Back To Meshes: Optimal Simulation-ready Mesh Prototypes For Autoencoder-based 3D Car Point Clouds

Thiago Rios*, Jiawen Kong†, Bas van Stein†,
Thomas Bäck†, Patricia Wollstadt*, Bernhard Sendhoff* and Stefan Menzel*
*Honda Research Institute Europe GmbH, Carl-Legien-Str. 30, 63073 Offenbach, Germany
†Leiden Institute of Advanced Computer Science (LIACS), Niels Bohrweg 1, 2333 CA Leiden, The Netherlands
Email: {thiago.rios, patricia.wollstadt, bernhard.sendhoff, stefan.menzel}@honda-ri.de,
{j.kong, b.van.stein, t.h.w.baeck}@liacs.leidenuniv.nl

***Abstract*—Point cloud autoencoders were recently introduced as powerful models for data compression. They learn a low-dimensional set of variables that are suitable as design parameters for shape generation and optimization problems. In engineering tasks, 3D point clouds are often derived from fine polygon meshes, which are the most suitable representations for physics simulation, e.g., computational fluid dynamics (CFD). Yet, the reconstruction of high-quality meshes from autoencoder-based point clouds is challenging, often requiring supervised and manual work, which is prohibitive during the optimization. Target shape matching optimization using existing mesh prototypes overcomes the difficulties of recovering shape information from the point coordinates. However, for autoencoders trained on data sets comprising shapes with high degree of dissimilarity, there is not a single mesh prototype that can fit any autoencoder-based point cloud, and the selection of a set of prototypes is nontrivial. In the present paper we propose a method for optimizing a selection of prototypical meshes to match the maximum number of shapes in the autoencoder output space as possible, which is achieved by linking the advantages of the latent space representation of an autoencoder and the state-of-the-art free form deformation (FFD) method. Furthermore, we approached the balance between costs (number of mesh prototypes) and number of covered shapes by varying the number of prototypes and the dimensionality of the autoencoder latent space, showing that higher-dimensional latent spaces encode finer geometric changes, requiring more sophisticated FFD setups.**

***Index Terms*—3D point cloud, meshing, geometric deep learning, evolutionary optimization**

I. Introduction

Point cloud autoencoders stand out from the literature on geometric deep learning as powerful shape generation techniques. These architectures compress 3D geometric data into a compact set of design variables, the so-called latent space, while learning an efficient method for reconstructing geometries [1], [2]. In automated computational shape optimization problems, e.g. aerodynamic design optimization of vehicles, the latent variables could perform as shape parameters. While the optimization algorithm searches iteratively for solutions in the latent space, the trained decoder retrieves the shapes as 3D point clouds, which serve as input to downstream tasks. Nevertheless, different aspects hinder the implementation of these methods. Among them are the interpretability of the latent variables [3], [4], the sparsity of engineering data sets and, ultimately, the need to recover surface meshes from the 3D point clouds generated by the autoencoder for performing downstream tasks, such as automated performance simulations of novel design proposals.

Recovering surface meshes from the Cartesian coordinates of a point set is an ill-posed problem. For a single point cloud, there is an infinite number of solutions that approximate the surface of the geometry. Thus, the available methods on mesh generation from point clouds either require additional information, e.g. surface normal vectors, or manual tuning through visual inspection [5], where the latter is especially costly. In order to automate the surface mesh reconstruction, we propose to use free form deformation (FFD) to manipulate prototypical meshes and match them to the generated point clouds. FFD is a well-established shape morphing technique that enables the optimizers to directly operate on complex computer aided engineering (CAE) meshes, as required e.g. for computer fluid dynamics (CFD) simulations, in shape optimization problems, reducing the computational costs during optimization by avoiding remeshing [6]

Therefore, in the present paper, we propose to link both representations and search for an optimal number of required mesh prototypes which can be altered through FFD manipulations to match as many shapes in the 3D point cloud autoencoder output space as possible. Concurrently to the maximum coverage, which determines design flexibility, we aim at minimizing the number of mesh prototypes, since each of them has high meshing costs. These optimal prototypes potentially allow us to perform an automated design optimization based on the latent parameters of a 3D point cloud autoencoder, where we combine the shape generative power of the deep

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

neural network, which can represent shapes with high degree of dissimilarity, to a fast setup of CAE models for engineering performance simulations, with FFD.

In terms of technical realization, we envision our approach within a fully automated shape optimization workflow (Fig. 1). In a preprocessing step, the autoencoder is trained on point clouds sampled from a data set of CAE meshes. e.g. car shapes. Based on the learned latent space, and according to our proposal in the present paper, the set of prototypical meshes is optimized, which remains available for the shape optimization task. Then, for each solution proposed by the optimizer, we select a mesh from the set of prototypes, which can be matched to the autoencoder-based point cloud through target shape matching optimization with FFD and forwarded to engineering CAE analyses, such as CFD. According to the performance, the algorithm either terminates or iterates with a different solution, until it converges to an optimum. Hence, combining the deep learning and FFD representations potentially decreases the dependency of the parameterization on the user expertise, while increasing the range of solutions that can be achieved, compared to the FFD parameterization only.

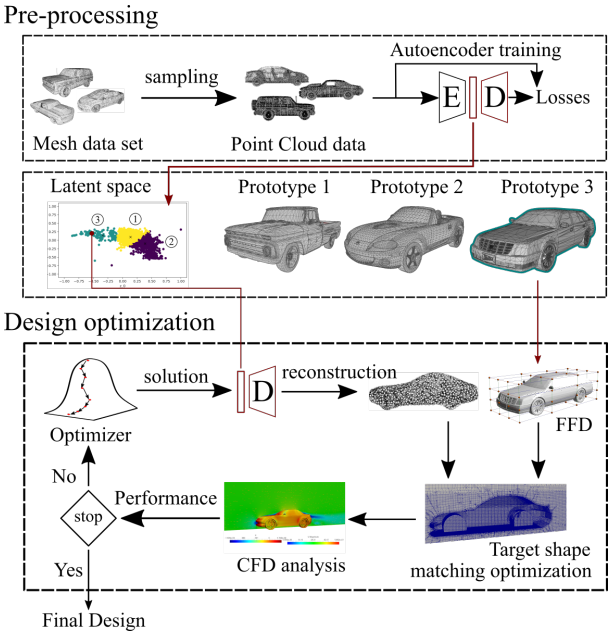

Fig. 1. Realization of the proposed method for an aerodynamic optimization.

The remainder of the paper is organized as follows: in Section II, we review the work related to point cloud autoencoders, surface recovery, meshing and shape morphing, which is central for the justification of our study. Then, based on the review, we describe our approach in Section III, providing details on the architecture of the point cloud autoencoder, FFD settings and workflow of the experiments. In Section IV, we discuss the results of the experiments performed on the shapes in the car class from ShapeNetCore [7], including an analysis on the trade-off between costs for generating the mesh prototypes and coverage, considering different number of prototypes and dimensionality of the latent representations. Finally, we conclude the paper with a summary and outlook in Section V.

II. Related work

The popularization of 3D-sensor technology and development of powerful graphic processing units (GPUs) enabled the recent advances in the field of geometric processing. They allowed the development of complex deep neural network models that learn on geometric data and, besides solving classical machine learning tasks, could be used as shape generators [2]. Among the explored geometric representations, 3D-point clouds are the simplest and most flexible [1], [5], [8] with compatibility to a wide range of applications, such as autonomous driving, scene reconstruction and fast prototyping [9]–[12]. Furthermore, point clouds can be sampled from more complex representations, such as meshes and voxels, allow for shape transformations with topological change.

Yet, learning on point cloud data is considerably challenging, due to the unstructured nature of the data and the fact that models have to be invariant against permutations in the input point cloud [13]. In [14], the authors provide a comprehensive survey on the architectures for processing point clouds, presenting also benchmark results on different data sets. One of the proposed classes comprises point-based networks which successfully compressed and reconstructed point cloud representations, such as [13], [15]–[17]. In these architectures, the encoder operates in a point-wise fashion and extracts the latent representation with a global operator. Thus, the features calculated by the network also become invariant to the permutation of points [18]. Furthermore, in order to tackle the problem of calculating a distance loss function between two unordered sets of points clouds, these architectures were trained with loss functions that can cope with non-Euclidean representations, such as the Chamfer and Earth Mover's distances [19], [20]. Once trained, the decoder could be used as shape generative model, by recovering the Cartesian coordinates of the points from samples in the latent space, which also allows for geometric operations, such as shape interpolation [15].

Due to their shape generation capabilities, point cloud autoencoders have a promising potential as design representation in engineering optimization problems. However, engineering simulations often adopt numerical algorithms that rely on meshed discretization, such as finite volume methods. Hence, apart from autoencoders trained on point clouds that correspond to the vertices of isomorphic meshes, as in [3], [8], which preserve the ordering of the points, the surface reconstruction from generated point clouds has to be addressed in a post-processing task.

Recovering meshes from Cartesian coordinates is an ill-posed problem, i.e., the solution space is infinite. Therefore, many of the available methods require additional information on the geometric properties, such as surface normals and topology. In [5], the authors reviewed several methods for

surface reconstruction and proposed taxonomy of the methods based on the working principle of each algorithm, and compared the sensitivity to point cloud artifacts, required input prior information, target shape class and type of output representation. From the methods presented in the paper, two re applicable to point clouds generated by an autoencoder: the tangent planes and graph cut algorithms.

The tangent plane algorithm [21] operates locally and remeshes the point clouds in two steps: tangent plane approximation and contouring. In the first step, the algorithm calculates a signed distance function on a set of neighbor points that is used to estimate the normal and central position of the tangent plane to this set, which is later refined into a surface by the contouring algorithm. However, due to the signed nature of the distance function, [21] claimed that the estimate of the normal was sensitive to noise and thus to the selection of the point neighborhood size, which often decreased the smoothness of the reconstructed surfaces.

The graph cut algorithm [22] overcomes this difficulty by adopting an unsigned distance metric, relaxing the constraints on the estimate of the surface local orientation. Initially, the method calculates a confidence interval over a uniform grid, which indicates the probability that the surface passes through a position (of the grid) in the space. Then, it generates a weighted graph in space, on which a minimum cut problem is solved, providing the minimum subset of the graph that maximizes the total confidence. Although requiring voxelization, [22] reported that the algorithms performed rather efficiently, e.g. in reconstructing the Dragon from the Stanford 3D scan repository with 318000 nodes in 150 s, and could be coupled with further mesh refinement algorithms.

In addition to the review in [5], the ball pivoting algorithm (BPA) [23], PointTriNet [24] and Point2Mesh [25] also approximate surface meshes on unordered point clouds. The principle of the BPA is to triangulate every three points that fit into a ball of radius ρ without containing any further point inside the ball, and to recursively pivot the ball around the triangulated edges. Due to its simplicity, the BPA requires low memory resources and the time complexity scales linearly with the size of the models. However, it workes by exhausting the triangulation alternatives and underperformes on point clouds with highly heterogeneous sampling [23].

In [24], the authors approached the triangulation of points with the PointTriNet, which comprised two deep neural network models: one for classifying points as node candidates and another to suggest new candidates. Since the networks were trained on local information, the authors claimed that the approach was scalable to higher dimensional models. However, their proposal tackled the triangulation of points rather than surface reconstruction and required post processing tasks for vanishing artifacts and smoothing the mesh. Similarly, the authors of Point2Net [25] approached the remeshing task with a convolutional neural network (CNN). The model was trained for iteratively deforming an initial mesh into the shape generated by the autoencoder, using the output of the layers as shape priors. The authors claimed that CNNs learn preferably recurring structures and correlations. Therefore, they inherently smooth out noise and imperfections in the point clouds. However, training the architecture required high computational power and, similar to morphing approaches, the matching between the initial mesh to the output point cloud relied on shape similarities and the quality of the initial geometry.

Addressing the challenges of mesh reconstruction and avoiding high computational costs, we we here approximate the point cloud surface using mesh deformation algorithms. These morphing techniques were already explored in the literature on shape optimization [26]–[29], and provide an intuitive low-dimensional design representation. A representative algorithm in this class is free form deformation (FFD) [30], which maps the geometry to a uniform control lattice, using trivariate Bernstein polynomials, as an $\mathbb{R}^3 \rightarrow \mathbb{R}^3$ function. Hence, when deforming the lattice, the embedded shape deforms accordingly, which allows for continuously morphing the initial shape into a geometry of interest. Furthermore, depending on the density of the lattice, FFD preserves the continuity of the surfaces at high-order and potentially the quality of the initial mesh.

III. Methods

In our approach, we aimed at finding the optimal set of prototypical meshes to match autoencoder-based 3D point clouds using FFD. Therefore, we initially provide the necessary background on the autoencoder architecture and FFD settings. Then, we describe our three-step approach, which starts with a feasibility assessment and leads to the experiments for optimizing the set of prototypical meshes.

A. 3D point cloud autoencoder

Our 3D point cloud autoencoder was based on the proposal in [15]. It comprised an encoder with five 1D-convolutional layers, followed by a *max pooling* layer to extract the latent representations, and a decoder with three fully connected layers (Tab. I). From the implementation in [15], we modified the activation function in the last convolutional layer from rectified linear unit (ReLU) to hyperbolic tangent, and added a sigmoid function as activation to the ouput layer. Hence, the latent variables were constrained to the space $\mathbf{Z} \in [-1, 1]^L$, easing the definition of constraints for search and sampling in the latent space, and the output Cartesian coordinates to $\mathbf{x} \in [0, 1]^3$, which corresponds to input space of the autoencoder. For further details of the architecture, as well as the validation and comparison to the original implementation, see [18], [31].

For the purposes of this paper, the autoencoder was trained on the shapes from the car class of ShapeNetCore [7]. In order to approximate the dimensionality of the data to industry-like CAE data, the car meshes were refined by recursively applying the midpoint subdivision algorithm [32] up to eight iterations with an earlier stop criterion in case the maximum edge length was smaller than 1E-04 (criteria defined experimentally, results not shown). The point clouds were sampled from the vertices

TABLE I
ARCHITECTURE OF OUR 3D POINT CLOUD AUTOENCODER.

Layer	Type	Activation	Features	Output dimensions
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	L	[N×L]
6	maxPool	-	L	[1×L]
7	FC	ReLU	256×3	[256×3]
8	FC	ReLU	256×3	[256×3]
9	FC	sigmoid	N×3	[N×3]

1D-C: 1D-convolution L: Number of latent variables
N: Size of the point cloud FC: *Fully connected*

of the meshes according to a uniform probability distribution, generating approximately 3500 point clouds with 8192 points each. The data set was split into 90/10 % partitions for training and testing the autoencoder, respectively, without applying any data augmentation technique.

The weights were optimized using the AdamOptimizer [33], with a learning rate $\eta = 5E-4$, $\beta = 0.99$ and $\beta_1 = 0.9$. The training data were randomly organized in batches of 50 shapes and trained over 700 epochs. The losses were determined as the mean Chamfer distance (CD) [19] between the input and recovered point clouds in a batch. The auencoders were trained on a machine with 2 CPUs Intel Xeon Silver 4110, clocked at 2.10 GHz, with 4 GPUs NVIDIA GeForce RTX 2080 Ti.

B. FFD settings

The FFD lattice comprised 64 control points, uniformly distributed in the three Cartesian directions (Fig. 2). We selected this configuration due to its simplicity, reducing the computational effort to compute the deformations, and to ensure C^3-continuity (curvature) of the surfaces in any direction.

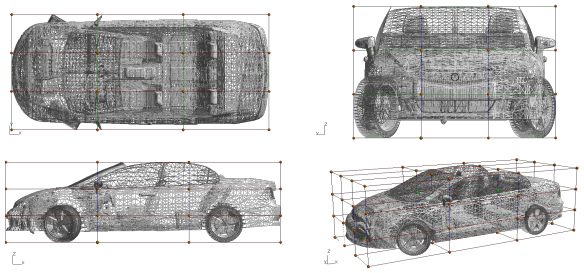

Fig. 2. Example of a prototypical mesh in the FFD lattice used in the experiments.

The shapes were deformed by the displacement of the eight central points of the lattice. In order to impose symmetric deformations, the position of the control points was mirrored with respect to the xy-plane (vertical-longitudinal midplane). Therefore, the deformations were parameterized by the Cartesian coordinates of only four control points, yielding twelve parameters.

In order to avoid generating shapes that could not be interpreted by the autoencoder, the displacement of the control points was restricted to 50% of the lattice span for each direction. We reached this value through an experiment, where we considered 50 different combinations of maximum displacement in Cartesian direction, up to 1.2 times the span of the lattice. For each case, we randomly generated 30 deformations for three samples of car shapes taken from ShapeNetCore and calculated the mean reconstruction loss using the autoencoder trained on a 2D-latent space. Analyzing how the losses varied with respect to the maximum displacements, we observed that the autoencoder was more sensitive to the deformations in the x-direction (Fig. 3). The behavior is justified by the normalization of the input data, which considered the length of the longest car in the data set as a reference metric. Since the sizes in the transversal directions are often smaller than the length, the boundaries of the input space became closer to the shape extremes in the x-direction. Therefore, when excessively deformed, the point cloud is displaced to the outside of the training input domain, leading to worse reconstructions.

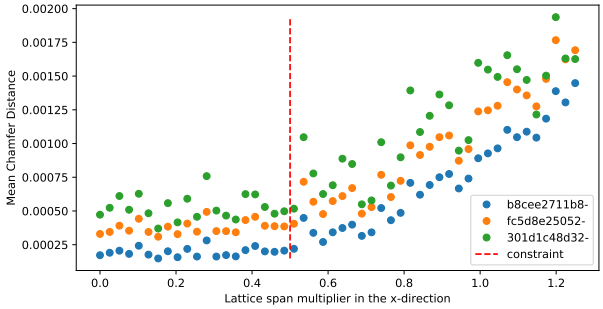

Fig. 3. Analysis of the reconstruction losses for different ranges of deformations in the x-direction. The shape ID in the legend is truncated for visualization purposes.

C. Optimizing the set of prototypical meshes

Based on the provided background, we propose a method for searching mesh prototypes using representations in the latent space. We assume that similar shapes yield neighboring representations in the latent space. Since the similarity between shapes is central to the quality of the target shape matching solution, we expect that a prototype can be matched to any design in the latent space within a hypervolume defined by the deformations at the limits of the FFD parameters.

Nevertheless, the correspondence between shape similarity and position in the latent space is not enforced by the training algorithm of the autoencoder. Additionally, the quality of the target shape matching result also varies with respect to the lattice configuration. Hence, our approach starts with two experiments to assess the feasibility of our approach.

1) Shape similarity analysis: The objective of this analysis is to verify whether shapes with similar geometric characteristics are represented close to each other in the latent space. The similarities between two point clouds given in the Cartesian

space can be measured according to the modified version of the Hausdorff distance [34], defined as follows:

$$\mathcal{H} = \frac{1}{2}\left[\sum_{i=1}^{N} \min_{\mathbf{g_i} \in G_0} d(\mathbf{g_i}, G_T)^2 + \sum_{j=1}^{N} \min_{\mathbf{g_t}_j \in G_T} d(\mathbf{g_t}_j, G_0)^2\right] \quad (1)$$

where G_0 and G_T are the initial and target point clouds respectively, N the number of points, and $\mathbf{g_i}_j$ the j-th point sampled from point cloud i. Since $\mathcal{H}(G_0, G_T) \in \mathbb{R}_0^+$ and, for a perfect match, $\mathcal{H}(G_0, G_T) = 0$, the magnitude of the function is associated to the degree of dissimilarity between shapes.

By clustering designs in the latent space, one can define each center of the clusters as a reference point cloud G_0 and measure both Hausdoff distance in the Cartesian space and Euclidean distance in the latent space, for any pair of shapes in the cluster. The expected outcome is that the larger the distance bewteen designs and cluster centers, the larger the values of the Hausdorff distance. Hence, we clustered the representations of the car shapes calculated using an autoencoder trained on a 2D-latent space into three clusters using the k-means algorithm [35], and projected the values of the Hausdorff distance to the shape in the center of cluster 0 onto the 2D-scatter of the latent representation of the designs (Fig. 4), which visually confirmed our hypothesis.

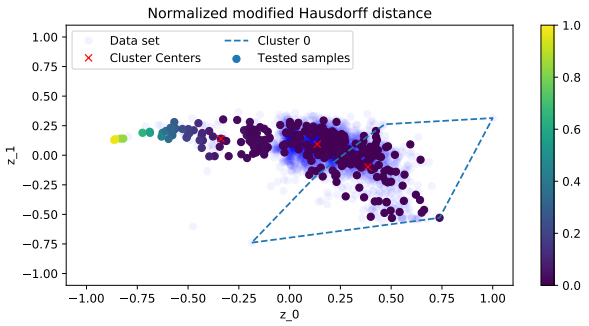

Fig. 4. Normalized value of the modified Hausdorff distance, calculated between the center and random samples of clustered designs in the latent space. The label z_i indicates the i-th latent variable.

When repeating the procedure for the 5D-latent representations, we compared the distance metrics using a scatter plot, since visualizing the high-dimensional spaces is nontrivial (Fig. 5). Similarly, the distances in the latent space correlated positively with the Hausdorff distance, indicating that the hypothesis holds for higher-dimensional latent spaces. Therefore, we concluded that we could use the Euclidean distances in the latent space as a measure of similarity between designs for the search of prototypical meshes.

2) Target shape matching optimization: Target shape matching describes the process of approximating an uknown shape from an initial geometry, based on a block-box function that measures the difference between the optimized and the target shape. So, in a second step, we matched potential prototypical meshes, assumed to correspond to the centers

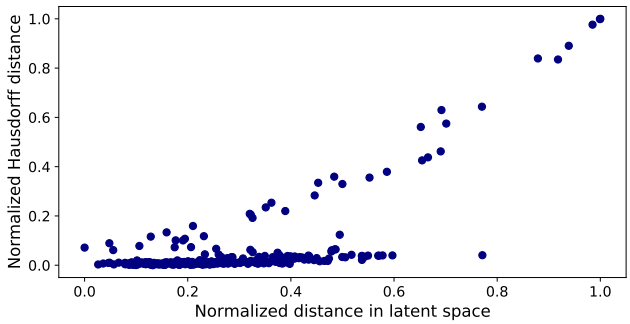

Fig. 5. Distance metrics calculated between designs and center of the cluster 0, for both point clouds and respective latent representations.

of clusters calculated from the 2D-latent representations, to autoencoder-based point clouds. Our main motivation was to assess the feasibility of our approach, but we also demonstrate that the quality of the target shape matching optimization depends on the similarity between initial and target shapes, which justifies the selection of multiple prototypical meshes.

Hence, we randomly generated ten designs from each cluster center by shifting the latent variables up to three times their standard deviation, calculated from the shapes in the cluster, and recovered the corresponding point clouds using the decoder. For matching the meshes to the point clouds, we embedded the prototypes in an FFD lattice considering theafromentioned settings, and defined the optimization problem as follows:

$$\min_{\mathbf{x}} f(\mathbf{x}) = \mathcal{H}(G_{FFD}, G_{AE}) \quad (2)$$

Where $\mathbf{x}$ is the set of FFD parameters, $G_{FFD}(\mathbf{x})$ the point cloud sampled from the deformed prototypical mesh, and G_{AE} an autoencoder-based point cloud from the generated set.

For the optimization, we used the covariance matrix adaptation evolutionary strategy (CMA-ES), which is a gradient-free method, which is suitable for small population sizes, with a high convergence ratio and low number of parameters [36]. We used a (5,20) CMA-ES with an initial step size $\sigma_0 = 0.01$, and limited the number of generations to 30.

Projecting the Hausdorff distance of the best individuals onto the latent space representations (Fig. 6), we observed a pattern similar as in the previous analyses: The shapes farther away from the cluster centers yielded worse matching results. Therefore, we concluded that in order to match a set of prototypical meshes to any shape in the data set, within a certain quality range, the number and characteristics of the meshes should be optimized, justifying our study.

D. Optimization of the mesh prototype set

In order to optimize the selection of mesh prototypes, we followed the optimization workflow in Fig. 7. The optimization starts with a random selection of k mesh prototypes from the available set of latent representations. Each mesh prototype is embedded in an FFD lattice with the fore-mentioned settings

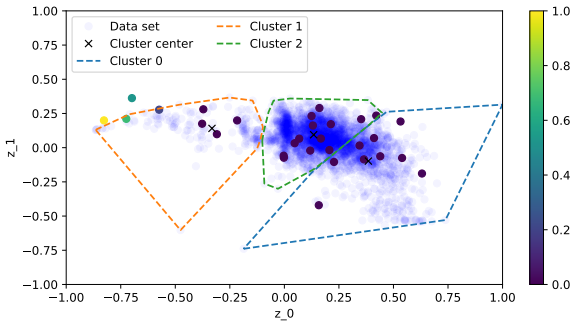

Fig. 6. Normalized Hausdorff distance of the fittest individual obtained at the last generation of each optimization run.

and randomly deformed, generating a set of shapes. The set of meshes is then sampled into point clouds and fed to a 3D point cloud autoencoder for calculating the latent representations and reconstruction losses. The deformations that yielded worse reconstructions than observed in the data set are removed from the set and the coverage is calculated from the latent representations of the remaining samples.

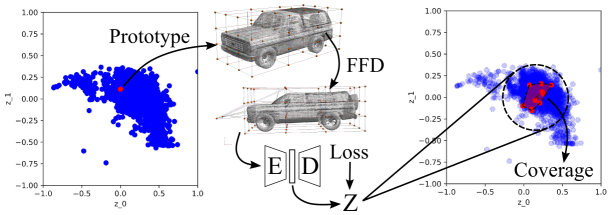

Fig. 7. Optimization workflow for maximizing the number of covered shapes in the data set.

We defined the coverage of a prototype as the hypervolume (or area for 2D-spaces) of a convex hull delimited by a set of samples in the latent space. In order to avoid intersections of hulls and generating prototypes close to infeasible regions of the latent space, we weighted the coverage by the fraction of shapes of the data set covered by each hull. Mathematically, the coverage obtained with a set of prototypes is defined as follows:

$$C_s = \sum_{p=1}^{k} \mathcal{V}_p \frac{N_{s,p}^U}{N_{s,d}} \tag{3}$$

where $\mathcal{V}_p$ is the hypervolume of the convex hull, $N_{s,d}$ is the number of shapes in the data set, and $N_{s,p}^U$ is the number of new data set samples covered by the prototype k, compared to the set obtained at iteration $k-1$.

At last, the selection of the prototypes was optimized by solving the following optimization problem

$$\min_{\mathbf{x} \in Z} f(x) = (C_s + \epsilon)^{-1} \tag{4}$$

where $\mathbf{x} = \{(z_0^1, z_1^1, \ldots, z_L^1), \ldots (z_0^k, z_1^k, \ldots, z_L^k)\}$ is the set of latent variables that represent the prototypes and ϵ is a tolerance factor with a magnitude close to 0. In our experiments, we used $\epsilon = 1E-06$

For this optimization we also applied the CMA-ES algorithm, with same strategy as in the previous target shape matching problem, however for a maximum of 50 generations instead of 30. To analyze the influence of the number of prototypes, dimensionality of the latent space and initial solution, we considered cases with one, three, five, seven and nine prototypes, for 2- and 5-dimensional latent spaces, and twenty different initializations.

IV. Experiments and Discussion

We started our analyses by verifying the convergence behavior of the optimizations (Fig. 8). We observed that, on average, the objective function was minimized in all cases, except for the optimization with a single prototype in the 5D-latent space. Furthermore, increasing the dimensionality of the latent space led to a higher variance of results, which reduced with an increasing the number of prototypes (Table II).

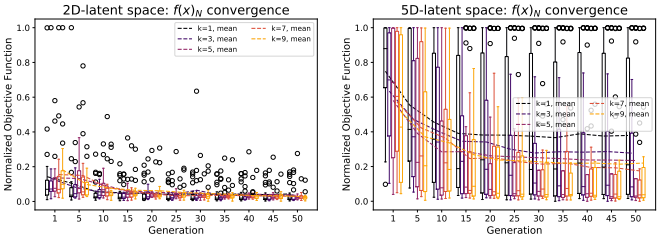

Fig. 8. Normalized objective function calculated for the best individuals for every five generations in the optimizations with 2D (left) and 5D (right) latent spaces

TABLE II
Average and standard deviation of the normalized objective function and number of shapes covered by the prototypes.

L	k	$f(X*)_N$	Covered Shapes
2	1	(4.30 ± 5.29)E-02	1826 ± 680
	3	(3.71 ± 5.11)E-02	2506 ± 674
	5	(3.51 ± 5.61)E-02	2775 ± 462
	7	(3.12 ± 2.52)E-02	2851 ± 423
	9	(2.65 ± 2.52)E-02	2970 ± 352
4	1	(3.79 ± 3.97)E-01	216 ± 186
	3	(2.77 ± 3.56)E-01	328 ± 248
	5	(2.35 ± 3.80)E-01	451 ± 275
	7	(1.75 ± 2.96)E-01	480 ± 273
	9	(2.20 ± 3.48)E-01	425 ± 282

Our interpretation of the results is in line with the analysis of the network features presented in [18]. First, the encoder learns preferably how to differentiate the distributions of points in the shapes of the data set, such that increasing the dimensionality of the latent space allows the autoencoder to learn more shape details. Second, the latent variables represent the combination of regions that the point cloud occupies in the input space, while the decoder learns how to map latent activations back into Euclidean space such as to reconstruct the point clouds.

Randomly deforming mesh prototypes, possibly yields point clouds with point distribution that are uncommon in the data

set. Since the 5D latent space describes finer changes in the occupancy of the input space, the decoder failed to reconstruct some of the generated shapes during the optimizations, which limited the size of the hull. A similar conclusion is drawn from the analysis of the covered shapes in the data set (Fig. 9). The optimizations with the 2D latent representations stagnated faster at a higher number of covered shapes than with the 5D latent space, also with respect to the number of prototypes. In the former, the stagnation is explained by the overlapping of hulls (Fig. 10), while in the latter, in addition to the overlapping, the size of the hulls was restricted by the reconstruction quality, limiting the number of covered shapes to a lower value.

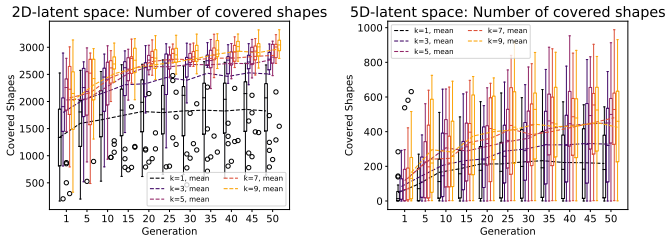

Fig. 9. Number of data set shapes covered by the set of prototypical meshes optimized in the 2D (left) and 5D (right) latent spaces.

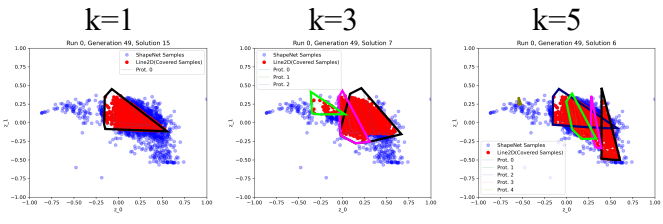

Fig. 10. Projection of the hulls obtained for different sets of prototypes onto the representation of the 2D-latent space.

Finally, we analyzed the trade-off between the costs of generating the mesh prototypes and number of covered shapes with a Pareto front-style diagram (Fig. 11). As observed, the optimization with the 2D latent representations performed better than with the 5D latent space, which is in line with our previous observations.

In addition to the discussed causes of such behavior, the characteristics of the lattice also explain the differences in the performance. Since the 5D latent space can describe finer changes in the point clouds, it also requires a denser lattice and potentially more deformation parameters to enable lower-scale deformations. As an example, we clustered the latent representations using the k-means algorithm with three clusters and sampled the three nearest shapes to the center of a cluster (Fig. 12). As observerd, the transition between shapes represented in the 5D latent space are more subtle than obtained with the 2D representations and challenging to be obtained with the proposed lattice configuration. Therefore, balancing the costs and coverage with higher-dimensional latent spaces becomes even more difficult, since it does not

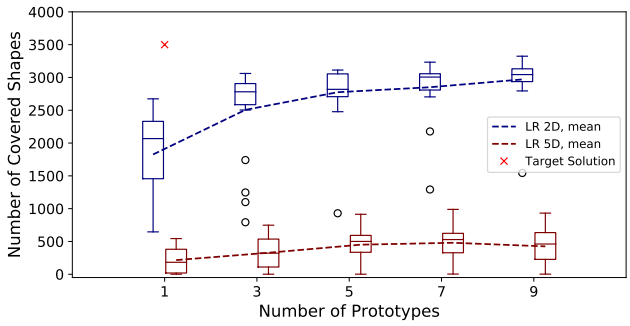

Fig. 11. Number of designs from the autoencoder training set covered by each optimized set of prototypes, considering different dimensionalities of the latent space

only require more prototypes, but the effort to determine the FFD settings also increases.

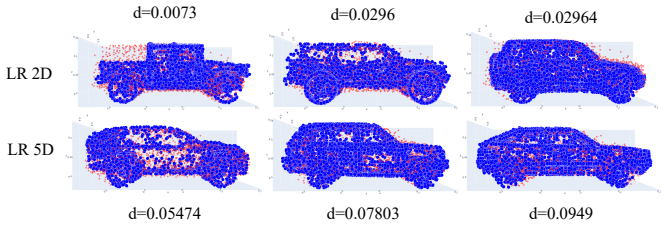

Fig. 12. Reconstruction of the nearest shapes to a cluster center, sampled for both representations. The variable d indicates the Euclidean distance to the cluster center.

V. Conclusions and Outlook

In the present paper we propose an approach for finding optimal sets of prototypical meshes that can be matched using FFD to autoencoder-based point clouds. We based our approach on a feasibility analysis, where we showed the correspondences between shape similarities, distances in the latent space and target shape matching performance. We optimized the selection of prototypes by searching candidate solutions in the latent space, which is lower-dimensional and thus more efficient than using conventional geometric representations.

We analyzed the trade-off between the number of prototypes and coverage as in a multi-objective optimization problem. We observed that the coverage stagnated quickly with an increasing number of prototypes. However, increasing the dimensionality of the latent space allowed the autoencoder to learn more shape details, which requires more refined FFD representations to match neighbor samples in the latent space. Hence, the costs of implementing multiple prototypes does not only vary with respect to the number of shapes, but also depends on the granularity with which the autoencoder can modify shapes.

Regarding the limitations of our approach, we favored computational effort over accuracy when searching for prototypes, since we based the coverage on the latent representations of deformed meshes. Furthermore, we neither explored variations

in the FFD setup, nor in the characteristics of the data set, which was constrained to a single class. Finally, we explored the relation between FFD and latent representations, showing that, in a shape optimization problem, we can combine both in order to profit from the diversity of shapes represented in the low-dimensional latent space and the smoothness in shape modifications provided by the free form deformation.

REFERENCES

[1] T. Friedrich, N. Aulig, and S. Menzel, "On the potential and challenges of neural style transfer for three-dimensional shape data," in *International Conference on Engineering Optimization*. Springer International Publishing, 2019, pp. 581–592.

[2] M. M. Bronstein, J. Bruna, Y. LeCun, A. Szlam, and P. Vandergheynst, "Geometric deep learning: going beyond euclidean data," *IEEE Signal Processing Magazine*, vol. July, pp. 18–42, 2017.

[3] N. Umetani, "Exploring generative 3D shapes using autoencoder networks," in *SIGGRAPH Asia Technical Briefs 2017*. ACM SIGGRAPH Asia, 2017, pp. 1–4.

[4] D. Bau, B. Zhou, A. Khosla, A. Oliva, and A. Torralba, "Network dissection: Quantifying interpretability of deep visual representations," in *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*. IEEE Computer Society, 2017, pp. 3319–3327. [Online]. Available: https://doi.org/10.1109/CVPR.2017.354

[5] M. Berger, A. Tagliasacchi, L. M. Seversky, P. Alliez, J. a. Levine, A. Sharf, C. T. Silva, A. Tagliasacchi, L. M. Seversky, C. T. Silva, J. a. Levine, and A. Sharf, "State of the Art in Surface Reconstruction from Point Clouds," *Proceedings of the Eurographics 2014, Eurographics STARs*, vol. 1, no. 1, pp. 161–185, 2014. [Online]. Available: http://lgg.epfl.ch/reconstar

[6] S. Menzel and B. Sendhoff, *Representing the Change - Free Form Deformation for Evolutionary Design Optimization*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 63–86. [Online]. Available: https://doi.org/10.1007/978-3-540-75771-9_4

[7] A. X. Chang, T. A. Funkhouser, L. J. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su, J. Xiao, L. Yi, and F. Yu, "ShapeNet: An information-rich 3D model repository," *arXiv preprint arXiv:1512.03012v1 [cs.GR]*, 2015.

[8] T. Rios, B. Sendhoff, S. Menzel, T. Bäck, and B. 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)*, 2019, pp. 791–798.

[9] A. Ioannidou, E. Chatzilari, S. Nikolopoulos, and I. Kompatsiaris, "Deep learning advances in computer vision with 3D data," *ACM Computing Surveys*, vol. 50, no. 2, pp. 1–38, 2017.

[10] Z. Cai, J. Han, L. Liu, and L. Shao, "RGB-D datasets using microsoft kinect or similar sensors: a survey," *Multimedia Tools and Applications*, vol. 76, no. 3, pp. 4313–4355, 2017.

[11] T. Yuan, X. Peng, and D. Zhang, "Direct rapid prototyping from point cloud data without surface reconstruction," *Computer-Aided Design and Applications*, vol. 15, no. 3, pp. 390–398, 2018. [Online]. Available: https://doi.org/10.1080/16864360.2017.1397889

[12] H. T. Park, M. H. Chang, and S. C. Park, "A slicing algorithm of point cloud for rapid prototyping," in *Proceedings of the 2007 Summer Computer Simulation Conference*, ser. SCSC '07. San Diego, CA, USA: Society for Computer Simulation International, 2007.

[13] C. R. Qi, L. Yi, H. Su, and L. 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*, ser. NIPS'17. Red Hook, NY, USA: Curran Associates Inc., 2017, p. 5105–5114.

[14] Y. Guo, H. Wang, Q. Hu, H. Liu, L. Liu, and M. Bennamoun, "Deep Learning for 3D Point Clouds: A Survey," *arXiv preprint arXiv:1912.12033 [cs.CV]*, 2019.

[15] P. Achlioptas, O. Diamanti, I. Mitliagkas, and L. Guibas, "Learning representations and generative models for 3D point clouds," in *Proceedings of the 35th International Conference on Machine Learning (ICML)*, vol. 80. Stockholmsmässan, Stockholm Sweden: PMLR, 2018, pp. 40–49.

[16] Y. Yang, C. Feng, Y. Shen, and D. Tian, "FoldingNet: Point cloud auto-encoder via deep grid deformation," *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 206–215, 2018.

[17] M. Gadelha, R. Wang, and S. Maji, "Multiresolution tree networks for 3D point cloud processing," in *Computer Vision – ECCV 2018*. Springer International Publishing, 2018, pp. 105–122.

[18] T. Rios, B. van Stein, S. Menzel, T. Bäck, B. Sendhoff, and P. Wollstadt, "Feature visualization for 3d point cloud autoencoders," in *International Joint Conference on Neural Networks (IJCNN) 2020*, 2020.

[19] H. Fan, H. Su, and L. Guibas, "A point set generation network for 3D object reconstruction from a single image," in *30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, vol. 2017-Januar, 2017, pp. 2463–2471.

[20] Y. Rubner, C. Tomasi, and L. J. Guibas, "The earth mover's distance as a metric for image retrieval," *Int. J. Comput. Vision*, vol. 40, no. 2, p. 99–121, Nov. 2000. [Online]. Available: https://doi.org/10.1023/A:1026543900054

[21] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle, "Surface reconstruction from unorganized points," *Computer Graphics (ACM)*, vol. 26, no. 2, pp. 71–78, 1992.

[22] A. Hornung and L. Kobbelt, "Robust Reconstruction of Watertight 3D Models from Non-uniformly Sampled Point Clouds Without Normal Information," in *Symposium on Geometry Processing*, A. Sheffer and K. Polthier, Eds. The Eurographics Association, 2006.

[23] F. Bernardini, J. Mittleman, H. Rushmeier, C. Silva, and G. Taubin, "The ball-pivoting algorithm for surface reconstruction," *IEEE Transactions on Visualization and Computer Graphics*, vol. 5, no. 4, pp. 349–359, 1999.

[24] N. Sharp and M. Ovsjanikov, "PointTriNet: Learned Triangulation of 3D Point Sets," 2020. [Online]. Available: http://arxiv.org/abs/2005.02138

[25] R. Hanocka, G. Metzer, R. Giryes, and D. Cohen-Or, "Point2Mesh: A Self-Prior for Deformable Meshes," vol. 39, no. 4, 2020. [Online]. Available: http://arxiv.org/abs/2005.11084{\%}0Ahttp://dx.doi.org/10.1145/3386569.3392415

[26] S. Menzel, M. Olhofer, and B. Sendhoff, "Application of free form deformation techniques in evolutionary design optimisation," in *6th World Congress on Structural and Multidisciplinary Optimization (WCSMO6)*, J. Herskovits, S. Mazorche, and A. Canelas, Eds. Rio de Janeiro: COPPE Publication, 2005.

[27] D. Sieger, S. Menzel, and M. Botsch, *On Shape Deformation Techniques for Simulation-Based Design Optimization*, 01 2015, vol. 5, pp. 281–303.

[28] R. Duvigneau, "Adaptive Parameterization using Free-Form Deformation for Aerodynamic Shape Optimization," Research Report, 2006. [Online]. Available: https://hal.inria.fr/inria-00085058

[29] M. Olhofer, T. Bihrer, S. Menzel, M. Fischer, and B. Sendhoff, "Evolutionary optimisation of an exhaust flow element with free form deformation," in *Simulation for Innovative Design, Proceedings of the 4th EASC - 2009 European Automotive Simulation Conference*, K. Seibert and M. Jirka, Eds. ANSYS Inc., 2009, pp. 163–174.

[30] T. W. Sederberg and S. R. Parry, "Free-form deformation of solid geometric models," in *Proceedings of the 13th Annual Conference on Computer Graphics and Interactive Techniques*, ser. SIGGRAPH '86. New York, NY, USA: ACM, 1986, pp. 151–160. [Online]. Available: http://doi.acm.org/10.1145/15922.15903

[31] T. Rios, P. Wollstadt, B. v. Stein, T. Bäck, Z. Xu, B. Sendhoff, and S. Menzel, "Scalability of learning tasks on 3d cae models using point cloud autoencoders," in *2019 IEEE Symposium Series on Computational Intelligence (SSCI)*, 2019, pp. 1367–1374.

[32] Q. Chen and H. Prautzsch, "General triangular midpoint subdivision," *Computer Aided Geometric Design*, vol. 31, no. 7, pp. 475 – 485, 2014, recent Trends in Theoretical and Applied Geometry. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167839614000600

[33] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

[34] P. Zhang, X. Yao, L. Jia, B. Sendhoff, and T. Schnier, "Target shape design optimization by evolving splines," 10 2007, pp. 2009 – 2016.

[35] D. Arthur and S. Vassilvitskii, "K-means++: The advantages of careful seeding," in *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, ser. SODA '07. USA: Society for Industrial and Applied Mathematics, 2007, p. 1027–1035.

[36] T. Bäck, F. Hoffmeister, and H.-P. Schwefel, "A survey of evolution strategies," in *Proceedings of the Fourth International Conference on Genetic Algorithms*. Morgan Kaufmann, 1991, pp. 2–9.