# Point2FFD: Learning Shape Representations of Simulation-ready 3D Models for Engineering Design Optimization

**Thiago Rios, Bas van Stein, Thomas Bäck, Bernhard Sendhoff, Stefan Menzel**

**2021**

**Preprint:**

This is an accepted article published in International Conference on 3D Vision. The final authenticated version is available online at: https://doi.org/10.1109/3DV53792.2021.00110 

# Point2FFD: Learning Shape Representations of Simulation-Ready 3D Models for Engineering Design Optimization

Thiago Rios[1], Bas van Stein[2], Thomas Bäck[2], Bernhard Sendhoff[1], Stefan Menzel[1]
[1]Honda Research Institute Europe GmbH, 63073 Offenbach, Germany
[2]Leiden Institute of Advanced Computer Science (LIACS), 2333 CA Leiden, The Netherlands
{thiago.rios, bernhard.sendhoff, stefan.menzel}honda-ri.de
{b.van.stein, t.h.w.baeck}@liacs.leidenuniv.nl

## Abstract

*Methods for learning on 3D point clouds became ubiquitous due to the popularization of 3D scanning technology and advances of machine learning techniques. Among these methods, point-based deep neural networks have been utilized to explore 3D designs in optimization tasks. However, engineering computer simulations require high-quality meshed models, which are challenging to automatically generate from unordered point clouds. In this work, we propose Point2FFD: A novel deep neural network for learning compact geometric representations and generating simulation-ready meshed models. Built upon an autoencoder architecture, Point2FFD learns to compress 3D point clouds into a latent design space, from which the network generates 3D polygonal meshes by selecting and deforming simulation-ready mesh templates. Through benchmark experiments, we show that our proposed network achieves comparable shape-generative performance than existing state-of-the-art point-based generative models. In real world-inspired vehicle aerodynamic optimizations, we demonstrate that Point2FFD generates simulation-ready meshes of realistic car shapes and leads to better optimized designs than the benchmarked networks.*

## 1. Introduction

Geometric data are ubiquitous in engineering design processes. During product development, engineers define different representations of 3D shapes to explore solutions, analyze performance and verify the compliance to manufacturing standards. However, handcrafted design features often bias the design exploration and constrain the solutions [42]. Furthermore, since these representations are often product-specific, this approach hinders the exploitation of engineering expertise embedded in similar previous products by transferring design properties.

Recently proposed geometric deep learning architectures address some of these challenges by enabling automated feature learning on unstructured data [5]. Autoencoder networks are particularly suitable for learning design features for engineering optimization, since autoencoders both compress geometric data into low-dimensional representations and learn a shape-generative model [43, 2, 37, 14, 38]. Furthermore, networks for learning on 3D point cloud data became widespread in the literature due to the data availability and simplicity of the representation [3, 16].

However, simulation-based design optimization algorithms often require designs represented as high-quality polygonal meshes. Meshing unordered 3D point clouds is an ill-posed problem, difficult to automate, and usually requires manual tuning and verification [3]. Recently, researchers approached the mesh reconstruction task with deep neural networks (DNNs). In these approaches, the DNNs either mesh point clouds as a post-processing task [40, 18, 36], or combine additional information, *e.g.* structural or shape priors, to learn mesh representations from 3D point clouds [27, 45, 14]. Nonetheless, the literature still lacks a method that addresses both simultaneously, representation and reconstruction of 3D representations tailored for engineering simulations.

In this paper, we propose *Point2FFD*[1]: An artificial neural network for automated generation of simulation-ready 3D meshes from learned latent representations. Our proposed architecture builds upon a 3D point cloud autoencoder to compress point cloud data into low-dimensional latent vectors. From the latent space, a two-branch network generates simulation-ready meshes by selecting and deforming existing mesh templates parameterized with free-form deformation (FFD). A 3D polygonal mesh is said simulation-ready if the mesh is artefact-free and suitable for performing engineering computer simulations, *e.g.*, com-

[1]Source code available in our repository GDL4DeisgnApps (https://github.com/HRI-EU/GDL4DesignApps.)

puter fluid dynamics (CFD). Hence, Point2FFD generates meshed 3D models with diverse topology, which avoids intense post-processing, as required for 3D point clouds, and allows engineers to directly manipulate simulation meshes, reducing the computational effort during optimizations.

We outline this paper as follows: In Section 2, we survey the literature related to our work and discuss the state-of-the-art on data-driven design representations. In Section 3, we present the Point2FFD architecture, as well as the utilized techniques to sample 3D point clouds, network hyperparameters and training settings. In Section 4, we provide details on the set-up and results of two experiments. First, we benchmark the performance of Point2FFD against similar 3D point cloud autoencoders used in engineering optimization. Second, we apply the representations learned by Point2FFD in real-world inspired vehicle aerodynamic optimization experiments. Finally, in Section 5, we conclude this paper with a summary and outlook of our work.

## 2. Related Work

**Deformation-based design representations.** In optimization, engineers prefer compact design representations to balance computational efficiency and design flexibility [48, 42]. Specifically for simulation-based shape optimization problems, the simulation algorithms often require 3D shapes represented as refined polygonal meshes, which are computationally expensive to generate. Hence, for this class of problems, deformation-based representations are often more efficient, since the methods operate directly on the vertices of the meshes used for simulations and reduce the number of re-meshing steps during an optimization [17].

Free-form deformation (FFD) is a prominent shape morphing technique [39]. FFD represents the deformation of 3D polygonal meshes by mapping the vertices to a low-dimensional lattice of control points utilizing tri-variate Bernstein polynomials. The control points operate as design features and intuitively deform the mesh when displaced in the 3D Cartesian space (Fig. 1). Depending on the amount of deformation and configuration of the lattice, FFD preserves the quality of the embedded mesh within reasonable tolerance and, thus, the quality of the simulation results [30, 11, 17].

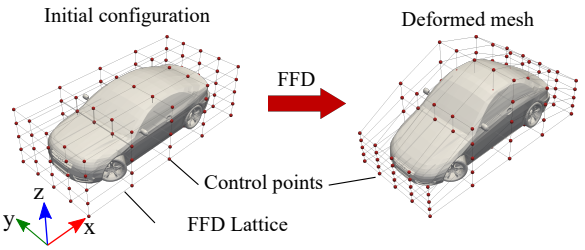


Figure 1. Deformation of a car shape using standard FFD techniques proposed in [39].

Many works explore FFD and variations of the technique in design optimization problems, particularly in the context of aerodynamic applications using CFD [29, 30, 31, 41, 22, 47]. However, FFD still requires human expertise to set up the lattice and select the degrees of freedom, which is challenging for unexperienced designers. Furthermore, FFD only generates isomorphic meshes, which constrains the design exploration. Point2FFD addresses these challenges by learning the design representations and by generating shapes based on templates with different topology.

**3D point cloud autoencoders.** The interest on learning point cloud data has been increasing due to the flexibility and low computational effort required to process 3D point clouds [16]. For 3D design tasks, point clouds also enable engineers to learn generative models on shapes with different topology, which is a current restriction for learning on polygonal meshes [6]. Furthermore, computer aided engineering (CAE) models are often aligned, scaled and the point cloud data derived from these models lack artifacts observed in scanned 3D shapes.

Qi *et al*. [8, 32] pioneered the research on deep neural networks for learning point cloud data. The authors in [2, 46] extend these networks to deep-generative models of 3D point clouds. In [34], the authors use a variation of the architecture given in [2] to learn latent representations of vehicle designs for a target-shape matching optimization, and in [35] they verify the scalability of the architecture to higher-dimensional CAE models. In [35], the authors exploit the transfer of the latent variables learned with the same autoencoder to foster commonality and knowledge transfer between shapes in a multi-task vehicle aerodynamic optimization problem. The authors in [38] extend the network to a variational autoencoder and assess the effects of the regularization of the latent space on the shape-generative capabilities of the autoencoder.

Yet, since these networks process input data with point-wise operators, the latent features miss global shape information and encode a notion of occupancy of the input space [36]. The works presented in [13, 26, 20] address the feature aggregation issues with more complex architectures, but for shape segmentation and classification tasks. Umetani [43] proposed to re-parameterize CAE models into organized point clouds, which the author used to learn 3D shapes. Although effective, the proposed autoencoder comprises only fully-connected layers, which scale poorly to higher-dimensional point clouds.

**3D mesh-generative models.** In practice, automated post-processing of 3D point clouds into CAE meshes is the main challenge for applying deep-generative models in engineering optimization. Most of the well-known techniques, *e.g*. Poisson [23], ball-pivoting algorithm [4] and graph-cut minimization [21], require manual tuning and verification, which is undesired in automated optimization pipelines.

Recent works propose novel DNNs for re-meshing unorganized point clouds. PointTriNet [40] triangulates points in $\mathrm{R}^3$ by combining two networks to search and classify samples in a point cloud. Point2Mesh [18] matches the vertices of a template mesh to an input point cloud by leaning shape priors and, thus, avoids any pre-training of the network. Rios *et al*. [33] proposed a method to search prototypical meshes for recovering the surfaces on point clouds using target-shape matching optimization with FFD.

The autoencoder-based networks proposed in [15, 45] learn both, shape representation and surface reconstruction. Given a pair with target and template shapes, the network learns to deform the vertices of the template mesh to match the target shape. Li *et al*. [27] proposed the GRASS autoencoders for learning latent representations and shape reconstructions based on the hierarchical structures of 3D designs. Finally, based on GRASS, Gao *et al*. [14] proposed SDM-NET: A variational autoencoder that learns to deform and combine shape primitives to generate meshed geometries with different topologies.

Instead of envisioning shape generation as an isolated task, we tailored Point2FFD to address the efficiency in engineering design optimization with learning-based representations. Our objective is to leverage engineering expertise embedded in templates of simulation-ready meshes parameterized with FFD to both, improve the quality of the simulation results and speed up the simulation setup during the optimization. Furthermore, the network still learns a compact design parameterization, which improves the performance of optimization algorithms by simplifying the search space.

## 3. Methods

In the following, we first describe the Point2FFD architecture (Section 3.1) and data set pre-processing utilized to train the network (Section 3.2). Then, we describe the baseline hyperparameters of the network, the algorithm and respective settings for training Point2FFD (Section 3.3).

### 3.1. Point2FFD Architecture

The Point2FFD architecture comprises two parts: An encoder and a shape-generative model (Fig. 2). By compressing the input data through the encoder, the network learns a compact design representation, the so-called latent space ($Z$). The shape-generative model is a two-branch network that selects a simulation-ready mesh template from a pool of available FFD-parameterized CAE models based on the latent representation, and predicts the lattice deformation to match the mesh template to the input shape.

The encoder follows the network proposed in [35] with five 1D convolutional layers. The first four layers are activated with rectified linear unit (ReLU) and the last layer with a hyperbolic tangent function. After the convolutions, a max-pooling operator reduces the dimensionality of the input data to a $L_z$-dimensional vector $Z^0 \in [-1, 1]^{L_z}$. The latent representation $Z$ is obtained by adding Gaussian noise to $Z^0$ (Eq. 1). During the training, the noise increases the robustness of the shape-generative model without requiring and additional term in the loss function that needs to be weighted, as in a variational autoencoder [38]. Hence, $\gamma$ is set to 1 only for training the model and $\sigma^2$ is defined according to the desired level of noise.

$$Z = Z^0 \left(\mathbf{1} + \gamma \mathcal{N}(0, \sigma^2)\right) \tag{1}$$

Following the latent space, the shape-generative model is divided into two branches. In the first, a multi-layer perceptron (MLP) selects the mesh template based on the similarity to the input shape. The MLP consists of two layers, where the first is activated with ReLU and the second with a softmax function. Thus, the index of the selected template corresponds to the index of the neuron with maximum softmax activation (winner-takes-all). In the second branch, a decoder predicts the displacement $\Delta V$ of the control points that parameterize the template. The decoder comprises three fully-connected layers with only the first two activated by ReLU.

The network forwards the selected template and predicted lattice deformation to an FFD operator [39]. We define a *template* as a pair $(B, V)$ of FFD parameters, where $B$ is the matrix of the tri-variate Bernstein polynomial coefficients, and $V$ is the vector with the coordinates of non-deformed control points. Each coefficient $b_{a,p}$ in $B$ is defined by the transformed coordinates $(s, t, u) \in [0, 1]^3$ of a point $a$ sampled from the template mesh (Eq. 2), and the position $p = (i, j, k)$ of a control point $v$ in a $(l \times m \times n)$-lattice (Eq. 3). Hence, the FFD operator that computes the $S_{def}$ deformed nodes of the template mesh is defined by $S_{def} = B(V + \Delta V)$.

$$(s_a, t_a, u_a) = \left(\frac{x_a - x_0}{x_{span}}, \frac{y_a - y_0}{y_{span}}, \frac{z_a - z_0}{z_{span}}\right) \tag{2}$$

$$b_{a,p} = (1-s)^{l-i} s^i \left[(1-t)^{m-j} t^j \left[(1-u)^{n-k} u^k\right]\right] \tag{3}$$

### 3.2. Data Set Processing

Prior to the training, we sample the input and template shapes with the shrink-wrapping algorithm proposed in [35]. First, we fit an initial mesh of a rectangular box to the dimensions of a target shape. Then, the algorithm iteratively approximates the vertices $\mathbf{x}_i$ of the initial shape to the corresponding nearest vertices $\mathbf{x}_{n,i}$ in the target shape, given a step size $\alpha$ (Eq. 4). After a defined number of iterations $t$, the algorithm relaxes the shrank mesh using a

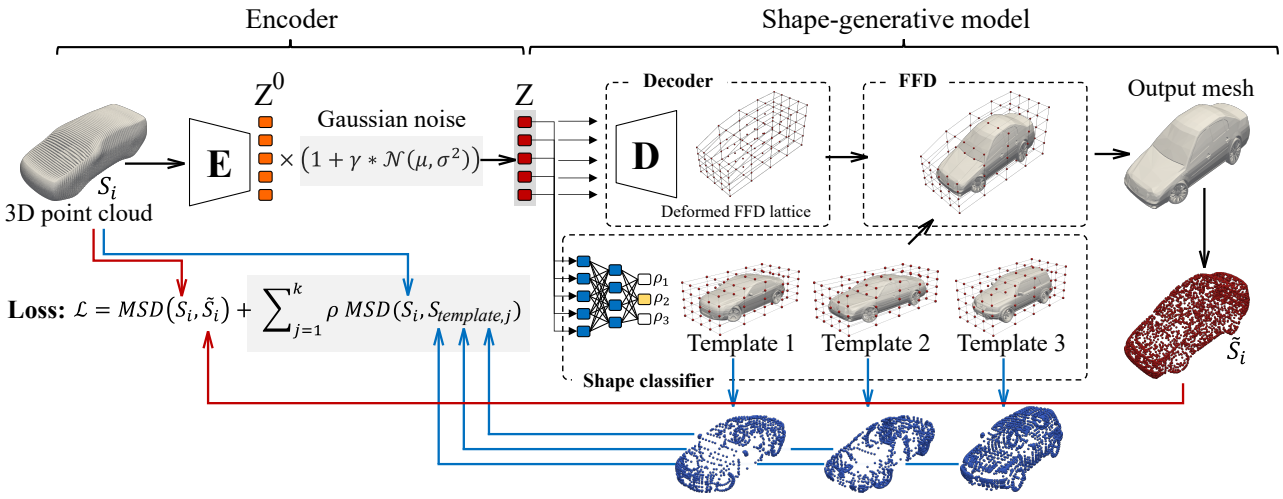


Figure 2. Point2FFD architecture and data flow for training the network.

Laplacian-based filter [44] to improve the distribution of the points. We shrink-wrap the geometries using step size $\alpha = 0.5$ [25] for six iterations, and smooth the shapes in a single step.

$$\mathbf{x}_i^{t+1} = \mathbf{x}_i^t + \alpha(\mathbf{x}_{n,i}^t - \mathbf{x}_i^t) \tag{4}$$

Shrink-wrapping generates organized 3D point clouds, which follow the vertex assignment of the initial mesh. Hence, since all shapes are sampled prior to the training and the point correspondence is known, it allows us to simplify the loss function and, thus, to increase the computational efficiency of the training algorithm.

### 3.3. Network Settings and Training

**Network hyperparameters.** The baseline hyperparameters of Point2FFD are based on the architectures proposed in [2, 38, 35] (Table 1). For training the model, we utilize $\gamma = 1$ and $\sigma^2 = 9.70\text{E-}03$ to restrict the noise to a level of 2.5% with 99% confidence interval. In shape-generative tasks, the noise is removed by setting $\gamma = 0$.

For training Point2FFD, the weights and biases are initialized randomly according to a normal distribution with mean $\mu = 0$ and standard deviation $\sigma^2 = 1\text{E-}02$. In particular, by initializing the decoder (layers $7_d$ to $9_d$) with lower $\sigma^2$ values avoids the network to learn local optima solutions by self-intersecting the template meshes.

**Loss function**. The loss function for training Point2FFD comprises 2 terms (Eq. 5). The first term computes the reconstruction error defined by the mean-squared distance (MSD) between corresponding points $\mathbf{x}_i, i = (1, ..., N)$ and $\tilde{\mathbf{x}}_i$ in the input and output point clouds, respectively. Since the similarity between template and input geometries improves the quality of the shape matching [33], the second term computes a weighted-average of the MSD between the available $K$ templates and input shapes. The weight for each template is defined by the corresponding output $\rho_j$ of the MLP-classifier. Thus, the second term is minimized when the MLP yields the highest selection probability $\rho_j$ to the template with lowest MSD value (highest similarity).

Table 1. Baseline hyperparameters of the Point2FFD architecture.

| Layer | Type | Activation | Features | Output |
|---|---|---|---|---|
| | | **Encoder** | | |
| 1 | 1D-C | ReLU | 64 | [N, 64] |
| 2 | 1D-C | ReLU | 128 | [N, 128] |
| 3 | 1D-C | ReLU | 128 | [N, 128] |
| 4 | 1D-C | ReLU | 256 | [N, 256] |
| 5 | 1D-C | tanh | 128 | [N, 128] |
| | | **Latent layer** | | |
| 6 | max-pool + $\mathcal{N}(0, \sigma^2)$ | | 128 | [1, 128] |
| | | **Decoder** | | |
| $7_d$ | FC | ReLU | 768 | [256, 3] |
| $8_d$ | FC | ReLU | 768 | [256, 3] |
| $9_d$ | FC | None | $3(lmn)$ | $[(lmn), 3]$ |
| | | **MLP** | | |
| $7_m$ | FC | ReLU | 25 | [25, 1] |
| $8_m$ | FC | softmax | $K$ | [K, 1] |

*Acronyms and variables:* 1D convolution (*1D-C*), point cloud size ($N$), fully-connected layer (FC) and number of available templates ($K$). The indices *m* and *d* in the layers indicate *MLP* and *decoder*, respectively.

$$\mathcal{L} = \frac{1}{N}\sum_{i=1}^{N} \|\mathbf{x}_i - \tilde{\mathbf{x}}_i\|^2 + \sum_{j=1}^{K} \frac{\rho_j}{N} \left( \sum_{i=1}^{N} \|\mathbf{x}_i - \mathbf{x}_i^j\|^2 \right) \tag{5}$$

Despite requiring ordered point clouds, MSD has advantages over permutation invariant functions, *e.g.* Cham-

fer Distance [12]. MSD is computationally more efficient, since the correspondence between points on different point clouds is known and fixed, and preserves the permutation-invariance of the features calculated by the encoder. For the same reason, MSD also avoids learning deformations with self-intersections, which reduce the quality of the shape reconstructions.

**Training algorithm**. In our experiments, we train Point2FFD using the Adam optimizer [24]. We set the algorithm with learning rate $\eta = 1.00\text{E-}04$, momenta $\beta_1 = 0.9$ and $\beta_2 = 0.999$, and maximum number of epochs to 750. We split the data set into 90% and 10% partitions for training and testing, respectively, which are fed to the network in randomly shuffled batches of 50 shapes.

## 4. Experiments

In this section, we assess the performance of Point2FFD in two sets of experiments. First, we benchmark our proposed network against the point cloud autoencoders on which we base our Point2FFD architecture. Second, we implement Point2FFD as shape-generative model in a real-world vehicle aerodynamic optimization problem, and compare the results to an analogous optimization performed with the point cloud autoencoder utilized in [35].

### 4.1. Verification Analyses

In our verification analyses, we utilized shapes from the car and airplane classes of ShapeNetCore [7]. We sampled the shapes according to the methods in Section 3.2 and tested Point2FFD in four different scenarios (Table 2). We evaluated the scalability of Point2FFD by increasing the point cloud size from *Scenario A* to *Scenario B*, as well as the performance of the classifier in *Scenario D* by mixing the object classes.

Table 2. Scenarios for benchmarking the networks.

| Scenario | Classes | Point cloud size | K |
|---|---|---|---|
| A | Car | 6146 | 1 |
| B | Car | 24578 | 1 |
| C | Airplane | 24578 | 1 |
| D | Car & Airplane | 24578 | 2 |

*Acronyms and variables:* Number of templates ($K$).

In all experiments, the data sets comprised 3450 randomly selected shapes and equal splits between the classes (*Scenario D*). Furthermore, we selected a single mesh template per class based on the similarity to the mean shape of each class (Fig. 3). We embedded the mesh templates in a lattice with $l = 16$, $m = 6$ and $n = 6$ control planes in s-, t- and u-direction, respectively.

We benchmarked Point2FFD against three point cloud autoencoders with similar architecture: The autoencoder

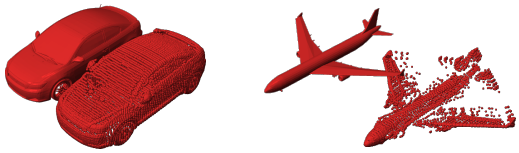

Figure 3. Mesh and sampled point clouds of the template models selected for the verification experiments.

proposed in [2] (PC-AE-Achlioptas), the modified version proposed in [35] (PC-AE-Rios), and the variational autoencoder proposed in [38] (PC-VAE). We selected the first two networks because PC-AE-Rios is trained with MSD instead of Chamfer Distance (CD) [12]. Hence, the differences in computational demand and reconstruction quality depend almost exclusively on the different loss functions. We also considered the PC-VAE in our analyses to compare the latent representation learned with Point2FFD to a regularized latent space. In all cases, we set the networks to the same latent space dimensionality (128) and utilized the networks' hyperparameters proposed in the reference work.

We implemented the architectures using Python with Tensorflow®for computation on Graphic Processing Units (GPU). In *Scenario A*, we trained the models on a machine with two CPUs Intel®Xeon®Silver, clocked at 2.10 GHz, and four GPUs NVidia®GeForce®RTX 2080 Ti with 12 GB each. For the remaining scenarios, we utilized a machine with similar CPUs, however with 2 GPUs NVidia®Quadro®RTX 8000 with 48 GB each. In all cases, the networks were trained on a single GPU and with the machines under similar workload.

We evaluated the performance of the networks based on three main criteria: number of parameters, training runtime and reconstruction losses (Table 3). The first two criteria provide an insight on the computational effort to generate the models. The third criterion, measured with CD, indicates the performance of the networks as shape-generative models.

**Number of parameters and runtime:** Since Point2FFD predicts the deformation of an FFD lattice, the number of trainable parameters is invariant with the point cloud size. However, the matrix $B$ of polynomial coefficients scales with the dimensionality of the point clouds and lattice. Hence, Point2FFD processes point clouds more efficiently, but requires larger memory allocation (total variables) to store the FFD parameters. This result is particularly evident in the Point2FFD runtime of *Scenarios B, C* and *D*, which is similar to PC-AE-Achlioptas. Furthermore, when comparing the runtime of PC-AE-Achlioptas and PC-AE-Rios, we concluded that the MSD reduces the computational effort compared to CD, especially in higher-dimensional point clouds, as previously discussed. In all cases, the PC-VAE required the highest computational effort, which was expected.

Table 3. Performance of the selected networks in learning shape-generative models in the *Scenarios A to D*.

| Scenario A: Car class, point cloud size: 6146 points | | | | |
|---|---|---|---|---|
| Network | PC-AE-Achlioptas | PC-AE-Rios | PC-VAE | Point2FFD |
| Loss function | CD [12] | MSD | $\alpha_1$CD + $\alpha_2$ KL-D | MSD + $\text{MSD}_{\text{proto}}$ |
| Free variables | 1 831 942 | 1 831 942 | 1 897 606 | **392 563** |
| Total variables | **1 831 942** | **1 831 942** | 1 897 606 | 3 934 387 |
| $\text{CD}_{training}$ | $(1.34 \pm 0.03)$E-04 | $(9.37 \pm 0.57)$E-05 | $(5.59 \pm 0.82)$E-04 | $(\mathbf{5.67 \pm 0.39})$E-05 |
| $\text{CD}_{test}$ | $(1.34 \pm 0.09)$E-04 | $(8.93 \pm 1.41)$E-05 | $(5.03 \pm 2.62)$E-04 | $(\mathbf{5.52 \pm 1.07})$E-05 |
| Runtime | 1 h 45 min 19 s | **1 h 42 min 24 s** | 2 h 34 min 2 s | 2 h 7 min 21 s |
| Scenario B: Car class, point cloud size: 24578 points | | | | |
| Free variables | 6 605 830 | 6 605 830 | 6 671 494 | **392 563** |
| Total variables | **6 605 830** | **6 605 830** | 6 671 494 | 14 552 219 |
| $\text{CD}_{training}$ | $(1.00 \pm 0.03)$E-04 | $(1.05 \pm 0.06)$E-04 | $(4.10 \pm 0.79)$E-04 | $(\mathbf{4.04 \pm 0.09})$E-05 |
| $\text{CD}_{test}$ | $(1.05 \pm 0.11)$E-04 | $(1.03 \pm 0.17)$E-04 | $(4.99 \pm 3.03)$E-04 | $(\mathbf{4.95 \pm 0.99})$E-05 |
| Runtime | 4 h 32 min 40 s | **3 h 42 min 13 s** | 6 h 39 min 58 s | 4 h 43 min 23 s |
| Scenario C: Airplane class, point cloud size: 24578 points | | | | |
| $\text{CD}_{training}$ | $(\mathbf{8.90 \pm 0.27})$E-05 | $(1.47 \pm 0.06)$E-04 | $(4.58 \pm 0.28)$E-04 | $(1.69 \pm 0.06)$E-04 |
| $\text{CD}_{test}$ | $(\mathbf{9.61 \pm 1.05})$E-05 | $(1.48 \pm 0.19)$E-04 | $(4.62 \pm 0.79)$E-04 | $(1.80 \pm 0.24)$E-04 |
| Runtime | 4 h 29 min 17 s | **3 h 42 min 22 s** | 6 h 31 min 50 s | 4 h 35 min 39 s |
| Scenario D: Car and airplane classes, point cloud size: 24578 points | | | | |
| Total variables | **6 605 830** | **6 605 830** | 6 671 494 | 28 709 875 |
| $\text{CD}_{training}$ | $(\mathbf{1.34 \pm 0.05})$E-04 | $(1.43 \pm 0.06)$E-04 | $(5.48 \pm 0.52)$E-04 | $(1.36 \pm 0.11)$E-04 |
| $\text{CD}_{test}$ | $(\mathbf{1.31 \pm 0.10})$E-04 | $(1.44 \pm 0.18)$E-04 | $(4.26 \pm 1.08)$E-04 | $(1.43 \pm 0.35)$E-04 |
| Runtime | 4 h 31 min 5 s | **3 h 43 min 3s** | 6 h 38 min 27 s | 4 h 39 min 53 s |

KL-D indicates the *Kullback-Leibler divergence* and the variables $\alpha_1$, $\alpha_2$ the weights for scaling the components of the loss function.

**Reconstruction losses:** We utilized CD to calculate the losses on the training and test sets. For a 95% confidence interval, we observed that Point2FFD achieved reconstruction quality at least comparable to the other methods. Since the networks trained with CD performed better on the data sets with airplane shapes, we visually inspected reconstructed samples from *Scenario D* (Fig. 4). We observed that PC-AE-Achlioptas and PC-VAE differentiated the shape topologies better than the other networks, but generated fuzzier point clouds. We also noticed that Point2FFD generated sharper geometries than PC-AE-Rios, which were over-smoothed. Therefore, we concluded that the shape reconstruction based on templates improves the quality of the generated models on networks trained with MDS-based loss functions (Fig. 4).

We also compared the templates assigned by Point2FFD to the classes of the shapes in *Scenario D*. We visualized the latent space by embedding the latent representations into a 2D space using UMAP [28] (Fig. 5). Point2FFD correctly identified $\approx 93\%$ of the labels and, despite the differences to the ground truth, the representations of airplanes and cars are clearly divided in the latent space.

**Conclusions:** In this set of experiments, we showed that Point2FFD achieved overall a better performance than similar 3D point cloud autoencoders considering computational effort quality of the generated shapes. We also showed that the network identifies correctly different classes of objects, even though the classifier learns the labels based on shape similarity. In the next set of experiments, we utilize Point2FFD as a shape-generative model in a real-world vehicle optimization problem, which requires high-quality surface reconstruction and has non-trivial shape labeling.

### 4.2. Vehicle Aerodynamic Optimization

The aerodynamic performance of vehicles is often associated to the aerodynamic drag. The power consumed by the drag force increases cubically with the vehicle's velocity and, thus, it has significant impact on the fuel consumption, especially at cruise speeds [10, 9].

In this section, we optimize three car shapes for minimizing the aerodynamic drag: a coupé, a sedan and a sport utility vehicle (SUV) (Fig. 6). We assume as underlying scenario that these vehicles were designed in previous product development cycles and their simulation-ready mesh representations are available. Hence, we use Point2FFD to learn design representations of benchmark car shapes based on the deformation of these templates to explore novel solutions during the optimizations.

**Experimental set up:** For this set of experiments, we trained Point2FFD with the same settings as in *Scenario A*

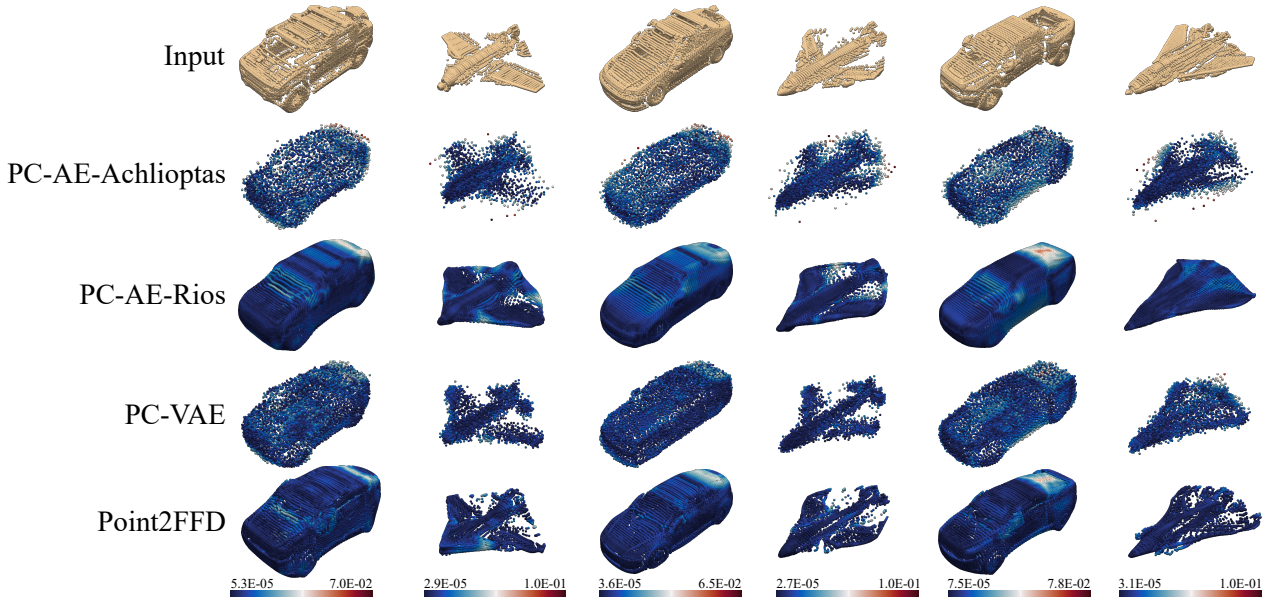


Figure 4. Analysis of shapes reconstructed by the models trained on the data set with car and airplane shapes (Scenario D). The colors indicate the distance between corresponding points in the input and output shapes.

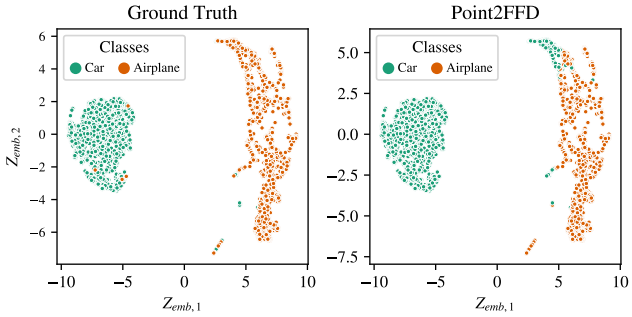


Figure 5. 2D-embedding of the learned latent representations colored according to the ground truth and predicted labels.

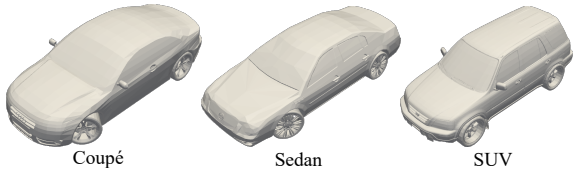


Figure 6. Initial designs of the coupé, sedan and SUV used in the optimizations.

(Section 4.1), apart from the latent space dimensionality, which we reduce to 50. Also, for comparing the optimization performance, we also trained the PC-AE-Rios using the same settings as Point2FFD. Since PC-AE-Rios generates only point cloud representations, we used the shrink-wrapping algorithm (Section 3.2) to recover the surfaces on the shapes generated by PC-AE-Rios.

We calculated the aerodynamic forces through CFD simulations implemented using OpenFOAM®. As simulation conditions, we assumed the vehicles driving in straight line with velocity $U = 110$ km/h. To reduce the computational effort, we assumed that the shapes are symmetric with respect to the xz-plane and, thus, performed half-car simulations.

Finally, we utilized the covariance matrix adaptation evolution strategy (CMA-ES), a state-of-the-art optimizer for computationally costly real-world design optimization [19]. In all cases, we set the population size to $\lambda = 16$, number of parents to $\mu = 5$ and maximum number of generations to 20. Our complete framework is computed in a cluster set up, which comprises supermicro-boards with 2 Westmere 4 Core Intel®Xeon®E5620, 2,4 GHz, 12 MB Cache and 24 GB of RAM. Each simulation is performed in parallel on 16 processors and with a runtime of approximately 2 hours.

**Results and discussion**: Based on the fitness of the individuals over the generations (Fig. 7), the optimizations with Point2FFD achieved results significantly better (>50%) than with PC-AE-Rios. Furthermore, by verifying the optimized shapes, we observe that Point2FFD yielded designs similar to drag-optimized milage marathon vehicles [1]. Despite the difference to the initial design, this result indicates that Point2FFD combined different features that were learned from the data during the optimizations to drive the design towards the optimality.

We conclude from these results that two main factors cause the difference in performance: The shape-generative capability of the networks and realism of the generated shapes. In terms of shape-generative capability, the verification analyses in Section 4.1 indicated that both net-

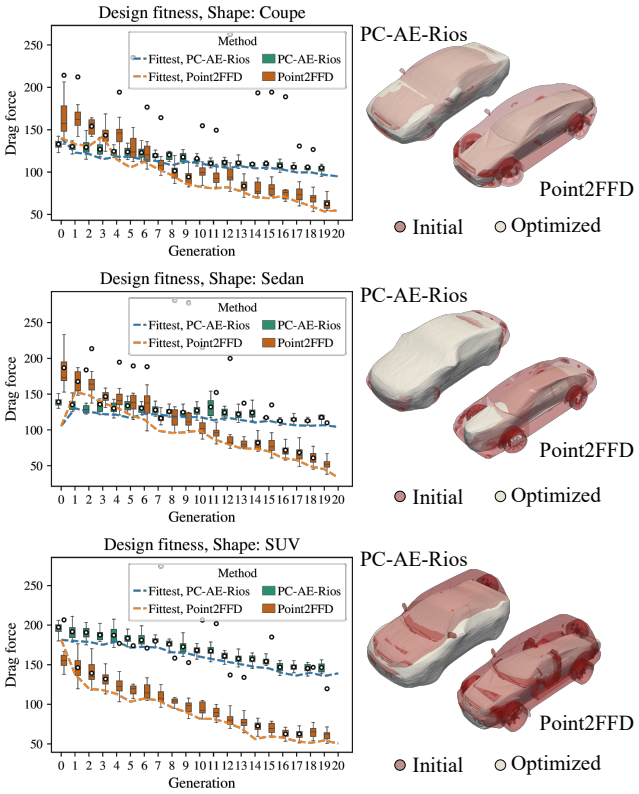


Figure 7. Normalized performance of the fittest individuals per generation and optimized shapes obtained with Point2FFD and PC-AE-Rios after 20 generations.

works have comparable performance. However, by generating shapes with FFD, Point2FFD potentially modifies shapes more smoothly, which enables the network to better explore unknown regions in the latent space than PC-AE-Rios.

The similarity of the reconstructed shapes to real car designs influences the quality of the simulation results. Since shrink-wrapping approximates coarsely the surface of the car shapes, the generated meshes lack detailed structures, *e.g.* wheels (Fig. 8). Differently, Point2FFD generates representations with higher degree of realism, which allowed the optimization algorithm to better exploit local design features and generate designs with better performance.

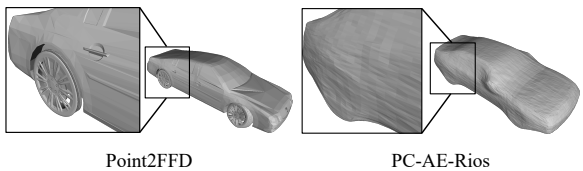


Figure 8. Comparison between meshes generated by Point2FFD and shrink-meshing for similar shapes at the first generation of the sedan optimization.

## 5. Conclusion

Motivated by the current challenges for utilizing deep shape-generative models in engineering optimization, we propose in this paper a novel method for learning representations of simulation-ready 3D geometric models. Our architecture learns compact latent representations from 3D point cloud data and generates simulation-ready polygonal meshes by selecting and deforming available template meshes parameterized with FFD.

Compared to similar point cloud autoencoders, our method has advantages in both computational efficiency and quality of the shape reconstructions. Although the representation of the FFD-templates is memory demanding, the number of parameters in Point2FFD is invariant with respect to the dimensionality of the input and output representations. Hence, our architecture scales to higher-dimensional point clouds with competitive performance.

Furthermore, by training the network with both, MSD losses and template-based shape reconstructions, we enabled the network to learn a smoother shape-generative model. We noticed in Section 4.1 that learning data sets with more complex shapes (airplanes) was more challenging for Point2FFD. However, we explored a single FFD lattice configuration and suited the shrink-wrapping sampling to the vehicle data set, which was our main application. Hence, we conclude that the Point2FFD still has potential to improve its performance by utilizing template-specific FFD lattices and different point cloud sampling techniques, which we will address in future work.

In a set of real world-inspired optimizations, we utilized the representation learned by Point2FFD to minimize the aerodynamic drag of three car shapes. We showed that our network generates polygonal meshes that preserve shape details, *e.g.*, wheel rims, which improve the fidelity of the simulation results. Furthermore, compared to a conventional point cloud autoencoder, our network enabled the optimization algorithm to better explore the design space and to find higher-quality optima solutions. In future work, we will build upon our architecture to address more complex optimization scenarios, *e.g.*, multi-domain optimization and increase the degree of control over the deformations for handling design constraints.

## Acknowledgement

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement number 766186 (ECOLE).

## References

[1] Essam Abo-serie, E. Oran, and O. Utcu. Aerodynamics Assessment Using CFD for a Low Drag Shell Eco-Marathon Car. *Journal of Thermal Engineering*, 3:1527–1536, 2017. 7

[2] Panos Achlioptas, Olga Diamanti, Ioannis Mitliagkas, and Leonidas Guibas. Learning Representations and Generative Models for 3D Point Clouds. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 40–49. PMLR, 10–15 Jul 2018. 1, 2, 4, 5

[3] Matthew Berger, Andrea Tagliasacchi, Lee M. Seversky, Pierre Alliez, Joshua A. Levine, Andrei Sharf, and Claudio T. Silva. State of the Art in Surface Reconstruction from Point Clouds. In Sylvain Lefebvre and Michela Spagnuolo, editors, *Eurographics 2014 - State of the Art Reports*. The Eurographics Association, 2014. 1

[4] Fausto Bernardini, Joshua Mittleman, Holly Rushmeier, Cláudio Silva, and Gabriel Taubin. The Ball-Pivoting Algorithm for Surface Reconstruction. *IEEE Transactions on Visualization and Computer Graphics*, 5(4):349–359, Oct. 1999. 2

[5] Michael M. Bronstein, Joan Bruna, Yann LeCun, Arthur Szlam, and Pierre Vandergheynst. Geometric Deep Learning: Going beyond Euclidean Data. *IEEE Signal Processing Magazine*, 34(4):18–42, 2017. 1

[6] Wenming Cao, Zhiyue Yan, Zhiquan He, and Zhihai He. A Comprehensive Survey on Geometric Deep Learning. *IEEE Access*, 8:35929–35949, 2020. 2

[7] 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. 5

[8] R. Qi Charles, Hao Su, Mo Kaichun, and Leonidas J. Guibas. PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 77–85, 2017. 2

[9] Harun Chowdhury, Hazim Moria, Abdulkadir Ali, Iftekhar Khan, Firoz Alam, and Simon Watkins. A Study on Aerodynamic Drag of a Semi-trailer Truck. *Procedia Engineering*, 56:201–205, 2013. 5th BSME International Conference on Thermal Engineering. 6

[10] Laurent Dumas. *CFD-based Optimization for Automotive Aerodynamics*, pages 191–215. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008. 6

[11] Giles Endicott, Toyotaka Sonoda, Markus Olhofer, and Toshiyuki Arima. Aerodynamic Improvement of a Transonic Fan Outlet Guide VaneUsing 3D Design Optimization. volume Volume 7: Turbomachinery, Parts A, B, and C of *Turbo Expo: Power for Land, Sea, and Air*, pages 1395–1404, 06 2011. 2

[12] Haoqiang Fan, Hao Su, and Leonidas Guibas. A Point Set Generation Network for 3D Object Reconstruction from a Single Image. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2463–2471, 2017. 5, 6

[13] Matheus Gadelha, Rui Wang, and Subhransu Maji. Multiresolution Tree Networks for 3D Point Cloud Processing. In Vittorio Ferrari, Martial Hebert, Cristian Sminchisescu, and Yair Weiss, editors, *Computer Vision – ECCV 2018*, pages 105–122, Cham, 2018. Springer International Publishing. 2

[14] Lin Gao, Jie Yang, Tong Wu, Yu-Jie Yuan, Hongbo Fu, Yu-Kun Lai, and Hao Zhang. SDM-NET: Deep Generative Network for Structured Deformable Mesh. *ACM Trans. Graph.*, 38(6), Nov. 2019. 1, 3

[15] Thibault Groueix, Matthew Fisher, Vladimir G. Kim, Bryan C. Russell, and Mathieu Aubry. 3D-CODED: 3D Correspondences by Deep Deformation. In Vittorio Ferrari, Martial Hebert, Cristian Sminchisescu, and Yair Weiss, editors, *Computer Vision – ECCV 2018*, pages 235–251, Cham, 2018. Springer International Publishing. 3

[16] Yulan Guo, Hanyun Wang, Qingyong Hu, Hao Liu, Li Liu, and Mohammed Bennamoun. Deep Learning for 3D Point Clouds: A Survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1–1, 2020. 1, 2

[17] James Hall, Daniel J. Poole, T. Rendall, and Christian B. Allen. *Volumetric Shape Parameterisation for Combined Aerodynamic Geometry and Topology Optimisation*, chapter 3354, pages 1–29. American Institute of Aeronautics and Astronautics, Inc., 2015. 2

[18] Rana Hanocka, Gal Metzer, Raja Giryes, and Daniel Cohen-Or. Point2Mesh: A Self-Prior for Deformable Meshes. *ACM Trans. Graph.*, 39(4), July 2020. 1, 3

[19] N. Hansen. The CMA Evolution Strategy: A Tutorial. *ArXiv*, abs/1604.00772(arXiv:1604.00772 [cs.LG]), 2016. 7

[20] Amir Hertz, Rana Hanocka, Raja Giryes, and Daniel Cohen-Or. PointGMM: A Neural GMM Network for Point Clouds. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 12051–12060, 2020. 2

[21] Alexander Hornung and Leif Kobbelt. Robust Reconstruction of Watertight 3D Models from Non-Uniformly Sampled Point Clouds without Normal Information. In *Proceedings of the Fourth Eurographics Symposium on Geometry Processing*, SGP '06, page 41–50, Goslar, DEU, 2006. Eurographics Association. 2

[22] Mikael Kaandorp, Stefan Menzel, and Sebastian Schmitt. An Aerodynamic Perspective on Shape Deformation Methods. In *18th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*. American Institute of Aeronautics and Astronautics, June 2017. 2

[23] Michael Kazhdan, Matthew Bolitho, and Hugues Hoppe. Poisson Surface Reconstruction. In *Proceedings of the Fourth Eurographics Symposium on Geometry Processing*, SGP '06, page 61–70, Goslar, DEU, 2006. Eurographics Association. 2

[24] Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. *CoRR*, abs/1412.6980(arXiv:1412.6980 [cs.LG]), 2015. 5

[25] Bon Ki Koo, Chang Woo Chu, Jae Chul Kim, and Young Kyu Choi. Srink-Wrapped Boundary Face Algorithm for Surface Reconstruction from Unorganized 3D Points. In *Proceedings of the 4th WSEAS International Conference on Signal Processing, Computational Geometry & Artificial Vision*, ISCGAV'04, Stevens Point, Wisconsin, USA, 2004. World Scientific and Engineering Academy and Society (WSEAS). 4

[26] Jiaxin Li, Ben M. Chen, and Gim Hee Lee. SO-Net: Self-Organizing Network for Point Cloud Analysis. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9397–9406, 2018. 2

[27] Jun Li, Kai Xu, Siddhartha Chaudhuri, Ersin Yumer, Hao Zhang, and Leonidas Guibas. GRASS: Generative Recursive Autoencoders for Shape Structures. *ACM Trans. Graph.*, 36(4), July 2017. 1, 3

[28] L. McInnes and John Healy. UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction. *ArXiv*, abs/1802.03426(arXiv:1802.03426 [stat.ML]), 2018. 6

[29] Stefan Menzel, Markus Olhofer, and Bernhard Sendhoff. Application of Free Form Deformation Techniques in Evolutionary Design Optimisation. In Jose Herskovits, Sandro Mazorche, and Alfredo Canelas, editors, *6th World Congress on Structural and Multidisciplinary Optimization (WCSMO6)*, Rio de Janeiro, 2005. COPPE Publication. 2

[30] Stefan Menzel and Bernhard Sendhoff. Representing the Change - Free Form Deformation for Evolutionary Design Optimisation. In Tina Yu, David Davis, Cem Baydar, and Rajkumar Roy, editors, *Evolutionary Computation in Practice*, chapter 4, page 63–86. Springer, Berlin, 2008. 2

[31] Markus Olhofer, Thomas Bihrer, Stefan Menzel, Michael Fischer, and Bernhard Sendhoff. Evolutionary Optimisation of an Exhaust Flow Element with Free Form Deformation. In K.W. Seibert and M. Jirka, editors, *Simulation for Innovative Design, Proceedings of the 4th EASC - 2009 European Automotive Simulation Conference*, pages 163–174. ANSYS Inc., 2009. 2

[32] Charles R. Qi, Li Yi, Hao Su, and Leonidas J. Guibas. PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 5105–5114, Red Hook, NY, USA, 2017. Curran Associates Inc. 2

[33] Thiago Rios, Jiawen Kong, Bas van Stein, Thomas Bäck, Patricia Wollstadt, Bernhard Sendhoff, and Stefan Menzel. Back To Meshes: Optimal Simulation-ready Mesh Prototypes For Autoencoder-based 3D Car Point Clouds. In *2020 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 942–949, 2020. 3, 4

[34] Thiago Rios, Bernhard Sendhoff, Stefan Menzel, Thomas Bäck, and Bas van Stein. On the Efficiency of a Point Cloud Autoencoder as a Geometric Representation for Shape Optimization. In *2019 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 791–798, 2019. 2

[35] Thiago Rios, Bas van Stein, Thomas Bäck, Bernhard Sendhoff, and Stefan Menzel. Multi-Task Shape Optimization Using a 3D Point Cloud Autoencoder as Unified Representation. *IEEE Transactions on Evolutionary Computation*, pages 1–1, 2021. 2, 3, 4, 5

[36] Thiago Rios, Bas van Stein, Stefan Menzel, Thomas Back, Bernhard Sendhoff, and Patricia Wollstadt. Feature Visualization for 3D Point Cloud Autoencoders. In *2020 International Joint Conference on Neural Networks (IJCNN)*, pages 1–9, 2020. 1, 2

[37] Thiago Rios, Patricia Wollstadt, Bas van Stein, Thomas Bäck, Zhao Xu, Bernhard Sendhoff, and Stefan Menzel. Scalability of Learning Tasks on 3D CAE Models Using Point Cloud Autoencoders. In *2019 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 1367–1374, 2019. 1

[38] Sneha Saha, Stefan Menzel, Leandro L. Minku, Xin Yao, Bernhard Sendhoff, and Patricia Wollstadt. Quantifying The Generative Capabilities Of Variational Autoencoders For 3D Car Point Clouds. In *2020 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 1469–1477, 2020. 1, 2, 3, 4, 5

[39] Thomas W. Sederberg and Scott R. Parry. Free-Form Deformation of Solid Geometric Models. *SIGGRAPH Comput. Graph.*, 20(4):151–160, Aug. 1986. 2, 3

[40] Nicholas Sharp and Maks Ovsjanikov. PointTriNet: Learned Triangulation of 3D Point Sets. In Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm, editors, *Computer Vision – ECCV 2020*, pages 762–778, Cham, 2020. Springer International Publishing. 1, 3

[41] Daniel Sieger, Stefan Menzel, and Mario Botsch. On Shape Deformation Techniques for Simulation-based Design Optimization. In Luca Formaggia Simona Perotto, editor, *New Challenges in Grid Generation and Adaptivity for Scientific Computing*, SEMA SIMAI Springer Series, pages 281–303. Springer, LNCS, LNAI, LNBI, May 2015. 2

[42] S.N. Skinner and H. Zare-Behtash. State-of-the-art in Aerodynamic Shape Optimisation Methods. *Applied Soft Computing*, 62:933–962, 2018. 1, 2

[43] Nobuyuki Umetani. Exploring Generative 3D Shapes Using Autoencoder Networks. In *SIGGRAPH Asia 2017 Technical Briefs*, SA '17, New York, NY, USA, 2017. Association for Computing Machinery. 1, 2

[44] J. Vollmer, R. Mencl, and H. Müller. Improved Laplacian Smoothing of Noisy Surface Meshes. *Computer Graphics Forum*, 18(3):131–138, 1999. 4

[45] Weiyue Wang, Duygu Ceylan, Radomir Mech, and Ulrich Neumann. 3DN: 3D Deformation Network. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1038–1046, 2019. 1, 3

[46] Yaoqing Yang, Chen Feng, Yiru Shen, and Dong Tian. FoldingNet: Point Cloud Auto-Encoder via Deep Grid Deformation. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 206–215, 2018. 2

[47] Bin Zhang, Zhiwei Feng, Boting Xu, and Tao Yang. Efficient Aerodynamic Shape Optimization of the Hypersonic Lifting Body Based on Free Form Deformation Technique. *IEEE Access*, 7:147991–148003, 2019. 2

[48] Yu Zhang, Zhong-Hua Han, Laixiang Shi, and Wen-Ping Song. *Multi-round Surrogate-based Optimization for Benchmark Aerodynamic Design Problems*. 2016. 2