Learning-based representations of high-dimensional CAE models for automotive design optimization

de Jesus de Araujo Rios, T.

5 Geometric Interpretation of the PC-AE Latent Features

5.1 Introduction

In the last chapter (Section 4.6), PC-AE-Rios is utilized as black-box shape-generative model in an evolutionary design optimization problem. The experiments show that the proposed autoencoder only generates realistic shapes from samples within the learned latent space, as expected from machine learning models. However, PC-AE-Rios also generates rough approximations of shapes fundamentally different than the samples in the training data, which indicates that the latent variables potentially map a notion of the distribution of points in the input space rather than geometric features with an intuitive semantic interpretation.

The motivation for this chapter is to find a geometric interpretation for the latent features learned by PC-AE-Rios. The understanding on how geometric features are mapped to the latent space allows us to better assess the limitations and advantages of PC-AE-Rios with respect to other data-driven techniques, which is also a known challenge for deep neural networks (DNNs) [124, 100, 195]. Hence, the methods and experiments in this chapter address the proposed research questions *RQ3* and *RQ4*.

The remainder of this chapter is outlined as follows: In Section 5.2, a novel feature visualization method for 3D point cloud autoencoders [129] is utilized to project the latent features onto 3D point clouds and identify the structures mapped by the latent variables. In Section 5.3, the latent variables are analyzed based on sensitivity analyses and shape-generative experiments, where PC-AE-Rios is compared to PCA-based design representations. Finally, in Section 5.4, the chapter is concluded by summarizing the main findings and answering the aforementioned research questions.

5.2 Feature Visualization for 3D Point Cloud Autoencoders

5.2.1 Background

The analysis of features learned by deep neural networks (DNNs) aims at revealing data patterns that justify the results and a particular behavior of a trained network. The currently available feature visualization techniques are based on input modification methods, which optimize the input data to generate a target set of activations, and gradient-based analyses, which indicate the sensitivity of particular activations to changes in the input data. Yet, these methods target convolutional neural networks (CNNs) for image processing and extensions of these methods to geometric deep learning architectures are currently sparse in the literature.

The work by Fergus & Zeiler [188] pioneered the study on feature visualization for convolutional neural networks (CNNs). The authors attach deconvolutional neural networks (*deconvnets*) [189] to deep layers of CNNs to map the activations back onto the input pixel space. The output of deconvnets indicates the contribution of each pixel to the pattern of activations obtained at the layer of interest, which reveals image features that justify the behavior of the network. Since then, different methods have been proposed, which are clustered in a recent survey into three classes [66]: Input modification, deconvolutional methods and reconstruction methods.

Input modification techniques quantify the changes in the activation maps by locally modifying the input data. The underlying assumption is that the modified pixels that yield the highest changes in the activations at the layer of interest are the most relevant to describe the feature encoded by the corresponding layer. In [188], the authors utilize the occlusion technique, where the input image is modified by iteratively occluding different regions with a small patch of gray pixels. The authors in [199] extend the approach by utilizing randomly colored pixel patches, which avoid introducing artificial features in the input images. Although effective for image processing, the method is expected to scale poorly to point-based networks due to the dimensionality of the data and lack of connectivity between points.

Deconvolutional methods follow the reverse path of input modification methods. Starting from an activation of interest, the methods back-propagate the contribution of the network neurons to the input domain, which reveals the contribution of the pixels to the obtained activation. The discussed work by Zeiler & Fergus [188] is an example of a deconvolutional method. In [146], for an image classification network, the authors utilize the derivative of the class score with respect to the input pixels as a measure of importance of the pixels for the classification result. Further work build upon the same method to improve the performance and adapt the approach to different architectures [150, 8]. For PC-AE-Rios, the correspondences between activations and input points are known due to the nature of 1D convolutions. Thus, the projection of the activations is straightforward and requires simpler approaches than deconvolutional techniques.

Reconstruction methods iteratively modify the input data to obtain specific patterns of co-activations at a layer of interest as in an optimization problem [165, 98]. Conventionally, the objective of the optimizations is to maximize a particular output of the network, *e.g.*, the probability of an input image x to represent an object in a class y. However, the objective functions are usually non-convex, the lack of regularization leads to high-frequency noise in the optimized images, and both, maximization and minimization of features, can be utilized to describe the behavior of DNNs [98, 113]. In [112], the authors propose a multifaceted feature visualization algorithm, which accounts for the response of each artificial neuron to multiple stimuli. By feeding a batch of images to a trained CNN, the authors compute the corresponding feature maps obtained at a specific layer and compress them into a 2D space by applying t-SNE [162]. Then, the authors cluster the 2D representations of the feature maps using the k-means algorithm into k clusters, where each cluster reveals a "facet" of the features learned by the CNN. The authors claim that the proposed approach improves the understanding of CNN features by generating optimized images with higher quality and providing different insights on the features encoded by the neurons due to the multifaceted visualization.

Despite the promising approaches for CNNs, the literature lacks specific

methods for geometric deep learning architectures, particularly for point-based networks. A straightforward extension of the currently available method is challenging, but some of the core ideas are still applicable, such as the identification of co-activation patterns by utilizing clustering techniques. Recently, a novel feature visualization technique for 3D point cloud autoencoders has been proposed [129], which builds upon the properties of 1D convolutional operators, as detailed in the following sections.

5.2.2 Feature Visualization

The principle of the feature visualization technique [129] is to project the activations of the 1D convolutional layer that is adjacent to the max-pooling layer back onto the input point clouds as color maps. Since the latent variables correspond to the maximum values of the visualized features, the regions of the point clouds that yield higher activation values indicate the geometric characteristics that are mapped by the visualized network feature.

In this research, the f-th feature $\mathcal{F}^{f,j}$ of the j-th 1D convolutional layer is defined as the set of activations obtained for the f-th neuron of the 1D convolutional operator (Eq. 4.1) for all points of an input 3D point cloud. In CNNs with 2D convolutions, the feature maps are represented by 2D matrices, where each matrix represents a feature detected by the network. In the case of 1D convolutions, a feature f is represented as a vector, which corresponds to the the f-th column of the output matrix $\mathcal{F}^{j}$ of the j-th 1D convolutional layer. Furthermore, since the points are processed individually, each point has a corresponding row of activations in $\mathcal{F}^{f,j}$, which is computed independently of the other points. Hence, the projection of the activations onto the input point cloud is a straightforward process, since each point has a direct correspondence to a set of activations in the 1D convolutional layers (Fig. 5.1).

Compared to extending CNN-based feature visualization techniques, the proposed method requires lower implementation effort while still provides significant insight on the information learned by the network. Furthermore, as 1D convolutions operate as MLPs shared across all points, methods based on input modification potentially cluster the points around the local maxima

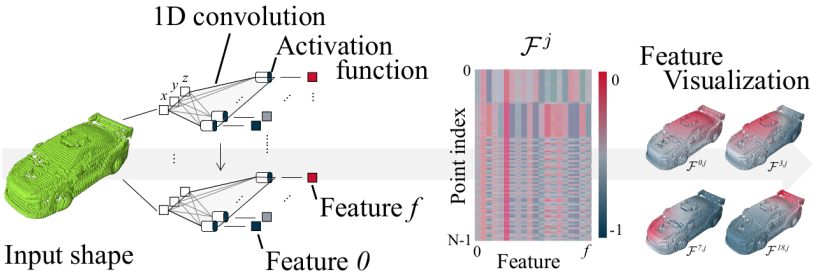

Figure 5.1 – Schematic of the workflow for visualizing the features learned by PC-AE-Rios based on an input car shape.

of each feature, which generates geometric structures with less intuitive interpretation. However, the proposed feature visualization technique is limited to networks that process input data through 1D convolutions and before any layer with global operators, *e.g.*, max-pooling.

5.2.3 Experiments Setup

In this set of experiments a clustering-based workflow is utilized to identify the geometric properties mapped by the latent variables of PC-AE-Rios (Fig. 5.2). The utilized method builds upon the multi-faceted approach proposed in [112], which utilizes clustering methods to identify patterns of co-activations in the learned data.

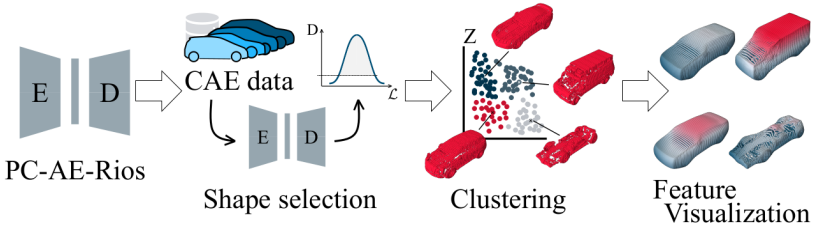

Figure 5.2 – Proposed workflow for visualizing and interpreting the features learned by PC-AE-Rios on a data set of CAE models.

In the first step, PC-AE-Rios is trained on a data set of car shapes, which are sampled from the ShapeNetCore repository into 3D point clouds with

24578 points using the proposed shrink-wrapping method (Section 4.5). The hyperparameters of the architecture are set to the baseline values (Table 4.1), apart from the size of the latent space, which is set to L_z={2,10,20} to ease the visualization of the complete set of features. The utilized training algorithm and hardware setup is the same as described in Section 4.1, and the utilized loss function is the mean-squared distance (MSD).

In the second step, the data set is filtered based on the shape reconstruction losses (Chamfer Distance, CD). For that purpose, a kernel density estimator model (KDE) is utilized to determine the most frequent loss values observed in the data set [119]. The KDE is set with a Gaussian kernel and a bandwidth defined by the mean span between CD values (Eq. 5.1), where N_{shapes} is the number of shapes in the data set. The trained KDE is utilized to evaluate the density of all data samples and the shapes with score above 20% of the maximum value are selected for the next steps of the analysis.

$$h_{KDE} = \frac{[\max(CD_{dataset}) - \min(CD_{dataset})]}{N_{shapes}} \,. \tag{5.1}$$

In the third step, the latent representations of the selected samples are clustered using the hierarchical density-based spatial clustering of applications with noise (HDBSCAN) algorithm [20]. The Euclidean distance is utilized to evaluate the similarity between samples and a minimum cluster size is set to 7, which is defined based on the available metadata of the repository. Further algorithm parameters are adjusted iteratively based on the silhouette score of the data samples [132] and latent space dimensionality L_z.

In the last step, the mean point cloud of each cluster is calculated by averaging the position of corresponding points for all shapes in each cluster. Then, the features of the mean shapes are calculated using PC-AE-Rios and projected onto the corresponding point clouds for visualization. To interpret the learned features, three experiments are performed: Comparison between the activations and point-wise differences between mean cluster shapes, invariance of the features to shifts of the input shapes, and comparison of the activations to the regions of the input space most densely occupied by the training data.

5.2.4 Results and Discussion

Selection of shapes: By filtering the data set, most of the removed samples are associated to higher loss values (Fig. 5.3). Also, in line with the results obtained in the previous chapter, the increase of the latent space dimensionality improved the reconstruction quality, and the CD values obtained with $L_z = 20$ are within similar range than obtained with $L_z = 128$. Thus, the 20D model provides a fair approximation of the reconstruction quality obtained with a 128D latent space and allows us to visualize the complete set of learned features with less effort. The selected subsets preserved 96.3%, 95.5% and 95.7% of the original data for training PC-AE-Rios with 2-, 10-, and 20D latent spaces, respectively.

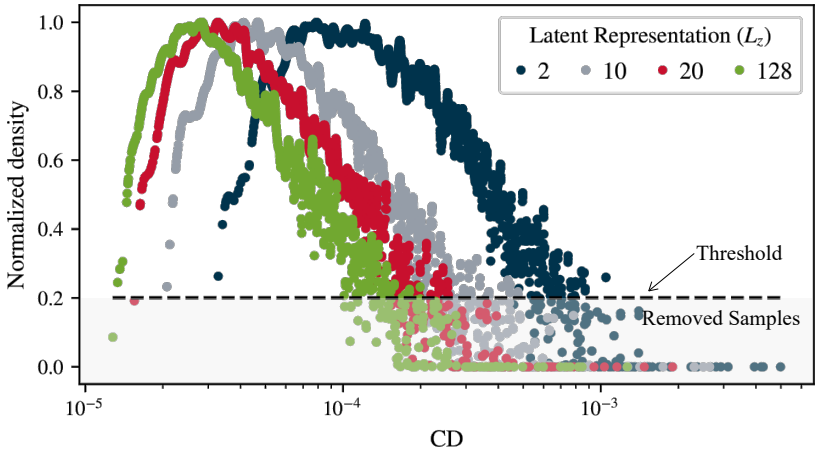

Figure 5.3 – Normalized density values obtained with the KDE models utilized to select the samples based on the reconstruction quality.

Clustering: The clustering algorithm identified six, clusters based on the 2D representations and 8 based on the 10- and 20D representations of the selected samples. For most clusters, the mean silhouette scores are greater than 0.5, which indicates that the samples within each cluster have high similarity (Fig. 5.4). The clusters 3, 6 and 7 obtained for the 2-, 10- and 20D representations, respectively, are the largest clusters ($\approx$ 2500 shapes) and present the highest deviations of the silhouette score. The imbalance between clusters is in line

with the information available in the metadata of the repository, where most of the samples are assigned to the same class.

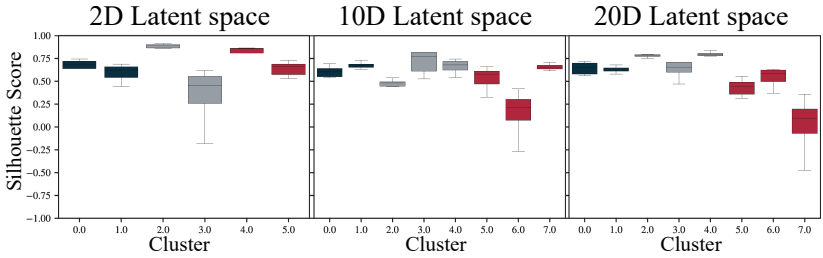

Figure 5.4 – Silhouette scores calculated for the obtained clusters and according to the dimensionality L_z of the latent space.

In a second verification of the clustering results, the shapes are compared to the mean 3D point cloud of their respective clusters. For that, the Euclidean distances between corresponding points are calculated, averaged, and projected onto the 3D mean point clouds for visualization (Fig. 5.5). The obtained mean distance values are higher and cover a larger region of the shapes that represent the largest clusters (3, 6 and 7 for 2-, 10- and 20D spaces, respectively), which indicates that these clusters comprise a more diverse set of shapes as indicated by the silhouette score.

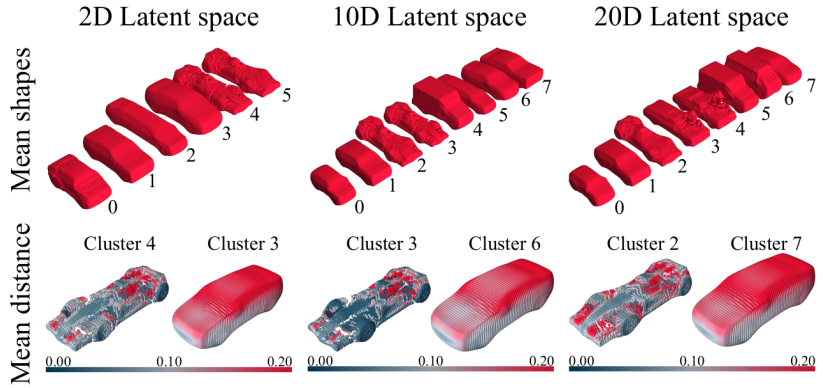

Figure 5.5 – Mean 3D point clouds obtained for each cluster and mean point-wise distance projected onto two shapes for comparing the similarity between shapes within a cluster.

Feature visualization: To ease the interpretation of the network features, the 2D latent space is analyzed first. By feeding the mean shapes of each cluster to PC-AE-Rios, the obtained color maps highlight irregular patches in the shapes, which apparently occupy similar regions in the input space (Fig. 5.6). The irregularity of the highlighted patches is also expected due to the point-wise nature of the 1D convolutions, which neglect large-scale information when processing the input data. Furthermore, PC-AE-Rios learns the representations by optimizing the weights according to a loss function (MSD), which neglects any structural segmentation of the shapes.

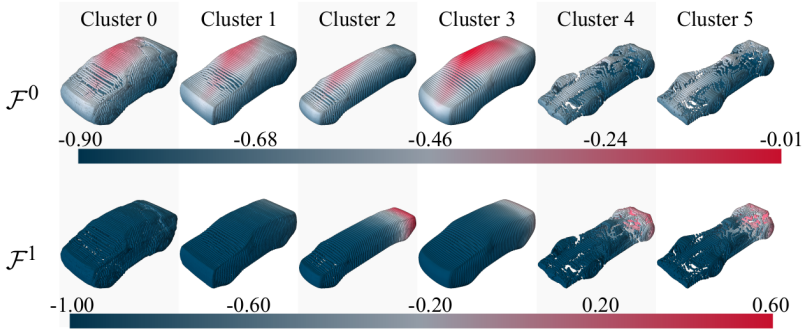

Figure 5.6 – Feature visualization method applied to the mean shapes of the clusters obtained using PC-AE-Rios trained with a 2D latent space. The scale indicates the activation values of the visualized features.

An interpretation of the visualized features is that they indicate the patches of the shapes that optimally describe the differences between data samples according to the utilized loss function and available degrees of freedom. To verify the hypothesis, the obtained features are visually compared to the point-wise distances between the mean cluster shapes (Fig. 5.7). In most shape combinations, the projected distances highlight regions on the top and rear of the car shapes, similarly to the features $\mathcal{F}^0$ and $\mathcal{F}^1$, respectively.

Analogous patterns are observed when extending the analysis to PC-AE-Rios trained with 10- and 20D latent spaces. To quantify the similarity between patterns, for the 10D representations, the squared Spearman correlation factor

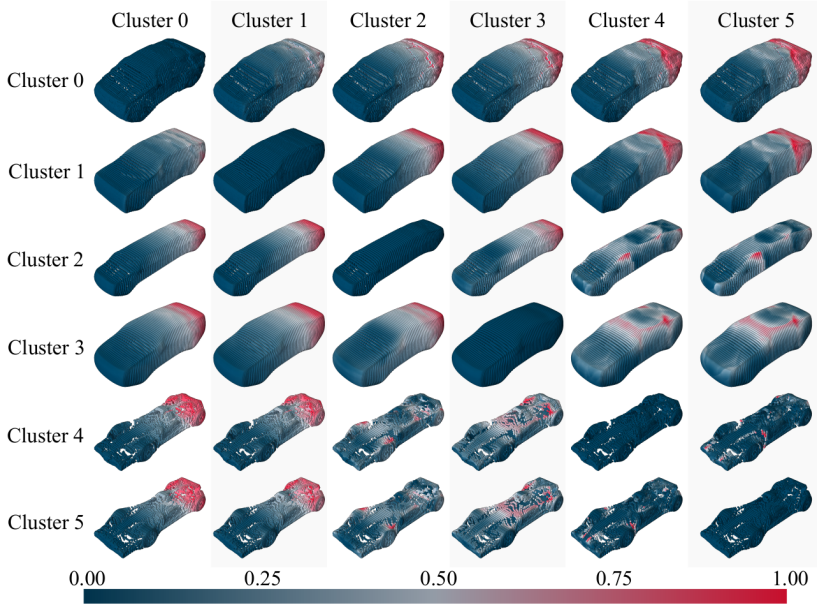

Figure 5.7 – Visualization of the normalized point-wise distances obtained for all pairs of mean cluster shapes (10D latent representations).

is calculated between the activations and point-wise distances for all pairs of shapes (Fig. 5.8). The results show that most of the features have a monotonic relation to the distance between points for at least a pair of shapes. Therefore, it is reasonable to infer that PC-AE-Rios learns preferably the geometric characteristics of 3D point clouds that distinguish the samples in the data set, which is generally expected of machine learning models.

The obtained feature visualizations also suggests that the latent variables map fixed regions of the input space rather than complex geometric features, *e.g.*, side mirrors. Hence, a shift-invariance experiment utilizing PC-AE-Rios trained with a 10D latent space is performed to verify the hypothesis of shift-invariance. In this experiment, an input shape is recursively shifted by $d_s \in [-0.75, 0.75]$ in steps of 0.25 along each of the Cartesian axes and fed to PC-AE-Rios for calculating the corresponding features and shape reconstruction.

The visualization of the obtained features for different shifted positions

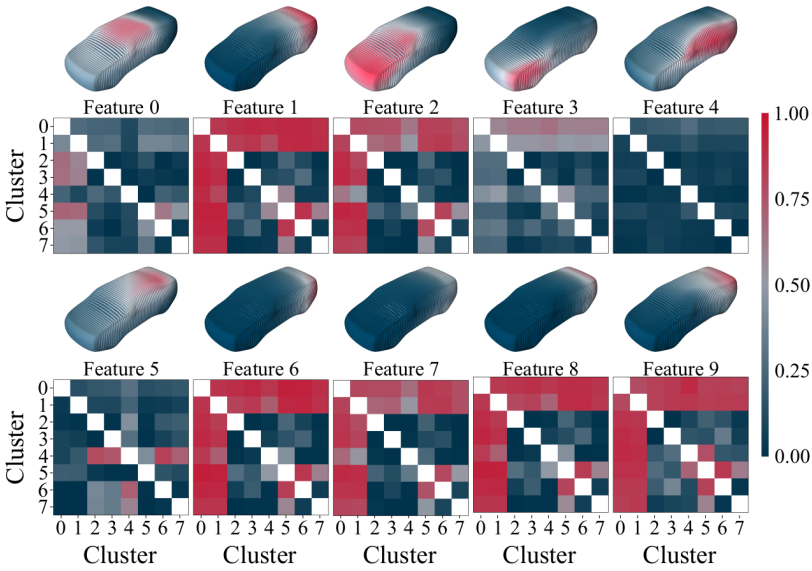

Figure 5.8 – Squared Spearman correlation factor calculated between network features and point-wise distances between mean cluster shapes. The normalized features are also visualized for reference.

shows that the highlighted regions shift according to the input shape (Fig. 5.9). Hence, the network features map the occupancy of fixed irregular 3D patches in the input space, and are sensitive to shift, rotation, and scale of the input data. This is also confirmed by observing the reconstructed designs, which mismatch the input shape and change according to the shift of the input point cloud.

In the last experiment, the network features are compared to the density with which the point clouds of the training set occupy different regions of the input space. The density is defined by a KDE model that learns the distribution of the points observed in the utilized training data. The KDE model is set with a Gaussian kernel and with a bandwidth of 10% of the standard deviation of the coordinates for each Cartesian axis. The density is evaluated on a uniform lattice with 64000 points (40 planes in each Cartesian direction) defined in $[0.1, 0.9]^3$. For comparison, the same lattice is fed to PC-AE-Rios and the

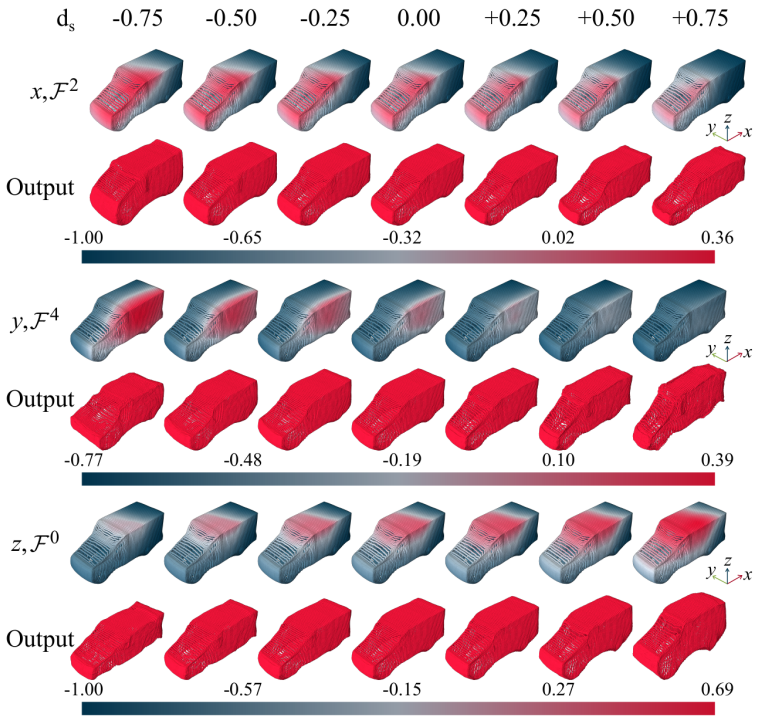

Figure 5.9 – Visualization of three features computed for a shape with different shifted positions and corresponding shape reconstructions.

corresponding activations are computed according to each of the trained PC-AEs.

To visualize the results (Fig. 5.10), the transparency of the points in the lattice varies linearly with the activation values and only the points that yield a density higher than obtained in empty regions of the input space, *e.g.*, $(0.9, 0.9, 0.9)$, are shown (green shape). The regions mapped by the activations, for all latent representation dimensionalities, surround the shape identified by the KDE model, which represents the most common regions observed in the data set. Hence, in line with the previous results, the experiment shows that the features learned by PC-AE-Rios highlight design differences observed in the data set, and the features become more refined as the dimensionality of the latent space increases.

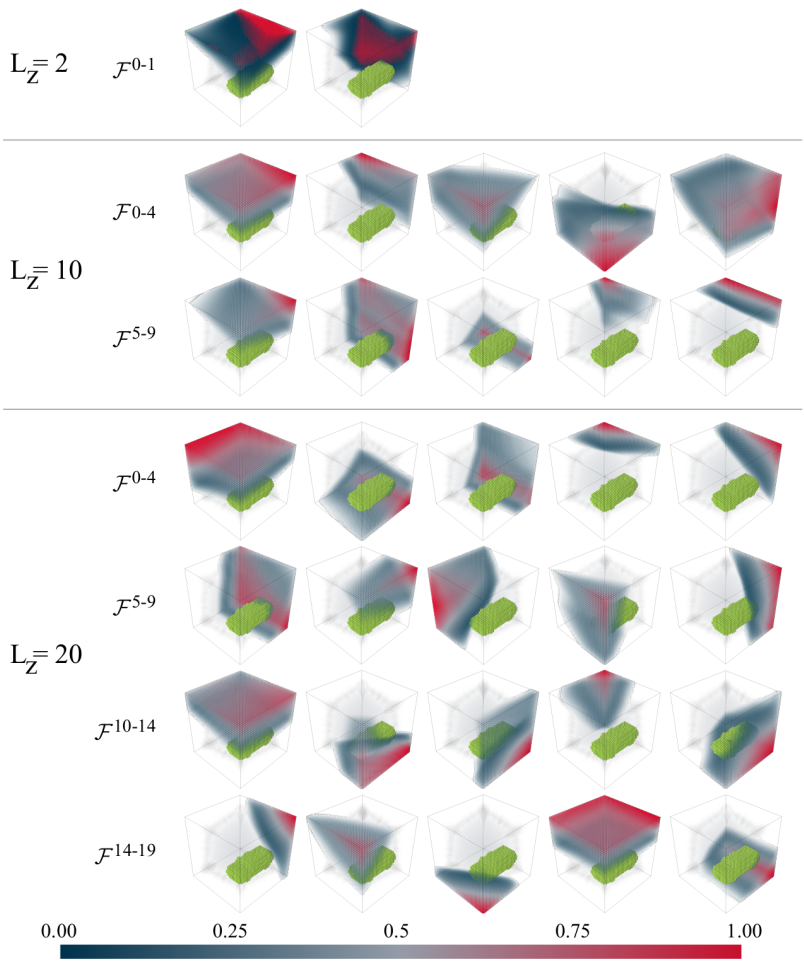

Figure 5.10 – Comparison between the high-density regions identified by the KDE models (in green) and the normalized features mapped by PC-AE-Rios for different latent representation sizes (color maps).

5.2.5 Summary

In this section, a novel feature visualization technique is utilized to interpret the latent features learned by PC-AE-Rios. The method exploits the properties of 1D convolutional layers to project the activations of deep encoder layers

as color maps onto input shapes. Differently from prior work on feature visualization for CNNs, the method is tailored for efficiently operating on point-based autoencoders with sequential 1D convolutional layers.

Based on a series of experiments, the features learned by PC-AE-Rios are interpreted as the occupancy of irregular 3D patches in the space $S = \{x \in \mathbb{R}^3 \mid x \in [0,1]^3\}$ by the corresponding input shape. These patches are fixed with respect to the input space. Thus, the latent features are variant with respect to the position, orientation, and scale of the input shape. Furthermore, the position of the patches is learned in a way to optimally describe design variations observed in the training data according to the loss function and limited by the available degrees of freedom, *i.e.*, dimensionality of the latent space. Thus, an increase in the dimensionality of the latent space enables PC-AE-Rios to learn the occupancy of more refined patches and, consequently, to reconstruct finer shape details.

The characteristics highlighted in these three experiments are in line with the mathematical formulation of 1D convolutions and proposed training conditions. As 1D convolutions process points individually, it is expected that, despite the depth of the architecture, the encoded features learn exclusively local and high level features. Furthermore, all training samples have the same alignment, position and scale. Hence, even if 1D convolutions were able to capture shift and rotation invariance, the proposed training conditions do not enforce the features to learn invariant features. Therefore, these properties of the latent features of PC-AE-Rios are expected to hold or other point-based networks with similar architecture, *i.e.*, networks that process input data through 1D convolutions and without intermediate operators, that are trained under similar conditions.

5.3 Exploiting Local Geometric Features Using PC-AEs

In the previous section, a feature visualization method is utilized to interpret the features learned by PC-AE-Rios. Yet, since the method is applicable only to similar architectures, an alternative approach is required for comparing the

autoencoder to other data-driven design representation techniques. Hence, in this section, PC-AE-Rios is analyzed from a shape-generative perspective, which allows us to compare the performance of the autoencoder to PCA-based representations.

5.3.1 Background

In evolutionary optimization, reducing the dimensionality of the search space often improves the computational efficiency of the optimization by simplifying the problem landscape [35]. Therefore, dimensionality reduction techniques (DRTs) are powerful techniques to automatically generate efficient design spaces for optimization. Furthermore, if previous or benchmark data is available, DRTs allow one to identify atypical sets of degrees of freedom, which potentially lead to design novelty and support novice engineers to formulate the optimization problem.

In the literature, the majority of the works focuses on the visualization of big data structures. Due to the quantity of available methods, identifying the advantages and applications targeted by each technique is a challenging task. Thus, in recent works, different taxonomies and benchmark experiments using the algorithms are presented to support data scientists to select techniques for particular applications [163, 187, 43]. In one of the surveys [43], the authors benchmark 44 techniques on 18 different data sets and evaluate the performance of the methods according to seven metrics, which include similarity of data patterns and preservation of pair-wise distances. Based on their experiments, t-distribution stochastic neighborhood embedding (t-SNE) [162], uniform manifold approximation and projection (UMAP) [105] and interactive document map (IDMAP) [110] have, on average, the best quality indicators, and UMAP is the method that best preserves the pair-wise distances between samples.

However, design optimization frameworks require a bijective mapping between geometric data and the design space (Fig. 5.11), which is infeasible with the aforementioned techniques. Alternatively, in a recent analysis of dimensionality reduction techniques [159], the authors compared the per-

formance in evolutionary optimization of the representations obtained with principal component analysis (PCA), kernel principal component analysis (KPCA), autoencoders (AE), and variational autoencoders (VAE). In a set of surrogate-assisted optimization experiments using benchmark functions, the authors show that AEs represent the designs with higher accuracy, but PCA allows the optimization algorithm to achieve solutions that are closer to the true optima. Similarly, D'Agostino *et al.* optimize the shape of a Burke-class destroyer using representations obtained with PCA and alternative nonlinear methods [34]. Based on the fitness of the optimized shapes, the authors claim that nonlinear methods are more efficient than PCA-based techniques, among which deep autoencoders provide the best performance overall.

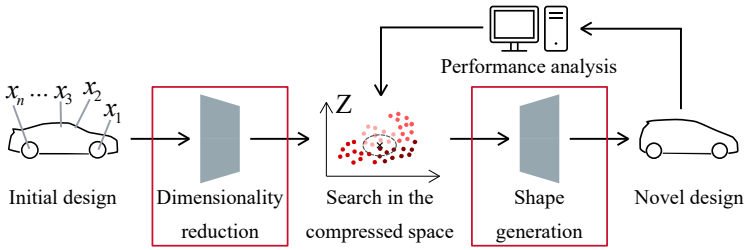

Figure 5.11 – Design optimization workflow with the steps where the mapping between design representations is utilized are highlighted in red.

However, despite the promising results obtained with PCA-based techniques and autoencoders, in the reference works, methods are applied on Euclidean data. In fact, an analysis of DRTs that include geometric deep learning architectures is currently missing in the literature. Hence, in the following set of shape-generative experiments, PC-AE-Rios and two PCA-based techniques are compared based on three criteria: Shape reconstruction quality, shape sensitivity, and shape-generative performance.

5.3.2 Experiment Settings

CAE Data Set: The data set utilized in this set of experiments comprises 3500 shapes randomly selected from the car class of ShapeNetCore. These shapes are sampled into 3D point clouds with 6146 points using the shrink-wrapping

method (Section 4.5). For training PC-AE-Rios, the point clouds are organized in 3D matrices with shape (B_s, 6146, 3), where $B_s = 50$ is the batch size. However, the other DRTs are formulated assuming that the input is described in an Euclidean domain. Hence, for PCA and KPCA, the data set is arranged as a 2D matrix, where each row contains the Cartesian coordinates of the points of one point cloud in the format $\{x_1, \ldots, x_{6146}, y_1, \ldots, y_{6146}, z_1, \ldots, z_{6146}\}$.

Design Representations: PC-AE-Rios is set with the standard architecture hyperparameters (Table 4.1) and three latent representation sizes are considered: $L_z = (5, 10, 20)$. The utilized training algorithm and hardware setup is the same as described in Section 4.1, and the weights of the network are optimized by minimizing the mean-squared distance (MSD, Eq. 4.10) for a maximum number of 750 epochs.

The representations based on PCA and KPCA are generated using the algorithms available in [119] and set with the standard hyperparameters proposed in the software library. For KPCA, five different kernels are utilized to generate the representations: Linear (KPCA-L), polynomial (KPCA-P), radial basis functions (KPCA-R), sigmoid (KPCA-S) and cosine (KPCA-C). The representations generated with PCA-based methods have the same dimensionalities as considered for PC-AE-Rios.

Evaluation Criteria: The shape-generative methods are compared according to three criteria: Shape reconstruction accuracy, shape sensitivity, and generation of crossover shapes. The reconstruction accuracy evaluates the capability of the methods to reconstruct the input data, similarly to the loss function utilized to train the autoencoders. The accuracy is calculated between input and output point clouds using the modified Chamfer Distance (CD) [44].

The second criterion is the shape sensitivity, which quantifies the impact of disturbances in the low-dimensional representations on the shape reconstructions. In this case, the total sensitivity ST between the j-th variable of the low-dimensional representations z_j and the displacement $d(p_i, \tilde{p}_i)$ of the i-th point in the output point cloud is determined according to the Sobol's method [136]. Hence, for the representations of 30 shapes randomly selected from the data set, $30(L_z + 2)$ variations within $\pm 30\%$ of the value of each variable z_j

are generated according to Saltelli's sampling scheme and the corresponding 3D point clouds are reconstructed. After computing ST for every point and variable in the low-dimensional space, the values are projected onto the output point clouds as color maps, similarly to the feature visualization, to identify the regions in the shapes that are affected by each of the low-dimensional variables.

The third criterion concerns the capability of the methods to combine and exploit the features represented in the low-dimensional space. Here, a crossover shape is defined as a design generated by mixing the characteristics of two or more other known designs, *e.g.*, a vehicle with the front design of a sedan and rear design of a sports car. The evaluation in this criterion is divided in two parts. In the first, for 50 different pairs of shapes, the linear interpolation in the low-dimensional spaces is compared to the direct interpolation of corresponding points in the Cartesian space. The underlying motivation is that the interpolation assesses global shape modifications and provides an insight on the smoothness of the feature landscape learned by each method. In the second part, the objective is to generate shapes by modifying local geometric features. Hence, crossover shapes are generated by transferring features that map local geometric characteristics according to the sensitivity analysis, *e.g.*, an airfoil in a vehicle. In this case, the methods are compared based on visual inspection and point-wise distances measure between initial and modified shapes.

5.3.3 Analysis of the Representations

Shape Reconstruction Losses: As observed in previous experiments (Section 4.3), the reconstruction quality improves with an increase of the dimensionality of the representations (Fig. 5.12). By comparing the DRTs, PC-AE-Rios and PCA perform similarly and better than KPCA, regardless of the utilized kernel, which is confirmed by visually inspecting the reconstruction data set samples (Fig. 5.13). Hence, for simplicity, in the following analyses only the representations learned by PC-AE-Rios, PCA and KPCA-P are considered.

Shape Sensitivity to Design Variables: For the second criterion, the methods are initially compared based on the maximum value of total sensitivity (ST)

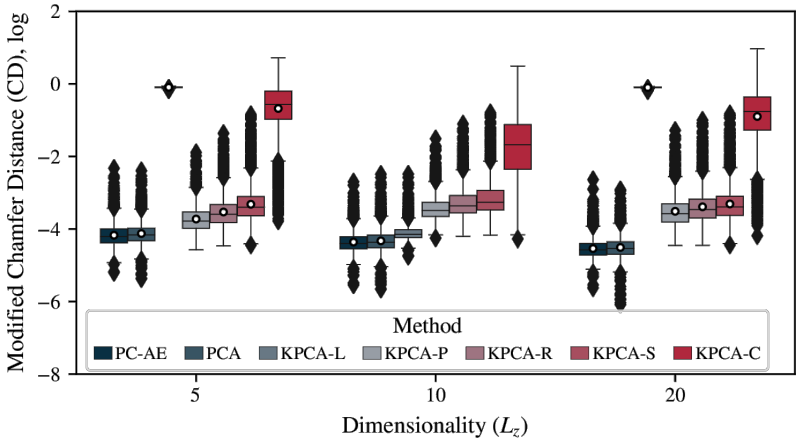

Figure 5.12 – Reconstruction losses obtained for different DRTs and representation dimensionalities. The index *-j* of the KPCA models indicates the utilized kernel.

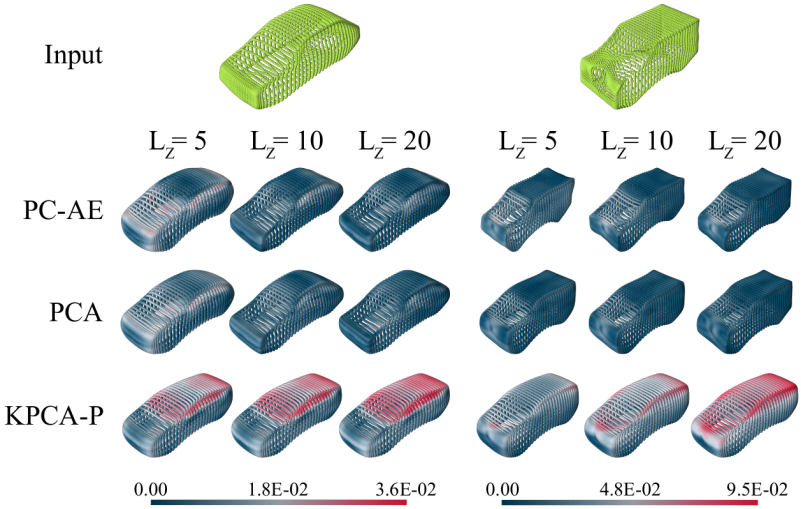

Figure 5.13 – Reconstructions of two data set samples obtained with the selected methods for different representation dimensionalities. The colors indicate point-wise distances measured with respect to the input shapes.

obtained for each of the variables of the low-dimensional representations (Fig. 5.14). For the PCA-based methods, the maximum ST values decrease with the rank of the variables, regardless of the dimensionality. However, for the representations learned with PC-AE-Rios, the maximum ST values are more uniformly distributed over the variables and, in general, higher than obtained with the other techniques. These differences in sensitivity are also in line with the formulation of the methods: PCA ranks the components based on their contribution to the variance of the data, while the autoencoder has more degrees of freedom and the utilized loss function neglects the ordering of the latent variables.

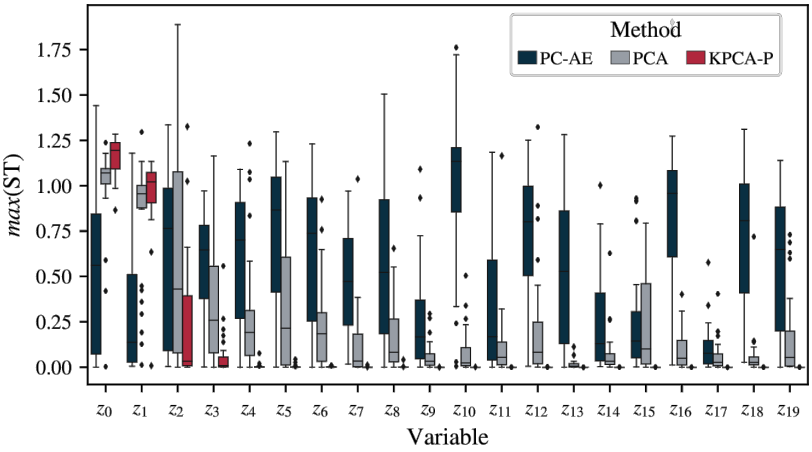

Figure 5.14 – Maximum values of total sensitivity (ST) for each variable of the 20D representations generated with PC-AE-Rios, PCA and KPCA-P.

In a second analysis, the total sensitivity values are projected onto the 3D point clouds for visual inspection (Fig. 5.15). In line with the previous analysis, PCA-based methods highlighted larger regions of the shapes and mostly for the three first components, which is particularly evident for KPCA-P. The regions highlighted by PC-AE-Rios are in general smaller and target complementary regions of the shapes. These results indicate that PC-AE-Rios potentially maps local geometric features more efficiently that PCA-based methods, which compress the point cloud data through global transformations.

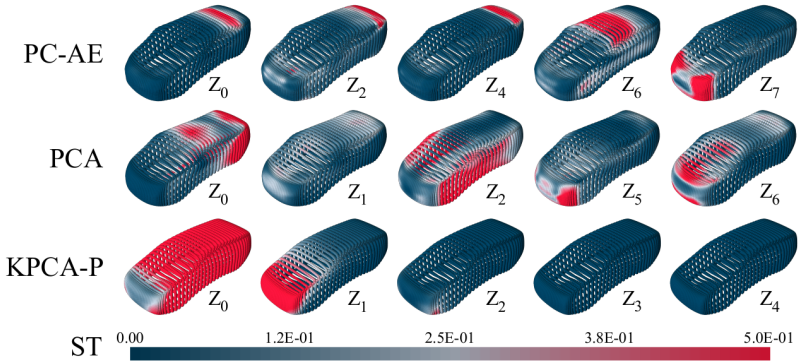

Figure 5.15 – Total sensitivity (ST) values projected as color map onto a sampled geometry for visualizing the areas affected by each variable of the representations.

Exploitation of Low-Dimensional Features: In the first experiment, the 10D representations of 50 pairs of shapes (Z_1, Z_2) are linearly interpolated to generate 20 intermediate representations Z_i (Eq. 5.2). The generated designs are reconstructed and compared to the shapes obtained by directly interpolating corresponding points in the Cartesian space. The utilized metric for comparing the designs is the mean-squared distance between corresponding points (MSD).

$$Z_i = Z_1 + s_t(Z_2 - Z_1),\ s_t = \{0.00, 0.053, \ldots, 1.00\}\ . \tag{5.2}$$

By comparing the MSD values obtained for each representation (Fig. 5.16), the results show that PC-AE-Rios and PCA perform similarly and generate designs with better quality than obtained with KPCA-P. Also, the visualization of a 2D embedding of the representations obtained using UMAP [105] shows that the results are not biased by interpolating similar shapes or within a constrained region of the represented space. Furthermore, the PC-AE-Rios generates intermediate shapes (s_t from 0.21 to 0.74) more dissimilar to the designs obtained with the interpolations in the Cartesian space than PCA. Thus, it indicates that the landscape of the feature space learned with PC-AE-Rios is less smooth than obtained with PCA.

The conclusions drawn based on the MSD values are confirmed by the

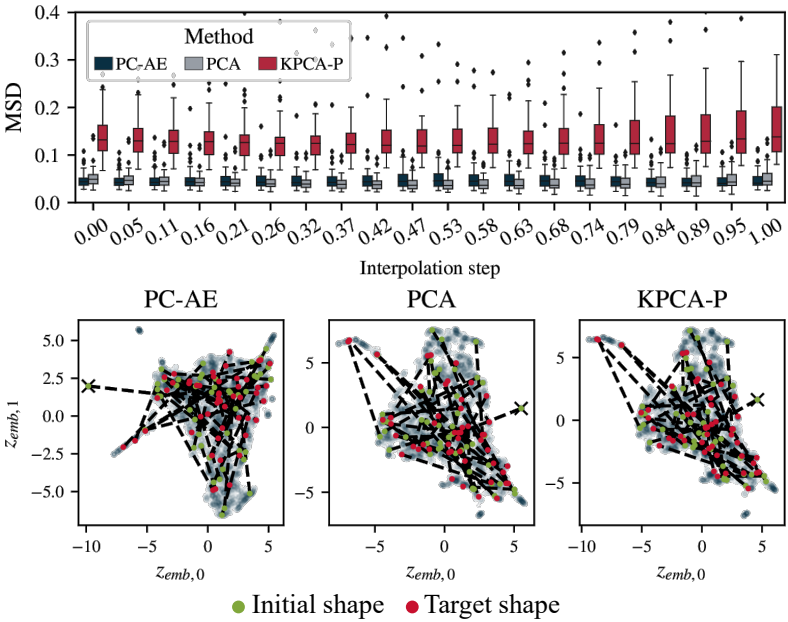

Figure 5.16 – MSD calculated at each interpolation step for all selected models and DRTs (top), and low-dimensional representations of the data set, initial and target shapes embedded in a 2D space (bottom).

visually inspecting the reconstruction of the interpolated shapes (Fig. 5.17). Except for KPCA-P, the methods generate similar intermediate designs and, visually, PCA modifies the design more uniformly than the other methods.

In the second experiment, the design modifications are performed by transferring sets of variables between the representations of different shapes. Specifically, the features that map the front of a particular car shape (according to the sensitivity analysis) are transferred to other 50 randomly selected shapes. Visual inspection of the designs and the MSD between the *target* and *source* designs are utilized to compare the representations. For clarity, the *source* and *target* are the designs from which and to which the selected features are transferred, respectively.

Based on the obtained MSD values, PC-AE-Rios generates shapes that are more similar to the target (initial) shape than obtained with the other methods:

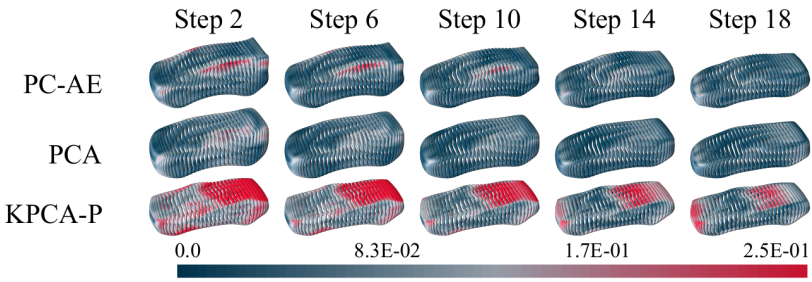

Figure 5.17 – 3D reconstruction of the interpolated representations for five different steps. The color map indicates the point-wise distance calculated with respect to the shapes interpolated in the Cartesian space.

(8.87 ± 1.18)E-02 against (9.52 ± 0.76)E-02 for PCA and (1.52 ± 0.32)E-02 for KPCA-P. Thus, PC-AE-Rios modifies the target shapes more locally and preserves better the general design characteristics than PCA-based techniques, which is confirmed by visually inspecting the reconstructed shapes (Fig. 5.18). The results also demonstrate that PC-AE-Rios learns a more diverse set of degrees of freedom than PCA-based techniques, as expected based on the mathematical formulation of each method.

5.3.4 Extended Experiment: Vehicle Aerodynamic Optimization

In the previous analyses, the shape generation is considered as a standalone task. In the following set of experiments, the representations are utilized in design optimization problems inspired in real-world scenarios. Hence, the performance of the methods is compared based on the convergence of the results and quality of the optimized designs.

Vehicle Aerodynamic Design Optimization: Two sets of single-objective optimization problems are proposed: The minimization of the aerodynamic drag (F_x) and lift (F_z) of five different 3D car shapes (Fig. 5.19). The aerodynamic drag plays an important role in the energetic efficiency of vehicles, as the power consumed by the drag force increases cubically with the vehicle's velocity [41]. Differently, the negative lift (often named as *downforce*)

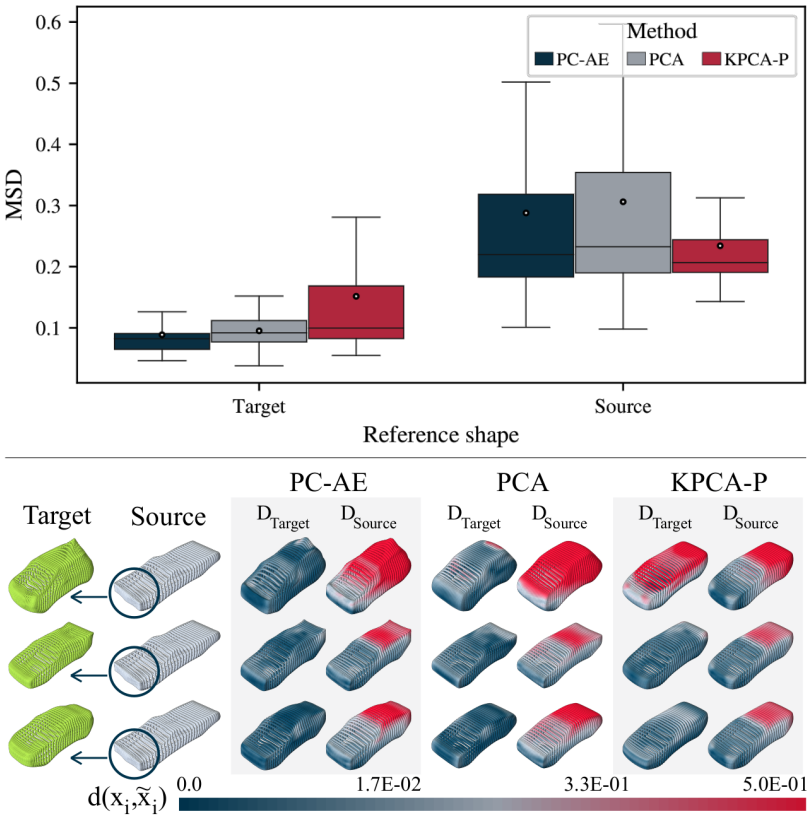

Figure 5.18 – Reconstruction of the shapes based on the representations with transferred features (top) and the MSD calculated between the reconstructed, source and target shapes (bottom).

increases the grip of the tires to the ground, which improves the drivability.

Both problems are formulated as constrained optimizations (Eq. 5.3). Thus, the objective function comprises two terms. The first term is the aerodynamic drag (F_d) or lift (F_l) force. The second term is a design penalty, which corresponds to the MSD between the generated shapes ($S(Z_i)$) and a reference design ($S(Z_{i,N})$). The reference design is the shape of the training set with corresponding low-dimensional representation $Z_{i,N}$ that yields the minimum Euclidean distance to the representation Z_i of the i-th generated individual.

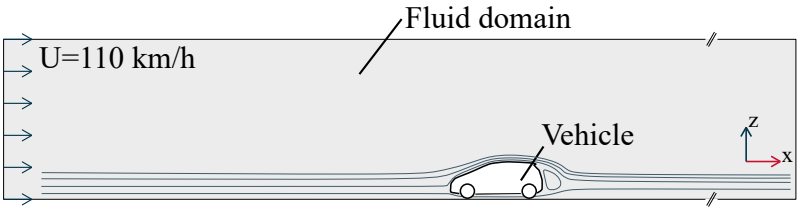

Figure 5.19 – Schematic of the simulation model utilized to calculate the aerodynamic forces during the optimizations.

The penalty is weighted by a factor ρ that adjusts the influence of the penalty on the objective function $\Psi(Z)$. In the following cases, ρ is set to 750 (defined empirically).

$$\min_{Z \in \mathbb{R}^{L_z}} \Psi(Z) = F_{d,l}(Z) + \rho MSD(S(Z_i), S(Z_{i,N})) \,. \tag{5.3}$$

The aerodynamic forces are calculated through computational fluid dynamics (CFD) simulations using OpenFOAM®[1]. For the simulation model, it is assumed that the shapes are symmetric with respect to the x-z vehicle midplane, and drive in a straight line in the x-direction with a velocity of U =100 km/h. The simulations are performed in a cluster of machines with 2 Westmere 4 Core Xeon E5620 clocked at 2.4 GHz and each simulation is solved in parallel with 16 processors.

Optimization settings: The utilized optimization algorithm is the covariance matrix adaptation evolutionary strategy (CMA-ES). Since the method yields high convergence ratios for small population sizes, the method is often preferred for optimizing problems involving computationally expensive function evaluations, such as with CFD. The selected strategy is ($\mu = 3, \lambda = 10$), the initial step size is set to $5\mathrm{E}{-}02$ and the maximum number of generations is limited to 15 due to the computational costs of the simulations.

The considered design spaces are the 20D representations of vehicle designs learned with PC-AE-Rios, PCA and KPCA-P. All representations are utilized for optimizing the same five vehicle designs randomly selected from the data

[1] Available in https://www.openfoam.com/

set and with respect to both, aerodynamic drag and lift forces. For simulating the shapes, water-tight genus-0[2] meshes are reconstructed on the car shapes using the shrink-wrapping approach described in Section 4.5.

Analysis of the Results: In a first analysis, the methods are compared based on the performance of the fittest individuals over the generations (Fig. 5.20). On average, PC-AE-Rios improved the efficiency of the optimizations compared to the other representations. In the last generation, the obtained values of $\Psi(Z)$ in the cases with PC-AE-Rios are 5.45% and 54.2% better than obtained with PCA for the drag and lift scenarios, respectively. In line with the previous experiments, KPCA-P led to poor performance and generates solutions with similar fitness as the initial vehicle designs.

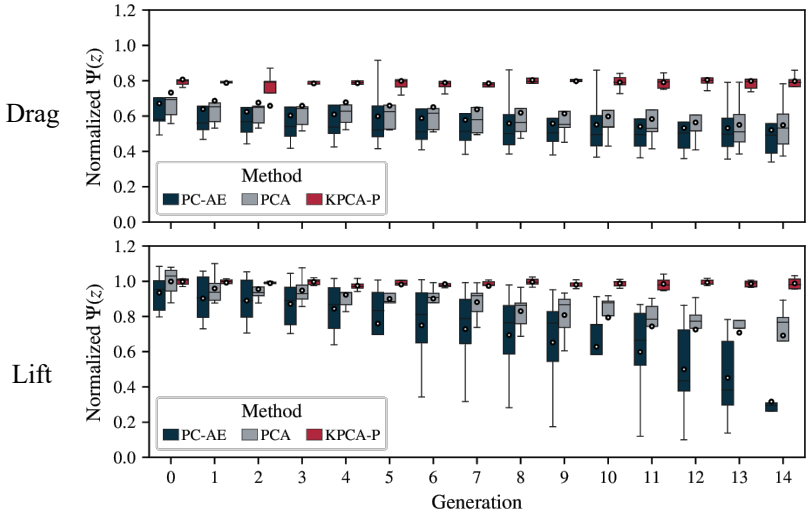

Figure 5.20 – Value of the objective function obtained for the fittest individuals per generation for all representations and optimization scenarios.

The justification for the difference in performance is the type of geometric modifications targeted by each optimization. In the case of aerodynamic drag, the force has a linear and positive correlation to the projected frontal area A_f

[2] The genus of a surface is defined as the maximum number of cuttings along a non-intersecting closed simple curve that does generate disconnected manifolds. *E.g.*, a sphere is a genus-0 shape, while a torus is a genus-1 shape.

of the shape. Since minimizing A_f requires global shape modifications, the optimizations with PCA and PC-AE-Rios achieve similar results. Differently, the lift force depends more on smoothly redirecting the airflow upwards, which is achieved by performing local shape modifications. Hence, since PC-AE-Rios performs local modifications better than the PCA-based methods, the optimization with the PC-AE-Rios generated better results.

This interpretation is also verified through the visual inspection of the optimized shapes (Fig. 5.21). The shapes that minimize the aerodynamic drag are similar and have a smaller projected frontal area than the initial designs. In the optimizations of the lift force, PC-AE-Rios generated a structure that is similar to a spoiler (highlighted with circles in Fig. 5.21), which is a solution typically utilized by engineers to increase the downforce. Hence, PC-AE-Rios modifies local geometric features more efficiently than PCA-based techniques, which improves the quality of the results in the proposed optimization cases.

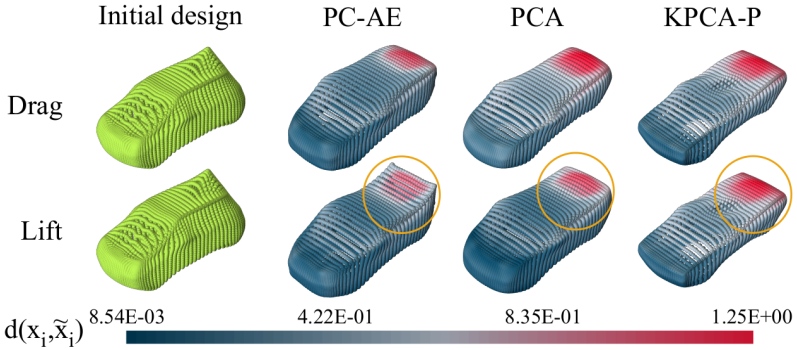

Figure 5.21 – Reconstruction of the optimized shapes obtained with the selected methods for two initial designs and both, drag and lift optimizations. The color indicates the distance between corresponding points in the initial and optimized shapes.

5.3.5 Outlook

In this section, the representation learned by PC-AE-Rios is analyzed from a shape-generative perspective and with respect to two other data-driven methods: PCA and KPCA. The DRTs are evaluated on a benchmark data set

of car shapes and compared based on shape reconstruction quality, sensitivity to design modifications and capability to modify local geometric features. In general, the representations learned with PC-AE-Rios and PCA generate shapes with higher quality than KPCA, which performs poorly regardless of the utilized kernel. The sensitivity analysis shows that PC-AE-Rios learns a more diverse set of degrees of freedom to manipulate the shapes than PCA. This property also allows the autoencoder to modify local geometric features more efficiently than PCA, as confirmed in the experiments with interpolation and transfer of features. Finally, in a set of vehicle aerodynamic optimization problems, PC-AE-Rios outperforms the other methods, particularly in the cases that require local shape modifications, which is the main advantage of the representation over the PCA-based techniques.

5.4 Conclusion

The work in this chapter is mainly based on works previously published by the author [129, 130], which are extended based on more recent findings of the research. The objective of the proposed methods and analyses is to improve our understanding on the information learned by PC-AE-Rios. For that purpose, the latent features are first analyzed from the data compression perspective, where a novel feature visualization technique is utilized to identify the regions of the input shapes that are mapped by the latent features. Then, in a second series of experiments, the latent features of PC-AE-Rios are evaluated from the shape-generative perspective, where the autoencoder is compared to PCA-based dimensionality reduction techniques. The analyses in this chapter address the research questions *RQ3* and *RQ4*, which are answered in the following paragraphs.

[RQ3] What is the geometric interpretation of the features learned by the selected deep-generative method?

The features learned by PC-AE-Rios indicate the occupancy of irregular 3D volumes defined in the input space of the autoencoder, which in this research is defined as $P = \{Z \in \mathbb{R}^3 \mid Z \in [0,1]^3\}$. As the volumes are

fixed with respect to the global coordinate system, the features vary with the position, orientation and scale of the input point clouds. Furthermore, the regions mapped by the features explain the main design variations observed in the data set, as verified by comparing the activations and point-wise distances obtained between representative shapes of design clusters. Hence, an increase in the dimensionality of the latent space allows PC-AE-Rios to refine the mapping and learn more localized differences of the shapes, which improves the shape-generative quality.

[RQ4] What are the advantages of learning design representations using a deep-generative model with respect to other suitable data-driven techniques?

Compared to alternative data-driven methods (PCA and KPCA), PC-AE-Rios learns a more diverse set of degrees of freedom, which allows one to efficiently modify local geometric features. For design problems that target large scale geometric features, *e.g.*, main dimensions of 3D shapes, PC-AE-Rios performs similarly to PCA-based techniques, which are based on data global transformations. However, for design problems that focus on modifying local geometric aspects independently, *e.g.*, optimization of the aerodynamic lift of 3D shapes, PC-AE-Rios allows one to combine local geometric features of different 3D designs more efficiently than PCA-based methods. Hence, as shown in the experiments, PC-AE-Rios performs efficiently as a shape-generative model for non-linear design optimization problems and finds better solutions within similar computational budget than PCA-based representations.

The answers of the research question also hint on a potential interpretation for knowledge transfer in the envisioned framework for experience-based optimization. If the knowledge transfer is realized by injecting past solutions into the population of new optimization problems, the exchange of latent features through recombination propagates nearly-optimal geometric characteristics to the offspring population, which potentially accelerates the optimization. Therefore, in the following chapter, a multi-task design optimization framework is proposed, which utilizes the latent space of PC-AE-Rios as a common design space to multiple optimization problems.