structure Coupling," International Journal of Small Craft Technology, Vol. 153 (Part B2), 2011, pp. 103–116. https://doi.org/10.3940/rina.innovsail.2010.08.

[57] Viola, I. M., Chapin, V., Speranza, N., and Biancolini, M. E., "Optimal airfoil's shapes by high fidelity (CFD)," Aircraft Engineering and Aerospace Technology, Vol. 90, No. 6, 2018, pp. 1000–1011. https://doi.org/10.1108/aeat-09-2017-0210.

[58] Geuzaine, C., and Remacle, J.-F., "Gmsh: A 3-D finite element mesh generator with built-in pre-and post-processing facilities," International Journal for Mumerical Methods in Engineering, 2009. https://doi.org/10.1002/nme.2579.

[59] Selig, M., Guglielmo, J., Broeren, A., and Giguère, P., Summary of low speed airfoil data Vol. 1, SoarTech Publications, 1995.

[60] Selig, M., Lyon, C., Giguère, P., Ninham, C., and Guglielmo, J., Summary of low speed airfoil data Vol. 2, SoarTech Publications, 1996.

[61] Lyon, C., Broeren, A., Giguère, P., Gopalarathnam, A., and Selig, M., Summary of low speed airfoil data Vol. 3, SoarTech Publications, 1998.

[62] Williamson, G., McGranahan, B., Broughton, B., Deters, R., Brandt, J., and Selig, M., Summary of low speed airfoil data Vol. 5, Department of Aeronautical and Astronautical Engineering, University of Illinois at Urbana-Champaign, 2012.

[63] Menter, F., "Zonal Two Equation k-w Turbulence Models For Aerodynamic Flows," 23rd Fluid Dynamics, Plasmadynamics, and Lasers Conference, 1993. https://doi.org/10.2514/6.1993-2906.

[64] Spalart, P., and Allmaras, S., "A One-Equation Turbulence Model for Aerodynamic Flows," Aerospace Sciences Meeting and Exhibit, 1992. https://doi.org/10.2514/6.1992-439.

[65] Mader, C. A., Kenway, G. K. W., Yildirim, A., and Martins, J. R. R. A., "ADflow—An open-source computational fluid dynamics solver for aerodynamic and multidisciplinary optimization," Journal of Aerospace Information Systems, 2020. https://doi.org/10.2514/1.I010796.

[66] Economon, T. D., Palacios, F., Copeland, S. R., Lukaczyk, T. W., and Alonso, J. J., "SU2: An Open-Source Suite for Multiphysics Simulation and Design," American Institute of Aeronautics and Astronautics Journal, 2016. https://doi.org/10.2514/1.J053813.

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

[68] Shen, Y., and Zhou, B., "Closed-Form Factorization of Latent Semantics in GANs," Conference on Computer Vision and Pattern Recognition, 2021. https://doi.org/10.1109/cvpr46437.2021.00158.

[69] Queipo, N. V., Haftka, R. T., Shyy, W., Goel, T., Vaidyanathan, R., and Kevin Tucker, P., "Surrogate-based analysis and optimization," Progress in Aerospace Sciences, Vol. 41, No. 1, 2005. https://doi.org/https://doi.org/10.1016/j.paerosci.2005.02.001.