



# Evaluation of geometric similarity metrics for structural clusters generated using topology optimization

Nivesh Dommaraju<sup>1</sup> · Mariusz Bujny<sup>2</sup> · Stefan Menzel<sup>2</sup> · Markus Olhofer<sup>2</sup> · Fabian Duddeck<sup>1</sup>

Accepted: 25 January 2022 / Published online: 22 April 2022  
© The Author(s) 2022

## Abstract

In the early stages of engineering design, multitudes of feasible designs can be generated using structural optimization methods by varying the design requirements or user preferences for different performance objectives. Data mining such potentially large datasets is a challenging task. An unsupervised data-centric approach for exploring designs is to find clusters of similar designs and recommend only the cluster representatives for review. Design similarity can be defined not only on a purely functional level but also based on geometric properties, such as size, shape, and topology. While metrics such as chamfer distance measure the geometrical differences intuitively, it is more useful for design exploration to use metrics based on *geometric features*, which are extracted from high-dimensional 3D geometric data using dimensionality reduction techniques. If the Euclidean distance in the *geometric features* is meaningful, the features can be combined with performance attributes resulting in an aggregate feature vector that can potentially be useful in design exploration based on both geometry and performance. We propose a novel approach to evaluate such derived metrics by measuring their similarity with the metrics commonly used in 3D object classification. Furthermore, we measure clustering accuracy, which is a state-of-the-art unsupervised approach to evaluate metrics. For this purpose, we use a labeled, synthetic dataset with topologically complex designs. From our results, we conclude that Pointcloud Autoencoder is promising in encoding geometric features and developing a comprehensive design exploration method.

**Keywords** Design exploration · Topology optimization · Design representatives · Data mining · Cluster analysis · Geometric similarity

## 1 Introduction

Recent advances in high-performance computing and simulation tools enable numerical optimization techniques to support engineers by automatically generating a large set of concepts satisfying design requirements. Topology optimization (TO) [1–6] is the most flexible type of optimization to generate novel structural concepts; it optimizes material layout subject to a volume constraint in a given design domain for an objective, e.g., structural compliance under specific loads and supports. One of

the popular gradient-based TO methods is a density-based approach using SIMP (Solid Isotropic Material with Penalization) using optimality criteria (OC) update schemes [1]. Figure 1 shows exemplary structures obtained using SIMP. Despite the origins in structural mechanics, TO has found applications in a wide range of physical disciplines such as fluid mechanics [7], electromagnetics [8], and acoustics [9]; it is widely used in the aerospace and automotive industry, civil engineering, materials science, and biomechanics.

Using TO, multiple designs can be generated using different methods which can be classified into the following three groups:

- *Parameter sampling.* Novel designs can be generated by varying the material properties, constraints, boundary conditions, and hyperparameters of the optimization algorithm. For example, in TO, changing the allowed mass in a prescribed design domain results in a new design (Fig. 1). Furthermore, the problem description,

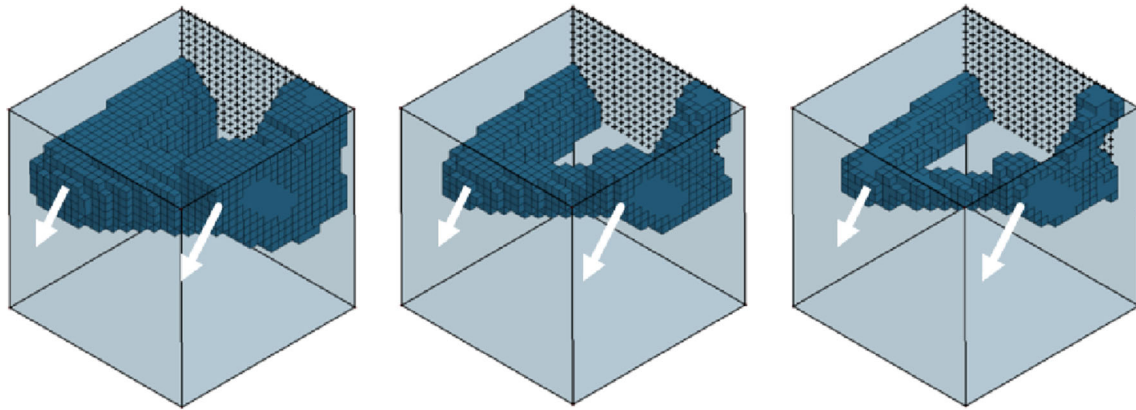
---

This project has received funding from Honda Research Institute Europe GmbH, Germany.

---

✉ Nivesh Dommaraju  
nivesh.dommaraju@tum.de

Extended author information available on the last page of the article.



**Fig. 1** TO designs optimized for structural compliance and constrained to different volume fractions in the cubic design space. The optimization objective is structural compliance of the design under

two fixed loads, shown by white arrows, with a fixed boundary. The allowed volume fraction is ranging from 0.3 to 0.1, from left to right

e.g., the boundary conditions, can also be varied without changing the final design objective. For example, while designing a component to support a fixed load using, say, three support legs, one can change the allowed fraction of load in the supports to obtain a new design [10].

- *Multiobjective optimization.* In practice, designs may need to be optimized for multiple objectives, e.g., energy absorption under crash loads, and structural stiffness under smaller static loads. Multiobjective TO [11] with conflicting objectives yields a set of Pareto-optimal designs, where choosing a design with better performance for one objective results in performance deterioration of another objective.
- *Multimodal optimization.* Highly complex and nonlinear objectives are normally multimodal, i.e., they have several local optima. Since not all the constraints are known in the early development phase, having a set of local optima is useful, in case some of them violate the unknown constraints in the future. Such designs can be identified using evolutionary algorithms [12] or by restarting gradient-based optimization algorithms, such as TO, from different initial configurations [13], converging to a different optimum in each of the runs.

The aforementioned methods can potentially generate multitudes of designs in the early product development phases. A challenging task is to explore the different concepts and identify a few *interesting* designs for further review. A selection of designs can be done based on experience, manufacturing cost, or design performance. For instance, the dream lens tool [10] is an interactive framework that guides a designer to a set of *interesting* designs using, for example, range constraints on performance. For Pareto-optimal designs obtained in multiobjective optimization, the decision-maker (DM) can select designs of interest,

based on the relative importance of different objectives [14, 15]. However, in practice, it is difficult to define objectively the intent of a DM, especially with a large number of objectives [16]. Hagg et al. [17] propose an unsupervised data-centric approach for exploring designs by finding clusters of *similar* designs based on performance, e.g., the drag coefficient of an airplane. The medoids of different clusters are interpreted as representative designs, which can be recommended to an engineer for review and further development. In this study, we use this automated approach to explore designs. However, the engineer still needs to choose the features, metrics, and clustering methods, which determine the final recommended designs.

Along with the performance of a design, differences in geometric properties such as size, shape, and topology are also important, especially at the early stages of product development. Quantifying the geometrical differences allows the identification of *similar* designs and their representatives in a dataset [18]. Furthermore, metrics for geometrical differences allow the use of similarity-controlled optimization methods [19, 20] to yield designs similar to a set of reference designs, which might be desirable due to economic reasons, manufacturing limitations, or ease of integration into the existing design process. In all of these applications, the metric used to measure geometrical differences is crucial and is the subject of our study.

Numerous metrics exist with varying degrees of accuracy and computational complexity. Due to the high-dimensionality of 3D geometric data, each metric may only compare a few of the geometric properties depending on the complexity of designs. Hand-crafted *geometric feature* vectors based on surface curvature, material distribution statistics, or spectral descriptors [21] can be used to distinguish designs. In contrast, data-centric methods—which are more successful in practice—extract features relevant to a specific 3D geometric dataset, e.g., PCA (Principal

Component Analysis) [22] can identify a reduced number of uncorrelated features that explain the variance in data. Other dimensionality reduction techniques popular in the machine learning field can also be used for *geometric feature* extraction, e.g., NMF (Non-negative Matrix Factorization) [23], t-SNE (t-distributed Stochastic Neighbor Embedding) [24], or UMAP (Uniform Manifold Approximation and Projection) [25]. More sophisticated methods based on deep learning networks exist too—Qi et al. [26] use PCAE (Pointcloud Autoencoder) to learn features from a pointcloud obtained by sampling points on the surface of the design. The reduced representation has an additional benefit of associating *geometric features* to a design like any other performance measure. Though, it is not clear if the Euclidean distance with the reduced representation is as meaningful as the *reference* metrics such as voxel distance. So, we propose a novel method to validate the Euclidean distance by comparing its similarity with *reference* metrics commonly used in 3D object classification.

In this paper, we quantitatively compare different metrics of geometrical differences based on criteria that are important when analyzing topologically complex design datasets obtained using structural optimization. The criteria considered in this work are as follows:

- The metrics should be sensitive to geometrical differences in size, shape, topology, and orientation in a design space. Invariance to rotation, reflection, and translation operations is considered as an advantage in 3D design classification, but not for TO designs, where the configuration relative to the boundary conditions is important.
- The metrics should allow the identification of clusters of similar designs even in topologically complex datasets. This allows the recommendation of diverse designs using clustering methods.
- Metrics that use vectors of *geometric features* associated with designs are preferred since the feature vectors enable 2D visualization of the complete dataset using manifold learning techniques [18, 24], easing data exploration. Furthermore, such features can be combined with performance features directly. However, in this case, we need to ensure that the Euclidean distance in the space of *geometric features* is still meaningful.

In what follows, Section 2 discusses different geometric representations, such as voxel and pointcloud representations, of 3D geometric data, which are used by some intuitive *reference* distance metrics of geometrical differences (Section 3). Section 4 introduces a few dimensionality reduction techniques to extract *geometric features* which are used to derive new metrics. In Section 5, we present methods to evaluate metrics based on different properties. Section 6 describes the datasets used to evaluate the metrics. The results of the evaluation are shown in Section 7. Using simple datasets, we verify if the features capture our intuition on geometrical differences. With topologically complex design sets, the properties of different metrics are highlighted. Since our final goal is to identify geometrically similar design classes in a dataset, we compare different feature extraction methods based on the clustering performance with the use of a challenging dataset. Finally, in Section 8, we explore topologically optimized designs using *geometric features*. Section 9 concludes this study with the key results and an outlook for further research.

## 2 Geometric representation

The geometric representation of a design determines the available metrics and feature extraction methods. In this study, we consider two geometric representations: voxel and pointcloud representation (Fig. 2). The former is a natural

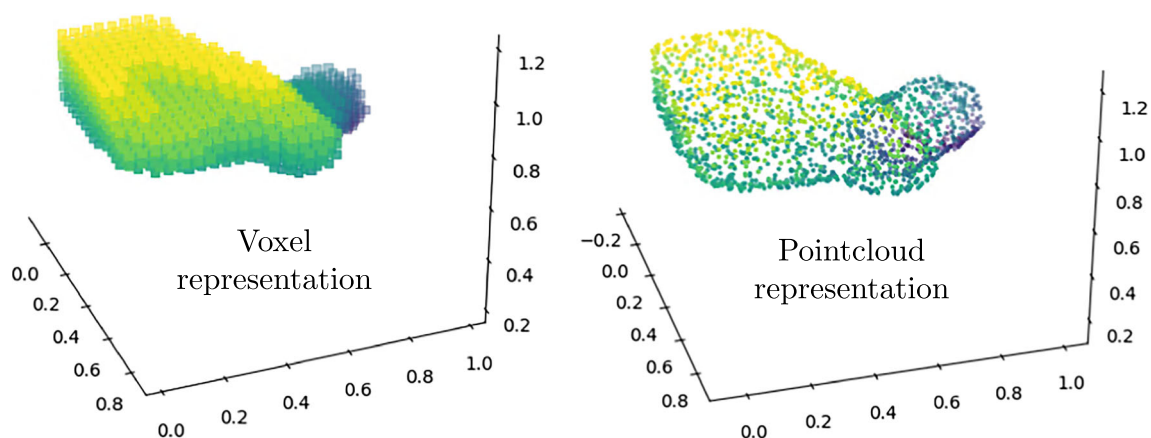


Fig. 2 Geometric representations of TO results

representation in TO since the design domain is generally discretized into voxels [1], while the latter is a compact and expressive representation popular in 3D object recognition, classification, and segmentation [26, 27].

**Voxel representation** A 3D domain containing the design is discretized into voxels using a regular grid. The voxel representation of the design is a vector of values  $\mathbf{x} \in \{0, 1\}^n$ , where  $n$  is the number of voxels. Each component  $x_i$  corresponds to a specific voxel in the design domain. If the design occupies the voxel  $i$ ,  $x_i = 1$ , otherwise,  $x_i = 0$ . For voxels that are only partially occupied by the design, we assign  $x_i = 1$  when the majority volume of the voxel is occupied. The vector  $\mathbf{x}$  can be large since the design space may be finely discretized to resolve the complexity of the design. This representation is convenient for TO designs which are generally optimized in a voxelized domain [1]. An interesting research question is how well the dimensionality reduction techniques, currently popular in the machine learning field, can identify underlying design patterns and extract relevant features from the voxel representations of TO designs.

**Pointcloud representation** Compared to the voxel representation, a pointcloud is a *compact* representation since it only samples the points on the surface of a design. Geometric learning methods using this representation [26, 27] are interesting since they can identify different classes of shapes in publicly available datasets such as ShapeNet [28].

Other geometric representations include octrees, 3D meshes, and multi-view projections. The octree-based representation [29] alleviates the high memory usage of a voxel representation by using a higher resolution of voxels only when it is required, e.g., near the surface of a design. 3D meshes, similar to pointcloud representations are compact. They represent the surface of a 3D geometry using a set of polygon faces. A graph representation can also be used to represent 3D meshes. Due to the high computational cost of these representations, there is an increasing interest among researchers to work with

low-dimensional pointcloud representations [30]. For our initial analysis of TO designs, we consider metrics provided by voxel and pointcloud representations.

### 3 Reference distance metrics

In this section, we describe a few intuitive metrics used to measure geometrical differences between designs. These metrics serve as a reference to compare with the metrics that are derived from dimensionality reduction techniques as explained in the next section.

**Voxel distance** Voxel representations of two designs can be compared when the voxel arrays have the same size and correspond to the same regular grid in the 3D domain. The Euclidean distance in the voxel representation is equal to the square root of the number of *non-overlapping* voxels that are occupied by only one of the designs. A disadvantage of this metric is that it is insensitive to the position of non-overlapping voxels. For two designs without any overlap, the metric is invariant to their relative position in the design domain, so long as there is no overlap and the voxel grid in the domain does not change.

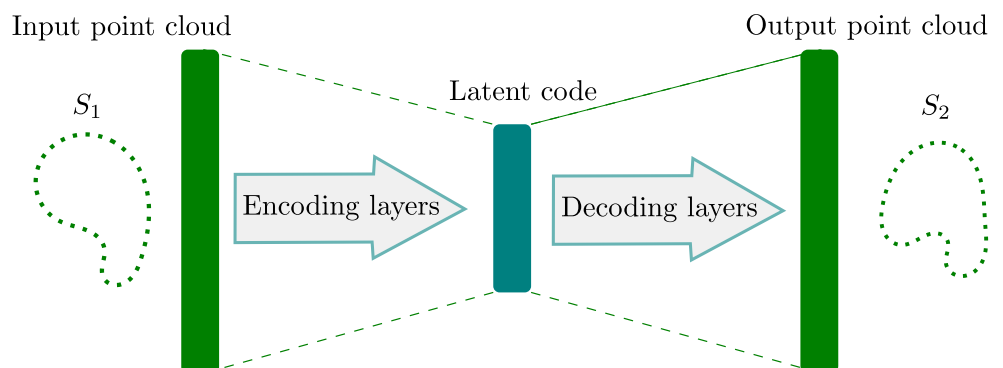
**Chamfer distance (CD)** A pointcloud is a compact representation obtained by sampling points on the surface of a 3D geometry. The CD [26, 27] is a metric used to measure the difference between two pointclouds, say  $S_1, S_2 \subset \mathbb{R}^3$  (Fig. 3). It is defined as follows:

$$CD(S_1, S_2) = \sum_{\mathbf{a} \in S_1} \min_{\mathbf{b} \in S_2} \|\mathbf{a} - \mathbf{b}\|_2^2 + \sum_{\mathbf{a} \in S_2} \min_{\mathbf{b} \in S_1} \|\mathbf{a} - \mathbf{b}\|_2^2, \quad (1)$$

where  $\mathbf{a}, \mathbf{b} \in \mathbb{R}^3$ .

**Earth mover distance (EMD)** Similar to CD, EMD [27] is calculated on a pointcloud representation. It solves an optimization problem to find a mapping  $\Psi : S_1 \rightarrow S_2$  between the points of the pointclouds such that the objective

Fig. 3 Pointcloud Autoencoder (PCAE)





$\sum_{\mathbf{a} \in \mathcal{S}_1} \|\mathbf{a} - \Psi(\mathbf{a})\|_2$  is minimized. We use an approximate but fast algorithm for EMD calculation proposed by Achlioptas et al. [27]. Still, EMD is computationally more expensive than the CD.

The metrics—voxel distance, CD, and EMD—are sensitive to any changes in configuration relative to the boundary conditions, e.g., due to rotation, translation, or reflection. The voxel distance compares very high-dimensional voxel representations. CD and EMD are functions defined with two designs as input. Other metrics such as Wasserstein distance [31] are not considered here since we only need a few reference metrics to demonstrate our metric evaluation method. As discussed previously, the emphasis of this study is to evaluate metrics based on dimensionality reduction techniques, the benefits of which will be discussed later.

#### 4 Metrics based on dimensionality reduction techniques

Dimensionality reduction techniques identify the underlying patterns in a dataset, summarizing each data point with a lower-dimensional feature vector. These methods are applied to reduce the high-dimensional 3D geometric data to yield a feature vector, referred to as *geometric features* in this work. Euclidean distance between these low-dimensional feature vectors, when meaningful, can be used as a metric of geometrical differences. In the later sections, we investigate if such derived metrics are indeed meaningful by comparing them with the reference distance metrics.

In this study, we investigate the use of dimensionality reduction methods such as PCA [22], NMF [23], t-SNE [24], UMAP [25], and PCAE [27]. Although this is not an exhaustive study, these methods are representative and widely used in different fields. PCA extracts non-redundant features using linear transformations [22, 32]. t-SNE and UMAP are nonlinear dimensionality reduction techniques used in machine learning to visualize high-dimensional data. PCAE is an effective feature extractor for pointcloud representation used in object classification [26, 27].

**Principal component analysis** PCA projects a set of  $n$  data points each with  $d$  features,  $X_{d \times n} = \{\mathbf{x}_i \mid \mathbf{x}_i \in \mathbb{R}^d, i = 1, \dots, n\}$ , to a new set of orthonormal basis vectors:  $\{\mathbf{b}_i \mid \mathbf{b}_i \in \mathbb{R}^d, i = 1, \dots, \min(n, d)\}$ . Each data point  $\mathbf{x}_i = \sum p_k \mathbf{b}_k$  can be identified with the new coordinates  $p_k$ . The new basis is constructed such that the new coordinates  $p_k$  are uncorrelated. The dataset  $X_{d \times n}$  has the highest variance in the first principal component and the variance in the components  $p_i$  decreases as the order  $i$  increases [33]. It is often sufficient to consider only a few of the principal components to explain the variance in data, resulting in

the dimensionality reduction. For a voxel representation as input,  $d$  is the number of voxels in the design domain and  $n$  is the number of designs.

**Non-negative matrix factorization** NMF [34] factorizes the input data  $X_{d \times n} = W_{d \times p} H_{p \times n}$  where  $W, H$  have non-negative entries. In general  $p \ll \min(n, d)$ , which means that each data point  $\mathbf{x}_i$  (column  $i$  of  $X$ ) can be expressed as a linear combination of columns of  $W$ , i.e.,  $\mathbf{x}_i = W h_i$ , where  $h_i$  (column  $i$  of  $H$ ) is the reduced dimensional representation. Since  $W$  and  $H$  have non-negative entries, data such as images or voxels are decomposed into interpretable components.

**t-distributed stochastic neighbor embedding** t-SNE [24] is a method to embed high-dimensional data in 2D or 3D. The method is especially useful when visualizing clusters in data because similar data points are kept close in the reduced coordinates with high probability.

**Uniform manifold approximation and projection** UMAP, similar to t-SNE, is a method to visualize high-dimensional data. McInnes et al. [25] argue that UMAP preserves the inter-cluster distance better than t-SNE. Empirical studies using t-SNE [24] and UMAP [25] show how high-dimensional data can be embedded into 2D without losing the cluster structure. For example, 2D image data of handwritten digits can be reduced to 2D clusters, where images of different digits are separated clearly.

**Pointcloud autoencoder** An autoencoder is an unsupervised learning method used to reduce the dimensions of an input representation. Umetani et al. [35] parameterize the surface of a shape, which is assumed to be approximately convex. The parameters defined to generate the surface mesh are used as the input vector to an autoencoder. Recent studies on 3D datasets [26, 27, 36] use autoencoders to extract features from pointcloud representations of designs and identify everyday objects such as chairs, cars, and airplanes. In this study, we use pointcloud representations as input to build a pointcloud autoencoder (PCAE). Pointclouds are simpler to compute and have no additional assumptions on the shape.

For PCAE, we use the neural network architecture proposed by Achlioptas et al [27]. Schematically, the architecture comprises two stacks of neural network layers: encoding and decoding layers. Encoding layers reduce the dimension of the input pointcloud to result in a latent code that is used by the decoding layers to reconstruct an output pointcloud similar to the input. At the start of the training process, the weights used by the network are randomly initialized and the reconstruction is inaccurate. By measuring the difference between output and input pointclouds using

a loss function, the autoencoder learns to adjust the weights of the network to reconstruct the input accurately. Since the latent code has fewer dimensions than the input representation, the PCAE achieves dimensionality reduction. Rios et al. [37] demonstrate that a pointcloud autoencoder allows the identification of nonlinear subregions in the design space, each preferentially occupied by a subclass of designs. This explains the usefulness of the latent code in object classification. In this study, we use the CD as the loss function instead of EMD to train the PCAE since CD is computationally cheaper than EMD and is found to be sufficient for our application. It provides meaningful *geometric features* and clusters similar designs in the test datasets (Section 7).

Each dimensionality reduction technique yields a *geometric feature* vector for a design. If the Euclidean distance in *geometric features* captures the geometrical differences, they can be treated as any other performance feature. An aggregate vector with *geometric* and performance features can be used to find similar designs. For example, this would help to highlight designs with a similar geometrical structure but with different performance values and vice versa.

To verify if the Euclidean distance in *geometric features* is meaningful, they should be compared with the reference metrics (Section 3) which are designed to measure geometrical differences. We propose a method to do this in the next section.

### 5 Methods for evaluating metrics

In this section, we propose two methods to evaluate the different target metrics. The first method compares a given *target* metric (TM) with a *reference* metric (RM) which is known to capture at least some aspects of geometrical differences. The second method evaluates the metrics by measuring their clustering performance.

#### 5.1 Metric correlations

It is difficult to define geometrical differences between any two 3D geometries objectively. The problem is simplified if the two designs differ only in a simple geometric property. For example, consider a *simple* set of designs that are

obtained by rotating a template design. The angle between any two designs can then serve as an RM. Although such metrics are not generally applicable, we can use them to evaluate more general TMs. If RM is a reasonable metric for a specific dataset, the distances measured by a more general TM should be at least *similar* to that of RM for the given dataset. We discuss here how to measure this *similarity* between any two metrics (e.g., TM and RM).

Consider a set of  $N$  geometries  $\mathbf{G} = \{G_e \mid e = 1, \dots, N\}$  and a collection of its geometry pairs:  $\mathbf{P} = \{p_i = (G_m, G_n) \mid m \neq n\}$ . A metric,  $M$ , measures distance between geometries of a pair ( $p_i$ ). The collection of distances measured by the metric is given by  $D_M = \{M(p_i) \mid p_i \in \mathbf{P}\}$ . To measure the similarity between the metrics, RM and TM, we find the correlation between  $D_{RM}$  and  $D_{TM}$ . A high correlation between the measured values indicates that the metrics are similar. Figure 4 shows the proposed workflow to compare a TM with an RM.

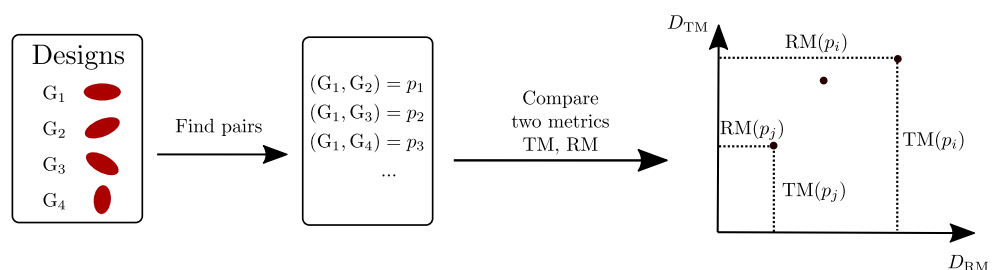
**Correlation measure** Pearson correlation ( $\rho_p$ ) [38] measures the linear correlation between any two input variables. It ranges between -1 and +1. If the variables have a positive linear correlation, we expect the correlation to be near +1. In our case, the two variables are  $D_{RM}, D_{TM}$ . In general, the values of  $D_{RM}, D_{TM}$  may not be linearly related. But if they have a monotonic relation, the metrics can distinguish geometries equally well. So, from hereon, we rely on Spearman correlation ( $\rho_s$ ) [38], rather than Pearson correlation ( $\rho_p$ ), to compare metrics. When the correlation between the distances measured by two metrics is high, i.e.,  $\rho_s$ , we say that *the two metrics correlate well* or *the metric correlation is high*.

The proposed method can empirically determine if a TM can quantify geometrical differences at least as good as the RM. We complement this evaluation by investigating whether the metrics are also good at clustering similar designs in a topologically complex dataset.

#### 5.2 Comparison based on clustering performance

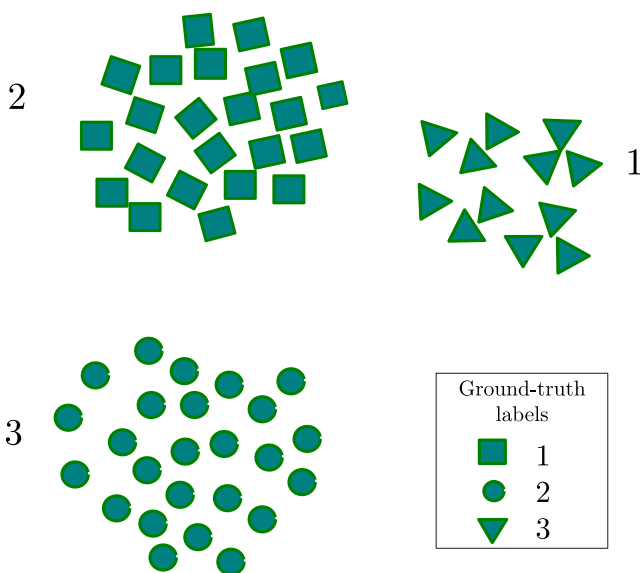
One of the objectives of this study is to identify groups of designs that look geometrically similar. Since geometries in a TO dataset have no pre-determined class labels, we concentrate on unsupervised object classification methods.

**Fig. 4** Workflow to measure similarity between two metrics RM and TM. Note that for each pair of geometries, we obtain two different distance measures by RM and TM. The correlation between the distances measured indicates the degree of similarity between the metrics



In addition to the clustering method chosen, the metric used for clustering leads to different clusters of designs. It is difficult to verify the performance of the different clusterings using unlabeled datasets, even though this is the target application. So, we use labeled test datasets for evaluating our method.

**Relabeling using the majority label method** Each design  $D_i$  in the labeled dataset has a ground-truth label  $g_i$  according to its class and a cluster label  $c_i$  obtained from clustering. Each cluster  $C_k$  in the dataset is a set of designs with a common cluster label  $k$ :  $C_k = \{D_i \mid c_i = k\}$ . If the clustering is successful in identifying the subclasses, then all or most of the designs in a cluster have a common ground-truth label. For example, Fig. 5 shows how a dataset with three subclasses can be assigned arbitrary cluster labels, even if all the subclasses are accurately identified by the clustering algorithm. In practice, when the clustering is not perfect, a cluster can have designs with different ground-truth labels but a *majority* of them may have a single ground-truth label that can be used to remap cluster labels (Fig. 5). This allows for the use of standard measures of classification performance where predicted labels and ground-truth labels are compared. One of the measures, called *precision*, finds the proportion of designs that differ in ground-truth and predicted (cluster) labels. We also use the weighted average of *F1-score* [39] which takes into account both the precision and recall scores for each label and then weighs the score by the number of samples in each cluster.



**Fig. 5** Clusters of designs: Ground-truth labels are different from labels (shown beside clusters) obtained by clustering. The label matching method maps the ground-truth labels 1,2,3 with cluster labels 2,3,1 respectively

In addition to F1-score and precision, we report adjusted mutual information score (AMI) [40], one of the state-of-the-art methods to measure multi-label classification accuracy. AMI is invariant to permutations of the labels and doesn't need the relabeling step discussed previously. It is adjusted for the chance which ensures that the random labeling gets a zero score. However, AMI is not as intuitive as the classification measures discussed previously. Furthermore, by analyzing the F1-score and precision of individual cluster labels, one can identify which design classes are mislabeled.

For evaluating the metrics, the designs available in the public domain such as ShapeNet [28] are topologically not as complex as the designs obtained in TO. So, we generate complex topologies with well-defined subclasses, which are described in the next section.

## 6 Design generation

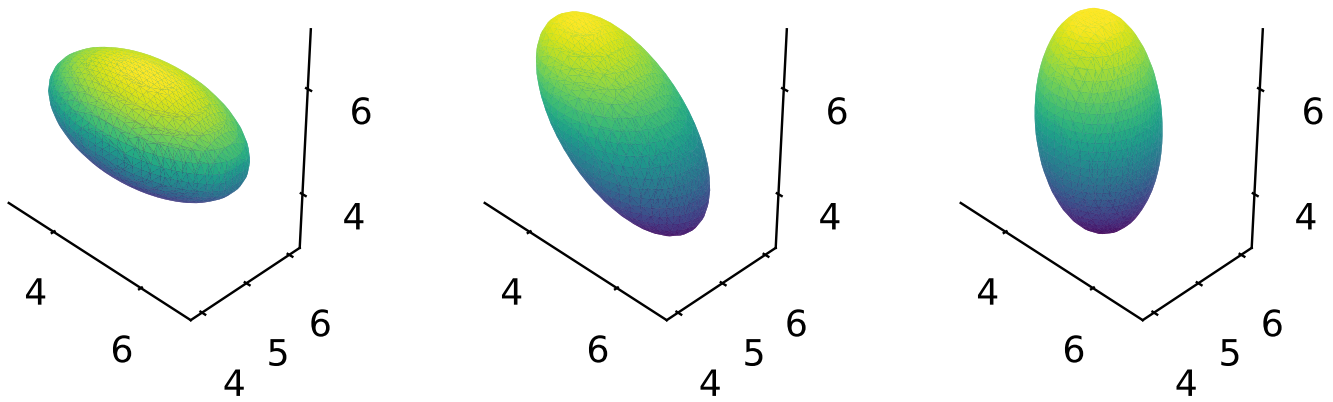
In this section, we present the datasets used to evaluate different metrics. The first part of the section covers simple datasets where the geometrical difference is easier to quantify. This is followed by more complex truss-like designs, which resemble structures created using TO.

### 6.1 Ellipsoidal designs

We generate three datasets where geometrical differences between the designs in a dataset can be easily quantified. Within a dataset, designs are obtained by translation, rotation, or elongation of a reference design. Although these datasets are simple, we can evaluate if our *target* metrics capture geometrical differences that arise from these simple transformations.

The reference geometry for these datasets is an ellipsoid, which can be represented using a Moving Morphable Component (MMC) with a form factor  $m = 2$  [41]. We use these datasets to illustrate our method and draw initial observations, while more sophisticated datasets are presented later. The reference MMC ellipsoid, also called beam here, can be transformed by varying the position of the center of mass ( $\mathbf{C} \in \mathbb{R}^3$ ), lengths along the three principal axes ( $\mathbf{L} \in \mathbb{R}^3$ ), and Euler angles ( $\mathbf{E} \in \mathbb{R}^3$ ) representing the orientation.

The resulting geometry is defined using a level set function  $\Phi : \mathbb{R}^3 \rightarrow \mathbb{R}$ : the surface of the object is given by  $\{\mathbf{x} \in \mathbb{R}^3 \mid \Phi(\mathbf{x}) = 0\}$ . The interior of the object is given by  $\{\mathbf{x} \in \mathbb{R}^3 \mid \Phi(\mathbf{x}) > 0\}$ . The level set formulation is convenient to construct the voxel and pointcloud representations used in this study. As described in Section 2, to construct a voxel representation, a common domain containing the designs is split into voxels. To construct the voxel representation of a design, occupied voxels can be found by evaluating the condition  $\Phi(\mathbf{x}_i) \geq 0$  at the centers  $\mathbf{x}_i$  of the  $i$ -th voxel. Using the marching cubes algorithm [42, 43],



**Fig. 6** Beam-rotation dataset: An ellipsoidal beam is rotated by different angles to create different designs

the volumetric representation is converted to a triangulated surface mesh. For pointcloud representation, points are uniformly sampled on the surface mesh, while ensuring uniform distribution in each triangular face [44, 45].

**Beam-rotation dataset** For this dataset, an ellipsoidal beam is rotated by different angles along a fixed axis to get new designs (Fig. 6). The difference in the rotated angle then serves as the reference metric. The rotation angle is kept below  $90^\circ$  due to the polar symmetry of the object. For rotations above  $90^\circ$ , the difference in angle is not a good metric. For example, consider a beam  $B_1$  rotated by angles  $\theta$  and  $180^\circ - \theta$  to give beams  $B_2$  and  $B_3$  respectively. Due to polar symmetry,  $B_2$  and  $B_3$  fully overlap, but the angular difference indicates that  $B_1$  is more similar to  $B_2$  than  $B_3$ . This dataset contains 20 designs.

**Beam-elongation dataset** For this dataset, an ellipsoidal beam is elongated by different lengths along a fixed principal axis to get new designs (Fig. 7). The difference in lengths along this axis then serves as the reference metric. This dataset contains 20 designs.

**Beam-translation dataset** For this dataset, an ellipsoidal beam is translated to different locations along a fixed axis to get new designs (Fig. 8). The difference in the position

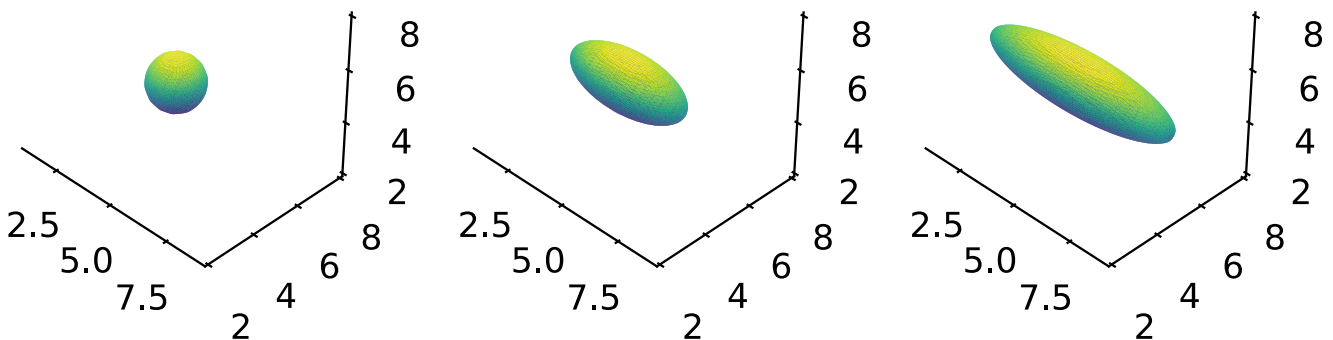
of the center of mass  $C$  then serves as the reference metric. This dataset contains 20 designs.

### 6.2 Topologically-complex designs

A more complex set of truss-like designs can be generated by combining multiple MMCs. Furthermore, we can use MMCs as a basis to generate labeled test datasets with well-defined subclasses.

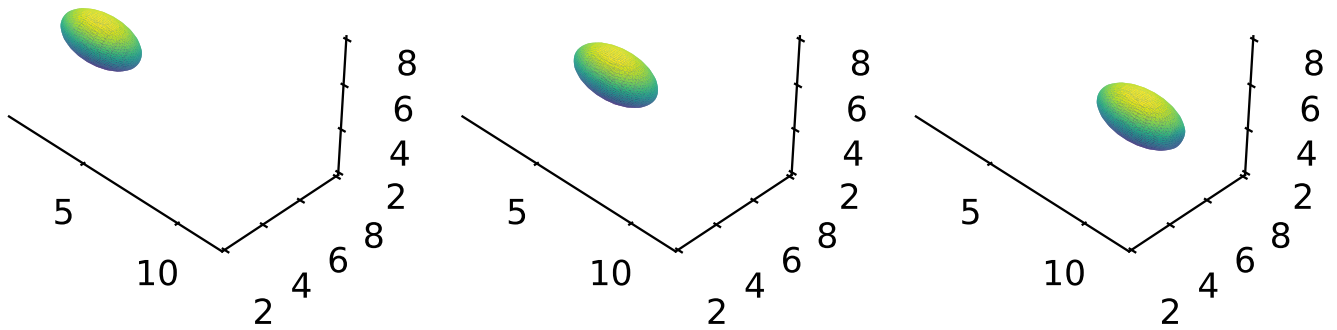
MMC framework and similar feature mapping techniques are increasingly used in TO [46–49] since they can be used to construct complex geometries using a few design variables. Zhang et al. demonstrate the generation of arbitrarily curved beams by overlapping ellipsoids [46]. With a relatively small number of MMCs, MMC-based feature mapping techniques are able to generate complex topologies that are comparable to designs obtained by state-of-the-art density-based TO [50, 51]. Even for highly nonlinear crash TO problems, it is found that the optimal structures are usually composed of interconnected beams [3]. So, it is reasonable to assume that a topologically optimized design can be represented using MMCs.

The interior of a geometry comprising multiple MMCs, say  $n$  MMCs, is defined by  $\max_{i=1, \dots, n} \Phi_i$ . As discussed previously, this allows for a conversion to voxel, surface



**Fig. 7** Beam-elongation dataset: An ellipsoidal beam is elongated to different lengths to create different designs





**Fig. 8** Beam-translation dataset: An ellipsoidal beam is translated by different amounts to create different designs

mesh, and pointcloud representations. A similar set of designs have been used by us [18] to demonstrate a method for design exploration. Since we investigate the effect of topology rather than the effect of the shape, which is extensively studied in the literature [26, 27, 36], we use  $m = 2$  to generate ellipsoidal MMCs.

Since we need complex topologies with well-defined subclasses as a test dataset, we propose to generate a connected truss-like design using a 3D geometric graph as a template. The nodes and edges of the graph are points and line segments in a 3D Euclidean space respectively. A design containing multiple beams is generated by aligning each beam, using one of its principal axes, along a distinct edge in the graph (Figs. 9, 10). We define a *basegraph* to generate a labeled test dataset, i.e., a set of designs with a subset of geometrically similar designs assigned a common *subclass* label. The basegraph is used to generate connected, distinct subgraphs which are used as a template to generate a subclass of similar designs. A subgraph can lead to multiple designs by varying the thickness of beams positioned along the edges of the graph.

In this study, we use three datasets using two different basegraphs with an increasing amount of complexity: a three cube (three back-to-back cubes) and a single cube, as shown in Figs. 9 and 11 respectively. For variations within

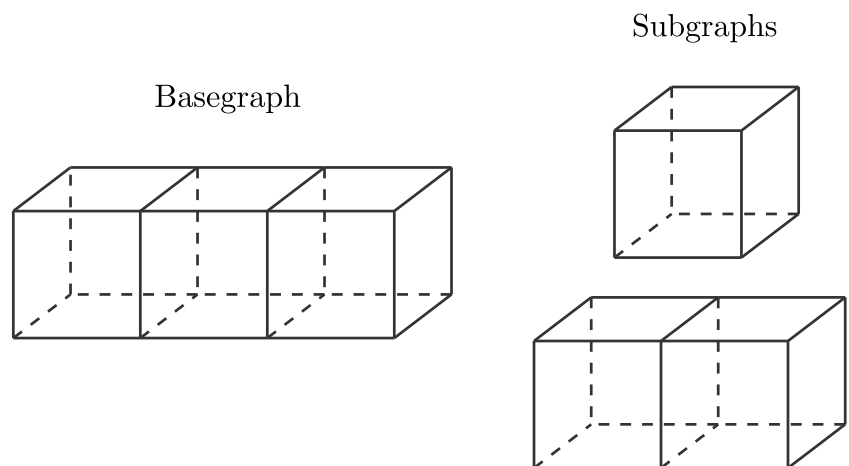
a subclass, we change the thickness of an MMC using a uniform random distribution.

**Three cube trusses** This set of 150 designs is based on the basegraph shown in Fig. 9. Using different subgraphs, 6 subclasses are generated of which two are based on subgraphs as shown in Fig. 9. Samples from six of the subclasses are shown in Fig. 10. Note that the center of mass changes significantly from subclass to subclass.

**Single cube trusses** This set of 275 designs is based on the basegraph shown in Fig. 11. As discussed, each subclass in the dataset is restricted to a connected subgraph. Eleven different subgraphs generate the subclasses. Samples from six of the subclasses are shown in Fig. 12. Note that the designs from different subclasses differ in orientation if they have the same topology.

**Randomized topologies** This topologically more complex dataset challenges the classification performance (Section 5.2) of the clusters obtained by using different metrics. The dataset consists of 50 subclasses with 20 designs per class. Having a sparse number of designs per class challenges the deep learning method, which works better with more data. Each subclass of designs is based on a subgraph

**Fig. 9** Three cube basegraph and subgraphs: Each subclass of designs is restricted to a subgraph of the basegraph. Hidden edges in the basegraph are shown as dashed lines in the basegraph and subgraphs



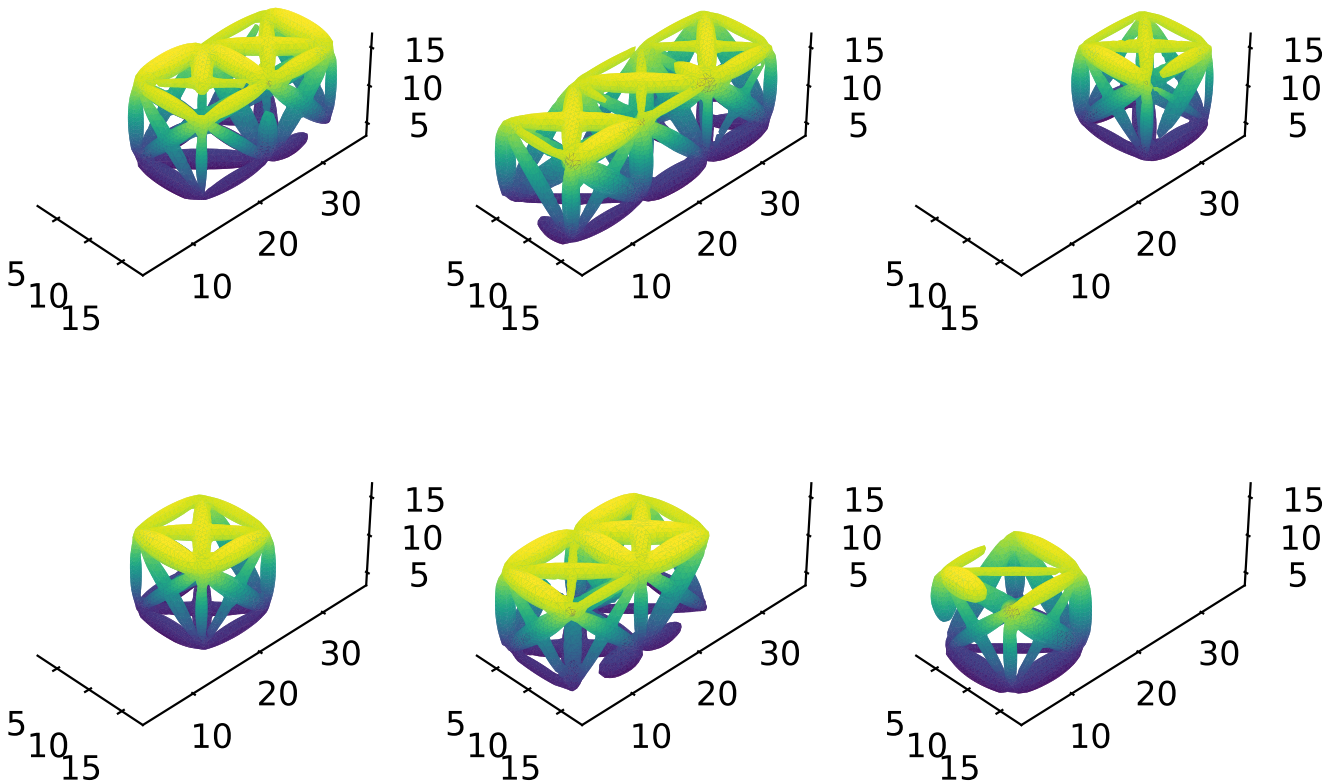


Fig. 10 Three cube truss dataset: Samples from 6 different subclasses are shown

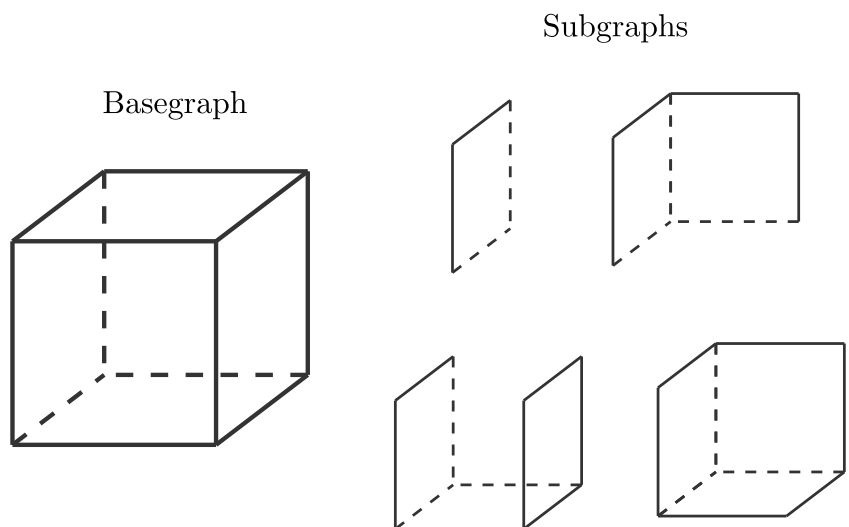
of the three-cube basegraph discussed previously. A subclass of designs is constructed using the following steps, which change not only the thickness of the beams in a design but also the length along the edge:

1. Define a subgraph for each subclass. The subgraph forms a template to construct different designs in the subclass.

- (a) Pick a subset of three-cube vertices (Fig. 9) of size between 5 to 10 randomly.
- (b) Construct an edge for each possible combination of vertices.

2. Mutate the thickness and the length of the beams which are placed along the edges of the subgraph to get a new design in the subclass. Care is taken such that the

Fig. 11 Single cube basegraph and subgraphs: Each subclass of designs is restricted to a subgraph of the basegraph. Hidden edges in the basegraph are shown as dashed lines in the basegraph and subgraphs



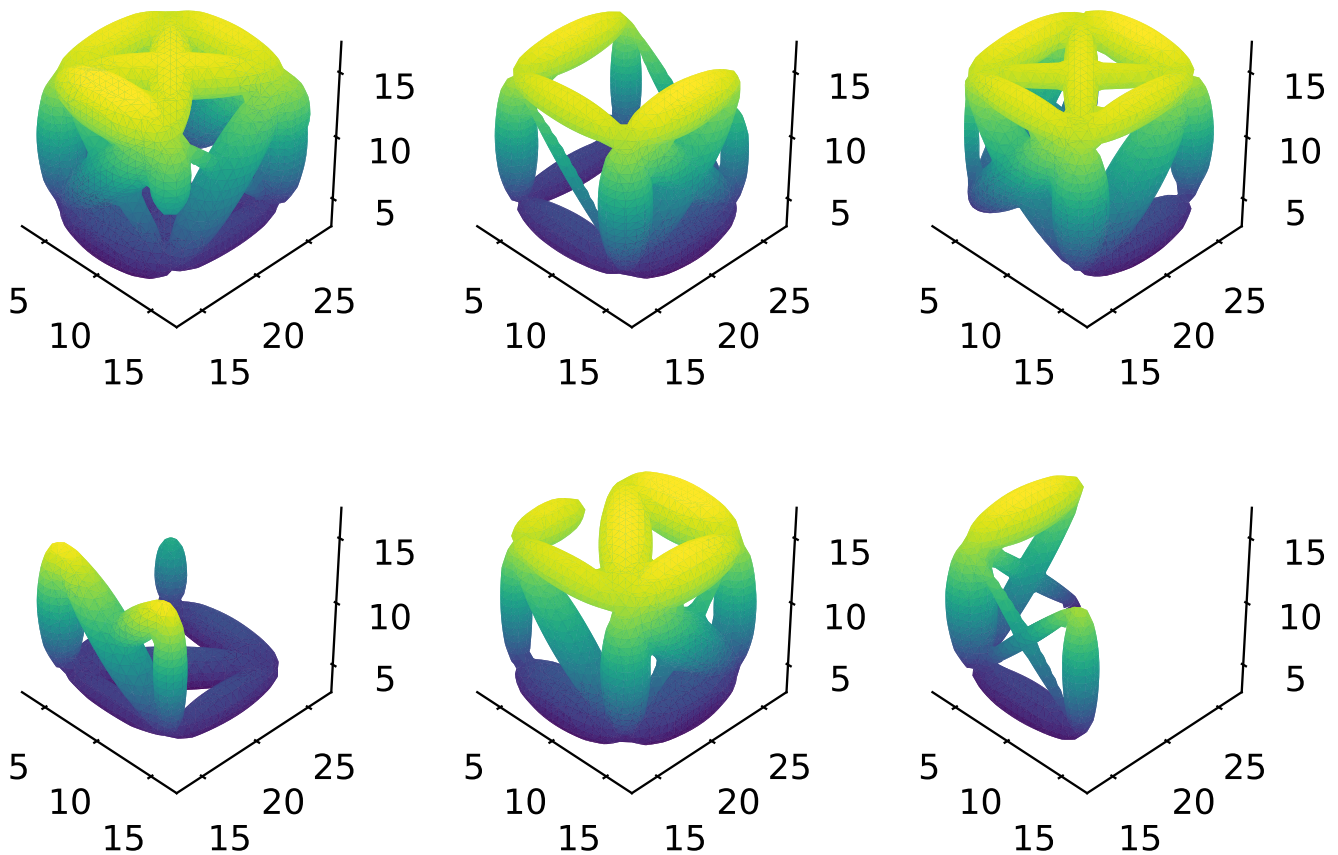


Fig. 12 Single cube truss dataset: Samples from 6 different subclasses are shown

resultant designs in a given subclass are similar to each other compared to designs from other subclasses.

- Vary the thickness of beams by sampling from a uniform distribution in  $[1.5, 4]$ . Note that the edge length of each cube in the base graph is 10 units.
- Remove a few randomly chosen beams. Sample a random variable  $s$  from uniform distribution  $U(0, 1)$ . Whenever  $s \leq 0.2$ , remove the beam.
- Shorten few randomly chosen beams ( $s \leq 0.2$ ). A cut beam can occupy a fraction between  $[0.5, 1]$  of the original length.

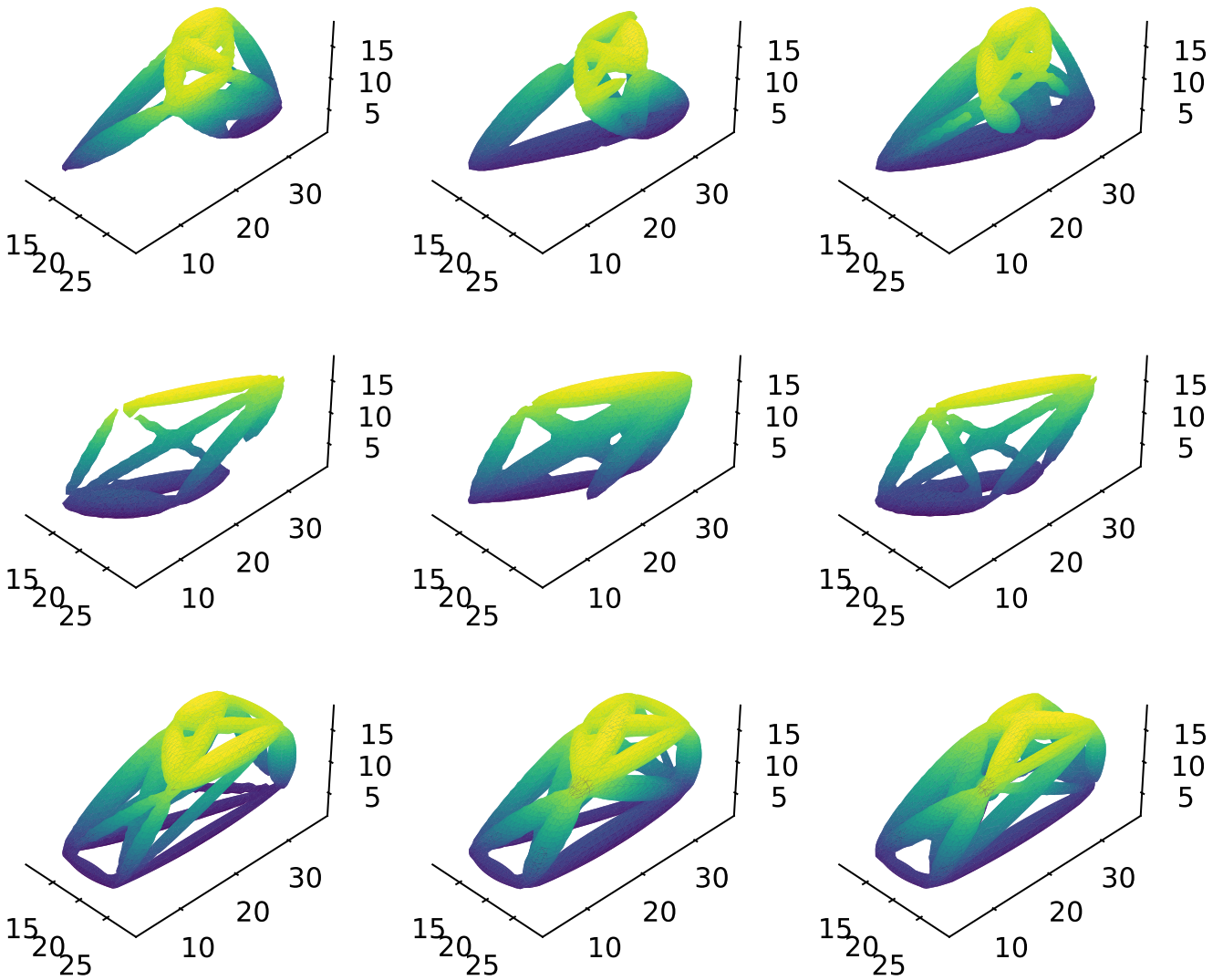
Variations in a subclass are controlled in a conservative way such that the designs still belong to the same subclass. All random values are picked using uniform distribution in the specified ranges. When not specified, the range is  $[0, 1]$ . Three samples of three of the subclasses are shown in Fig. 13.

### 6.3 Topologically optimized designs

This dataset comprises 1500 designs obtained using TO [1]. The design domain is restricted to a unit cubic domain with an optimization objective to minimize structural compliance. At the early stages of designs, the exact boundary conditions to be applied to the design may not be known. With the advances in high-performance computing, it is now possible to generate a large set of feasible design layouts for an engineering component by considering different design constraints. In our dataset (Fig. 14), we arbitrarily vary the position of the fixed boundary and the two loads, simulating an extreme use case where the boundary conditions are not known. In practice, only a fixed set of configurations for boundary conditions may be perturbed slightly for a given design task [10]. Nevertheless, we use this dataset to demonstrate our design exploration approach.

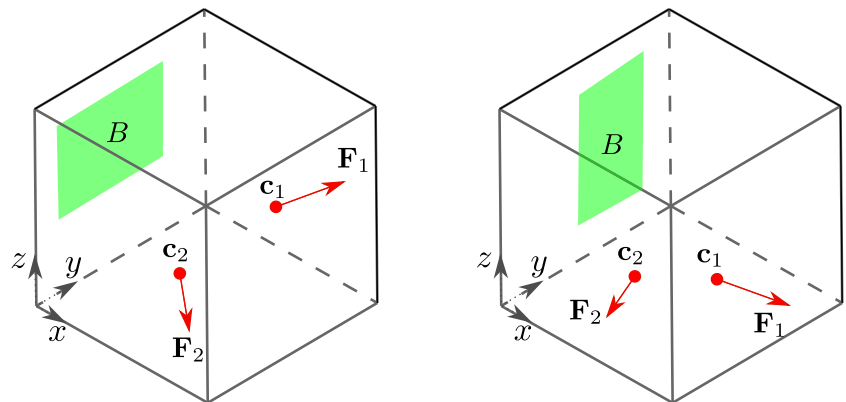
Different boundary conditions are generated as follows:

- We prescribe zero displacement for the nodes in an arbitrary patch in the fixed boundary face  $x = 0$  (green



**Fig. 13** Randomized topologies: Each row shows designs from a subclass. With low probability, beams are cut or removed from the underlying basegraph

**Fig. 14** TO in a unit cube: Two possible configurations of boundary conditions. The objective is to minimize structural compliance under radial loads  $F_1$  and  $F_2$  (centered around the points  $c_1$  and  $c_2$ ) given an arbitrary rectangular boundary patch  $B$  in a fixed face





area in Fig. 14). The rectangular patch is defined by its bounds along the  $y$ - and  $z$ -axes parallel to the face. The bound  $[b_{\min}, b_{\max}]$  along an axis is randomized such that  $0 \leq b_{\min} \leq b_{\max} \leq 1$  and  $b_{\min}, b_{\max} \in U(0, 1)$ .

- Two radial loads of unit magnitude are applied at randomly chosen centers within the cube. The load is distributed in a radius of 0.1 units. Each coordinate of the center is sampled from  $U(0, 1)$  such that the center is within the domain. The load vector is similarly calculated but its magnitude is normalized.

The density-based TO, SIMP [1], is used to optimize designs under static load for structural compliance, measured using internal energy stored in each element. A linear elastic material with the following properties is used: density ( $7.83 \cdot 10^{-9}$  ton/mm<sup>3</sup>), Young's modulus ( $2.07 \cdot 10^5$  MPa), and Poisson ratio (0.33).

The dataset has no prescribed subclasses or simple parameters that can capture geometrical differences, unlike the previous datasets. However, the dataset is representative of TO designs obtained in practice. Since the analysis of TO datasets is our intended application, we use this dataset to qualitatively evaluate a metric.

## 7 Evaluation on design datasets

In this section, we evaluate different metrics using the methods described in Section 5 on the datasets from Section 6.

### 7.1 Naming convention

A metric can be composed of other metrics starting with geometric representation and dimensionality reduction techniques

**Table 1** Abbreviations used in naming metrics. For example, a metric M: (Voxel, PCA, ED) means the voxel data is reduced using Principal Component Analysis (PCA). Euclidean distance (ED) in the reduced components gives the measured value ( $D_M$ )

Description	Abbreviation
MMC parameters	MMC par
Voxel representation	Voxel
Principal Component Analysis	PCA
Non-negative Matrix Factorization	NMF
t-distributed Stochastic Neighbor Embedding	t-SNE
Uniform Manifold Approximation and Projection	UMAP
Latent code of PCAE	AE code
Euclidean distance	ED
Chamfer distance	CD
Earth mover distance	EMD

if used. So, to avoid ambiguity, we choose the following naming convention using a tuple: (Representation, Dimensionality reduction techniques, Metric). Each component may use a short name for corresponding objects as in Table 1. The tuple represents the operation pipeline (metric pipeline) used to define the metric. For example, (Voxel, ED) means that Euclidean Distance (ED) is applied over voxel representation (Voxel).

### 7.2 Hyperparameters

For the voxel representation of each design, we use a common resolution of  $25 \times 25 \times 25$ . PCA and NMF use 10 components to reduce the voxel representation, which is found to be satisfactory for our datasets. As expected, t-SNE and UMAP use 2 components which enables the visualization of the dataset in 2D. Each design is represented with a pointcloud of 2048 points, which is used as the input to train the autoencoder. By default, the dimension of PCAE code is 128, except for the last dataset with TO designs, where 500 dimensions are needed to capture the increased complexity.

### 7.3 Metric correlations

For different datasets, we compare different metrics using the correlation coefficients ( $\rho$ ) as discussed in Section 5. Recall that values measured by a metric M are denoted by the variable  $D_M$ . We measure the correlation between distances measured by a given target metric (TM) with a reference metric (RM) which is known to be useful in capturing geometrical differences. Using the metric correlations, we discuss the deficiencies of a TM, if any.

**Beam-rotation dataset** The metric correlations for this dataset are shown in Fig. 15. RM is the Euclidean distance in MMC parameters which for any two designs in this dataset is the angular difference in their orientations. Figure 15a shows that the voxel distance correlates well with the RM. So, voxel distance agrees with our intuition of geometrical difference. Since the relation between the measured distances is nonlinear, the  $\rho_p$  is worse, as expected.

Similarly the TMs, CD and EMD, correlate well with the RM (Fig. 15b, c). However, for certain values of  $D_{RM}$ ,  $D_{TM}$  takes multiple values, which results in a vertical segment of dotted points (for both  $D_{CD}$ ,  $D_{EMD}$ ). The reason for this is that the designs for this dataset are obtained by rotating a reference ellipsoid in steps of a constant angle, i.e., the orientations of designs are equispaced. As a result, mul-

multiple pairs of designs have the same value for  $D_{RM}$  since within such pairs the relative location of geometries is the same. Yet,  $D_{TM}$  can take different values for the same pairs due to the deficiency of a TM. For the pointcloud based metrics such as CD and EMD, this effect occurs because the surface points are randomly sampled and may result in some variation in the relative location of points. The effect is observed to a lesser extent with the voxel distance as well (Fig. 15a). This is due to the discretization error of the voxel representation. Even if the relative angle between the two designs is held constant, the actual voxel representation depends on the absolute orientation in the design domain.

Figure 15d-h show TMs based on different dimensionality reduction techniques. The metric based on PCAE, the TM in Fig. 15d, correlates very well with the RM ( $\rho_s = 0.96$ ). The slight deterioration is also apparent from the spread of points along the y-axis. PCA reduction (Fig. 15e) with just 10 components is equally good, even if the original dimension of voxel representation  $\approx 1.5 \cdot 10^4$  is large. t-SNE, the TM in Fig. 15g, has the worst correlation. This is expected since it is designed only for the visualization of clusters, if any, in the high-dimensional data. However, UMAP in Fig. 15h, although designed for the same application as t-SNE, shows a much better correlation. Figure 15f shows NMF with slightly worse performance than PCA, possibly due to the additional non-negativity constraints on the reduced components. In summary, PCA and autoencoder-based metrics have a good correlation with the RM.

**Beam-elongation dataset** For this dataset, results are shown in Fig. 16. The RM measures the difference in lengths along the axis of elongation. The results are very similar to those with the beam-rotation dataset (Fig. 15).

**Beam-translation dataset** For this dataset, the results are shown in Fig. 17. The RM, Euclidean distance in MMC parameters, measures the difference in the location of design for this dataset. Figure 17a, with voxel distance as TM, shows a difference in the behavior compared to beam-rotation and beam-elongation datasets. The voxel distance does not change when the positional difference is more than a threshold ( $x = 4$ ). This is because the designs stop overlapping for this range and the sum of voxel differences ( $D_{TM}$ ) is constant (= total number of voxels in the two designs). So, voxel distance is disadvantageous as a metric for TO designs since the position of the non-overlapping material in the design domain of TO is relevant. Other

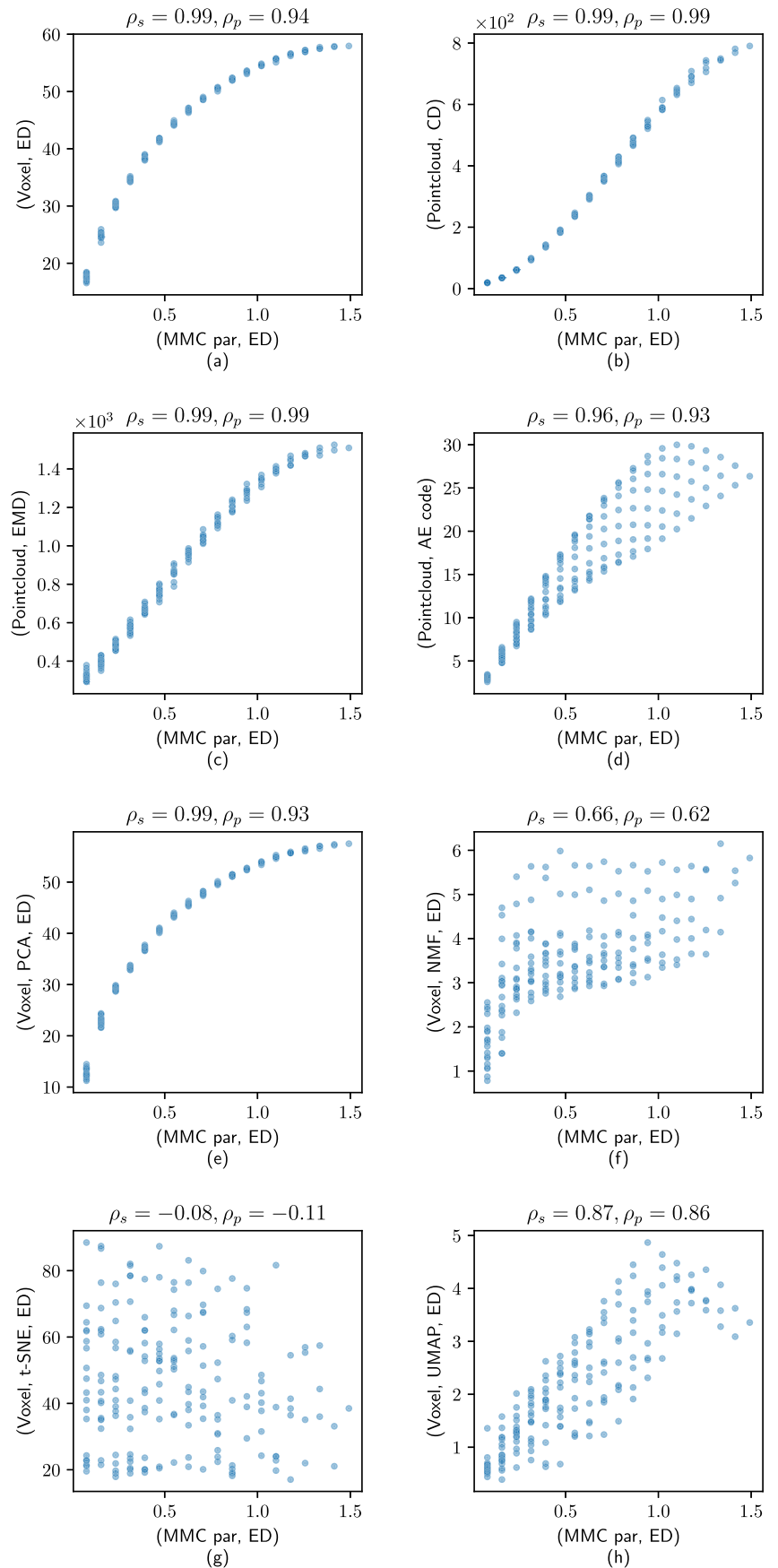
metric comparisons are very similar to those obtained with the beam-rotation dataset (Fig. 15).

The three datasets discussed here use an MMC beam with a form factor  $m = 2$ . It is interesting to see if using other MMC shapes similar to bipyramids ( $m = 1$ ) or cuboids ( $m = 6$ ) affect the results. So, we repeated each of the experiments above with  $m$  ranging from 1 to 6. For a given  $m$  value, the results are reasonably similar for a given transformation type. The correlation between CD and EMD is very high ( $\rho_s \geq 0.99$ ). The autoencoder-based metric also agrees well with CD ( $\rho_s \in [0.96, 1]$ ). As expected, voxel distance is similar to CD ( $\rho_s \in [0.98, 1]$ ) for these datasets. Other than t-SNE, using dimensionality reduction techniques on voxel representation doesn't significantly reduce the metric correlation with CD ( $\rho_s \in [0.95, 1]$ ). With t-SNE, the metric correlation is quite low ( $\rho_s \in [0, 0.3]$ ) even for the simple datasets described here. Given the consistency of results with different MMC form factors, we expect similar results with any other shapes.

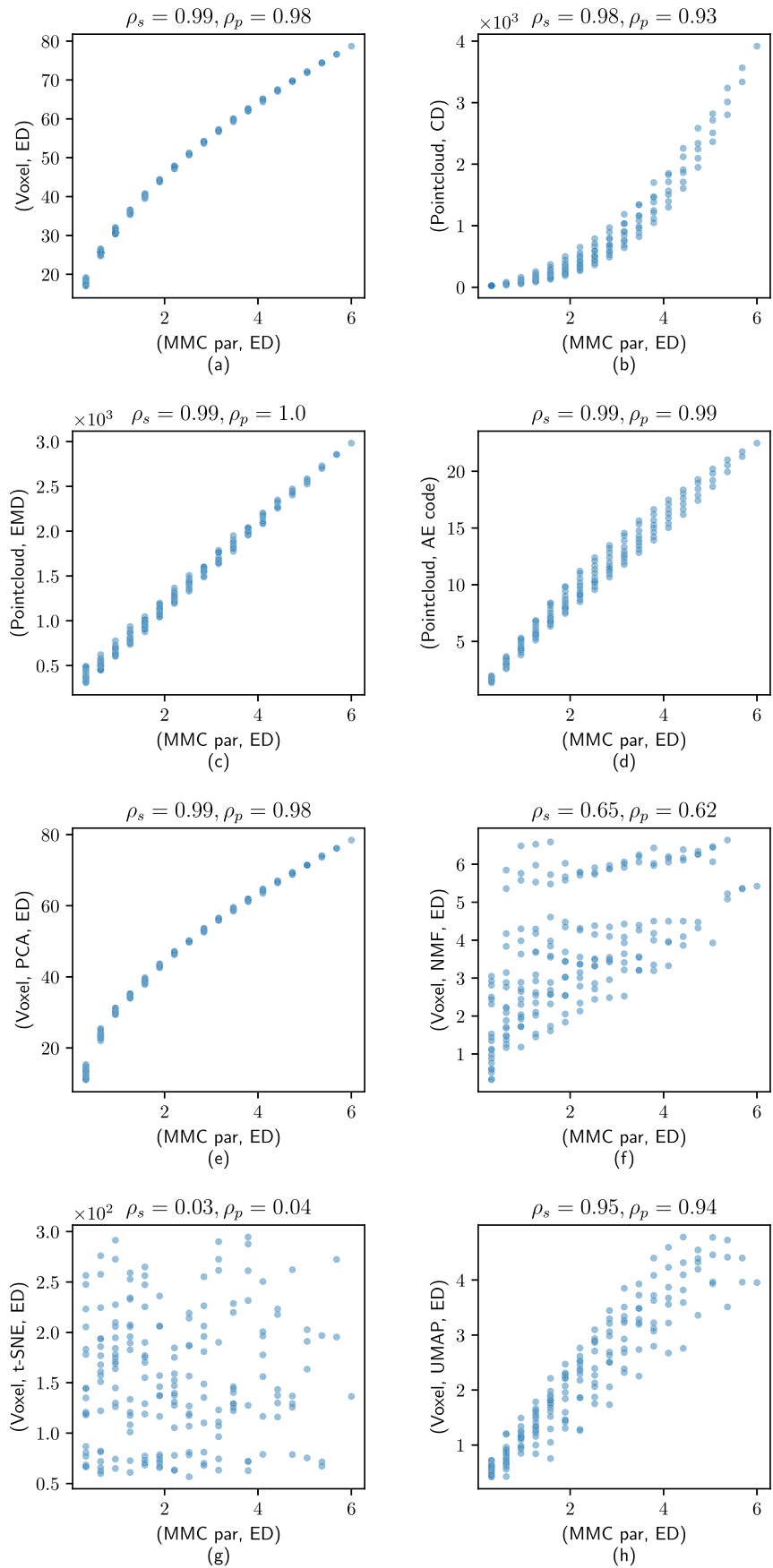
**Single cube truss dataset** For the datasets based on single cube, three cubes, and random topologies, the geometrical differences cannot be captured by differences in simple shape parameters. So, we compare metrics with chamfer distance (Pointcloud, CD) as reference (RM), which is popular in 3D object recognition [27]. Metric correlation study indicates that the voxel distance has almost no correlation ( $\rho_s = 0.14$ ) with RM while EMD is very similar to CD even for complex datasets. Since CD is cheaper to compute, one may prefer it to EMD. The latent code of PCAE (AE code) used in this study is trained with chamfer distance (CD) as the loss function and hence shows a very high correlation with it ( $\rho_s = 0.96$ ). Other metrics based on dimensionality reduction of voxel data show improvement in accuracy  $\rho_s \in [0.4, 0.6]$  compared to voxel distance  $\rho_s = 0.14$ . So, for complex designs, it is beneficial to reduce the dimensions by removing redundant features.

**Three cube truss dataset** The metric correlations for this dataset are similar to the results obtained for the single cube truss dataset. Voxel distance has a very low correlation with CD ( $\rho_s = 0.55$ ) which is improved by using dimensionality techniques ( $\rho_s \in [0.6, 0.8]$ ). The highest improvement is obtained with Autoencoder ( $\rho_s = 0.93$ ) using the AE code. The correlations are higher than the single cube dataset where the design changes are more complex.

**Fig. 15** Metric correlations for the beam-rotation dataset: Voxel distance and Pointcloud based metrics correlate well with differences in orientation

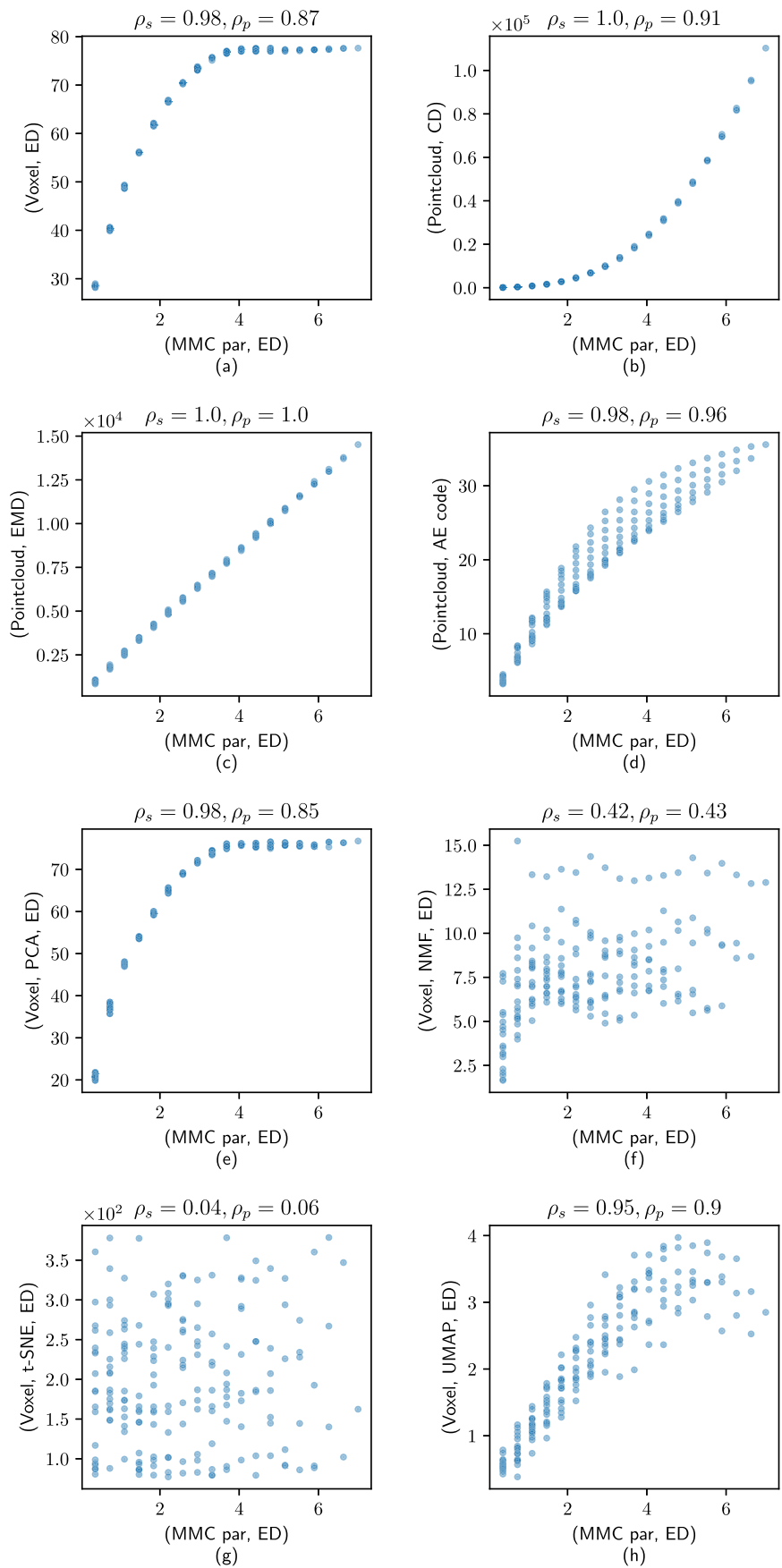


**Fig. 16** Metric correlations for the beam-elongation dataset: Voxel distances and Pointcloud based metrics correlate well with differences in design length

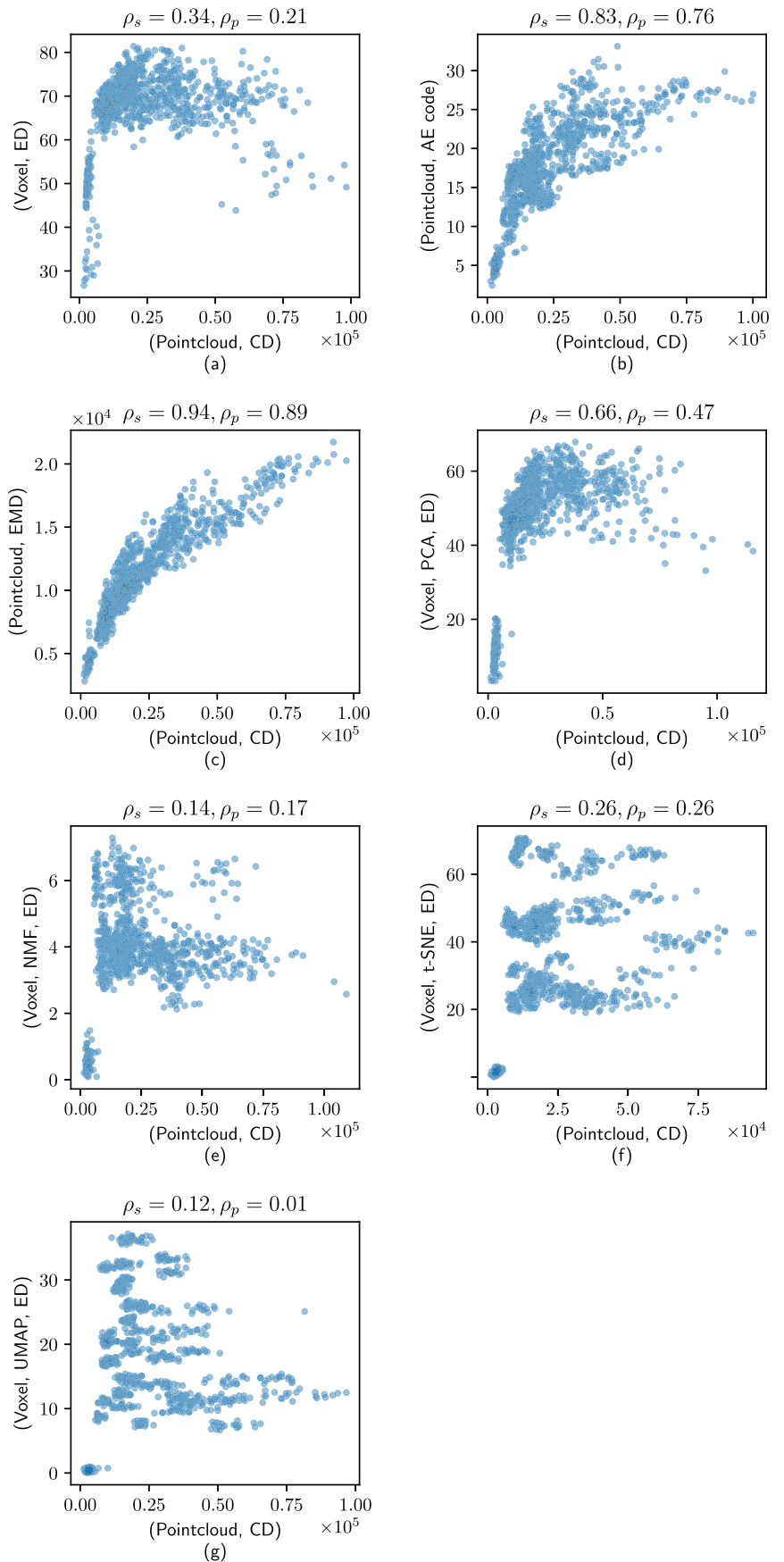




**Fig. 17** Metric correlation for beam-translation dataset: Pointcloud based metrics correlate well with the differences in position



**Fig. 18** Metric correlations for random topologies dataset with 1000 designs. Each graph consists of nearly  $5 \times 10^5$  pairwise distance measures



**Random topologies** This dataset has the most diverse topologies among the synthetic datasets. Figure 18 shows the correlations between the metrics. As expected, voxel distance doesn't correlate well with CD. EMD is similar to CD ( $\rho_s = 0.94$ ) even for the complex topologies. It is interesting to see that the dimensionality reduction of voxels doesn't result in any improvement in correlation with CD whereas the dimensionality reduction of point clouds using PCAE (AE code) is very useful ( $\rho_s = 0.83$ ). The results indicate that AE code has successfully learned the loss function, CD, it is trained on.

**Summary** The metric comparisons on datasets obtained by rotation, translation, and scaling show the deficiencies of the voxel distance and the metrics based on dimensionality reduction techniques. CD and EMD outperform these metrics in capturing simple geometric differences. Of the dimensionality reduction techniques, Euclidean distance in the AE code shows the best correlation with RM. From the results on metric correlations, CD, EMD, and Euclidean distance in AE code are the top choices for the metrics on geometry. Since the test datasets have complex topologies with strong variations in size and orientation, we expect similar results with TO designs.

#### 7.4 Using clustering performance

To evaluate the classification accuracy of different metrics, we consider the randomized topologies dataset of 50 subclasses with strong variations in topology, where each subclass has 20 designs. Metrics mentioned in Section 3 are used with the naming convention from Table 1. For example, the metric: (Voxel, PCA, ED) means that the voxel representation of the geometries is transformed using PCA into a low-dimensional vector. Finally, the Euclidean distance (ED) in the reduced space is the resultant metric. Different metrics are used to cluster the topologies and the corresponding accuracy measures are reported.

Table 2 shows the classification accuracy of different metrics using  $k$ -means clustering with a prescribed number of clusters ( $k = 50$ ), the same as the number of class labels. Note that since this is a labeled test set, the class labels are known beforehand. PCA (with 10 components) is better compared to NMF (with 10 components) for this dataset. The pointcloud metric (AE code, ED), UMAP, and t-SNE based metrics can identify all the subclasses accurately. The high precision of UMAP and t-SNE indicates that they can be used to visualize the clusters in the dataset (Fig. 19). Note that CD and EMD metrics are not used

since  $k$ -means requires an input feature vector and cannot handle pairwise distance matrices provided by CD and EMD.

Table 3 shows the classification accuracy of different metrics using OPTICS, which is more general than the  $k$ -means algorithm and can handle pairwise distance matrices. OPTICS identifies the appropriate number of clusters automatically. However, for this dataset, the clustering accuracy is lower compared to the  $k$ -means method. The order of the derived metrics using NMF, Voxels (Voxel, ED), PCA, UMAP, AE code is the same as in Table 2. In contrast, t-SNE performs worse than PCA-based metric, which is outperformed by UMAP. CD and EMD perform better than UMAP, but Autoencoder latent code results in the best metric for classification and is better than the loss function (CD) it is trained on.

## 8 Engineering application

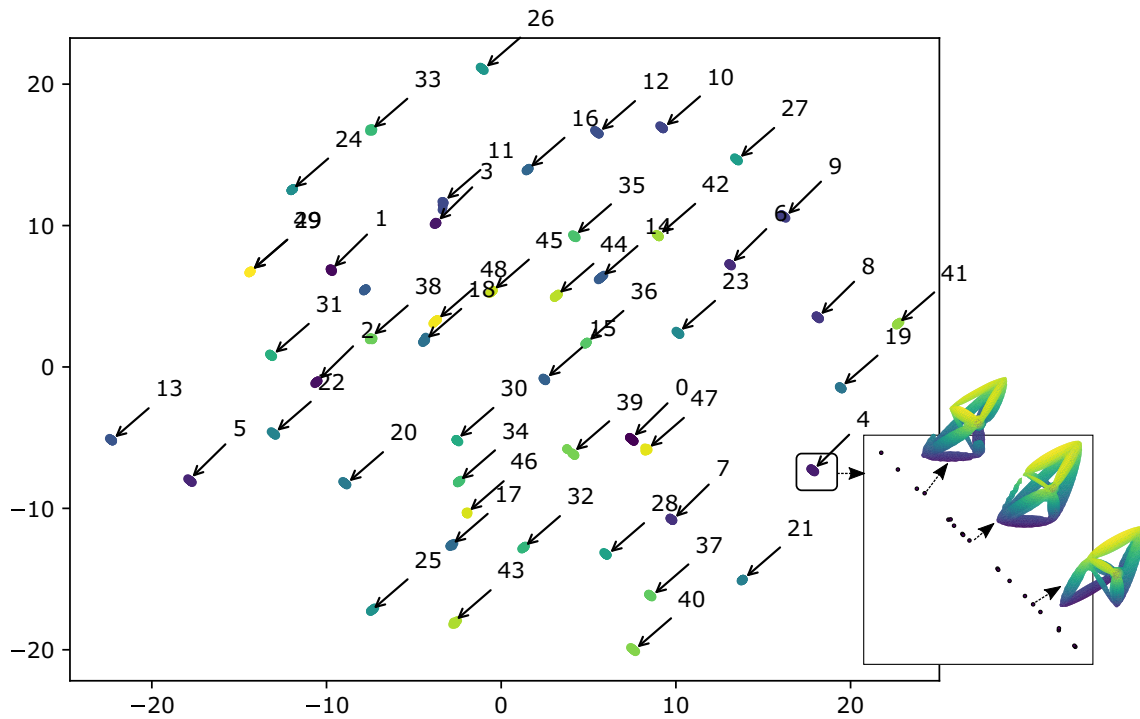
From our experiments discussed in Section 7, we find that the AE code obtained by PCAE reduction can identify similar designs in a topologically optimized, complex dataset. It also extracts *geometric features*, which are easy to cluster since they are low-dimensional.

Figure 20 shows the clusters in designs as visualized in UMAP with clusters identified using  $k$ -means algorithm with AE code as the input data. Since UMAP preserves the intra-cluster distances, the relative location in the 2D cluster plot is meaningful.

Figure 21 compares each of the cluster representatives with three different designs: two designs closest to it and one design farthest from it in terms of Euclidean distance in AE code. The most similar designs seem to share similar material distribution while the load conditions seem to be different. The most dissimilar designs (column d) seem to have completely different topology compared to the

**Table 2** Classification accuracy of different metrics with  $k$ -means clustering ( $k = 50$ )

Metric	Precision	F1 score	AMI score
(Voxel, NMF, ED)	0.88	0.86	0.92
(Voxel, ED)	0.96	0.95	0.98
(Voxel, PCA, ED)	0.97	0.96	0.98
(Voxel, UMAP, ED)	1.00	1.00	1.00
(Voxel, t-SNE, ED)	1.00	1.00	1.00
(AE code, ED)	1.00	1.00	1.00



**Fig. 19** Visualizing design clusters in the AE code data using UMAP. Each cluster is a set of 2D points with each point representing a design. The dataset contains 50 clusters, numbered from 0 to 49. The inset shows a selected cluster and three sample designs in it

representative. The figure also shows the relative distances in terms of AE code and CD from each design (column b-d) to the representative (column a). For a given metric, a measured distance is normalized using the farthest distance measured from the representative to yield the relative distance. According to these metrics, the closest designs to each representative are only dissimilar by 0.2 to 0.3 units compared to the farthest design, whose distance is normalized to 1 unit. As expected, AE code mimics the

chamfer distance quite closely even for the complex TO dataset.

We recommend using the PCAE followed by a UMAP reduction for intuitive and accurate visualization of design clusters. This is an improvement to the design exploration method proposed by us for Pareto-optimal designs using voxel representation and PCA [52]. Furthermore, the *geometric features* obtained by PCAE can be combined with performance measures. The aggregate vector can be used directly for design exploration based on geometry and performance. Since the Euclidean distance in the *geometric features* is meaningful, the similarity control in TO [20] can be easily implemented as well.

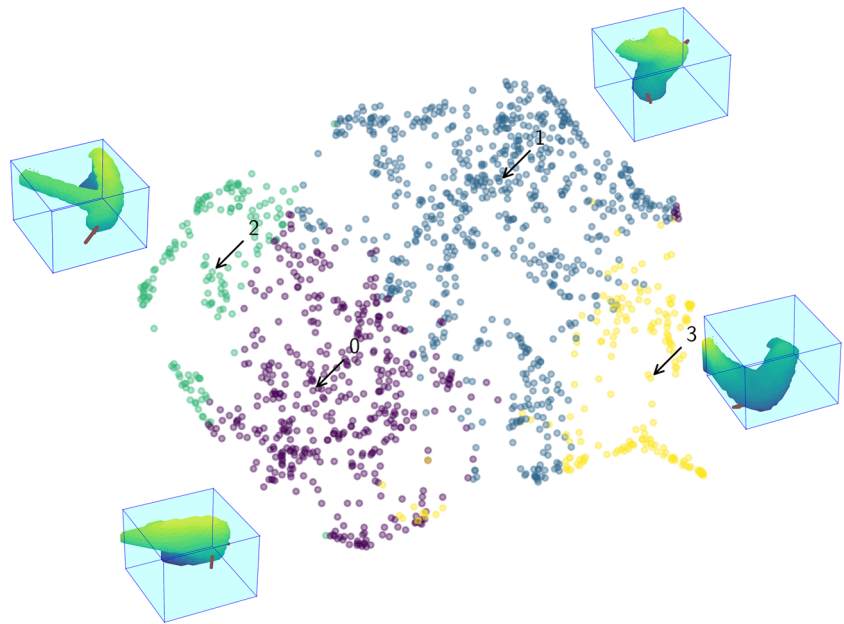
Table 4 summarizes the advantages and disadvantages of using different metrics. In terms of clustering accuracy, the pointcloud based metrics perform the best. t-SNE and UMAP perform well even with raw voxel data, although it is better to use them with the reduced data. By the high quality of *geometric features*, we mean that the Euclidean distance in the extracted features, if any, is meaningful. The latent code of the PCAE, which gives the most meaningful *geometric features*, can be thought of as a means to improve the pairwise distance metrics (CD in

**Table 3** Classification accuracy of different metrics with OPTICS as the clustering method

Metric	Precision	F1 score	AMI score
(Voxel, NMF, ED)	0.60	0.68	0.61
(Voxel, ED)	0.66	0.64	0.80
(Voxel, t-SNE, ED)	0.69	0.77	0.67
(Voxel, PCA, ED)	0.72	0.78	0.75
(Voxel, UMAP, ED)	0.74	0.82	0.68
(Pointcloud, CD)	0.86	0.88	0.90
(Pointclouds, EMD)	0.86	0.88	0.90
(AE code, ED)	0.94	0.95	0.95



**Fig. 20** Clusters in TO designs are visualized in 2D using UMAP. The  $k$ -means clustering algorithm identifies the 4 clusters, labeled 0-3, based on the AE code. The representative design for each cluster is shown as well.  $k$ -means clustering identifies the subclasses of designs even if the clusters are not separated, which is often the case with engineering data



our case) they are trained on, which do not extract any features. From computation ease, voxel distance is simple to calculate relative to the other metrics based on dimensionality reduction such as PCA or AE code since there are no additional data processing steps. CD and EMD are moderately expensive since they involve some analysis of pointclouds before comparing. This analysis is only valid if the metrics are not used often for a specific dataset. With repeated calculations, the reduced dimensional representation significantly speeds up the metric calculation, once the initial data processing is complete. From clustering effort, the metrics based on simple feature vectors are easier to compute. CD and EMD which require the construction of pairwise distance matrices can be very expensive. For exploring TO designs, the AE code might be a good compromise. It has high clustering accuracy with the extraction of meaningful *geometric features*. The moderate effort needed for training a PCAE model might be worth the benefits.

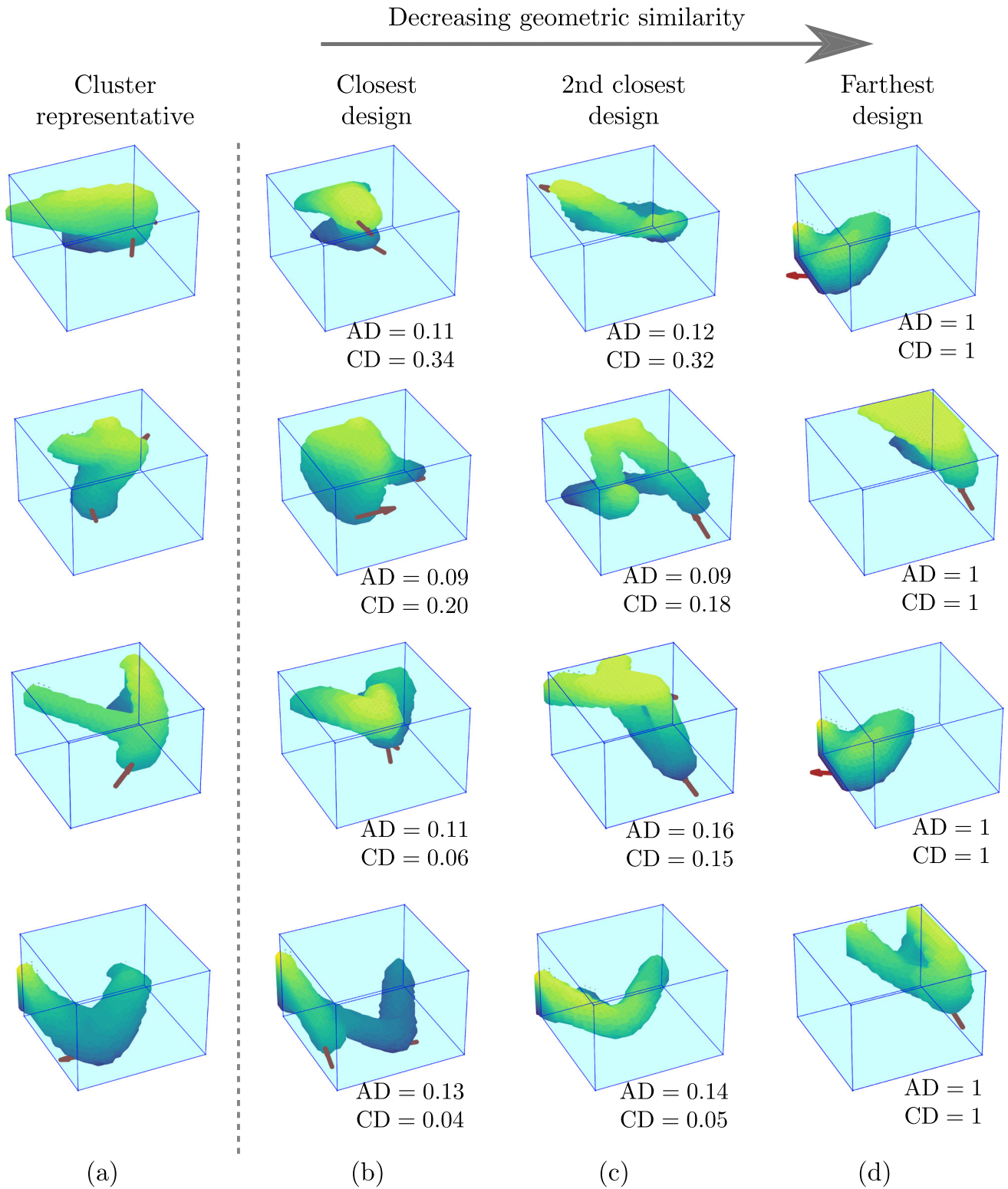
## 9 Conclusion

In this work, we address the problem of finding geometric features that can be used to explore topologically optimized structures and identify diverse designs based on geometrical properties such as topology, size, shape, and orientation in a design space. Extracting features in an automated fashion

using popular unsupervised machine learning techniques improves the usability of topology optimization (TO) and other design generation methods in the engineering design process.

The challenge is to choose a feature extraction method from several methods available: PCA (Principal Component Analysis), NMF (Non-negative Matrix Factorization), t-SNE (t-distributed Stochastic Neighbor Embedding), UMAP (Uniform Manifold Approximation and Projection), or PCAE (Pointcloud Autoencoder), a geometric deep learning method for pointcloud representation. We propose to choose the geometric features based on two properties: the ability to capture geometrical differences and to identify similar designs. The former property is evaluated using a novel method to compare with reference distance metrics, while the latter is measured using state-of-the-art methods for evaluating clustering performance. The proposed test datasets comprise topologically more complex designs than publicly available datasets such as ShapeNet [28].

For clustering geometrically similar designs, PCA can extract better features compared to the voxel representation. t-SNE and UMAP, which are visualization techniques for high-dimensional data, as expected, perform even better. The performance improvement of t-SNE and UMAP comes at a cost that the distances in the reduced representation may not reflect the geometric differences, as shown in our experiments with even simple datasets. PCAE,



**Fig. 21** TO designs similar to cluster representatives found using AE code (Fig. 20). The first column (a) contains the four design representatives. For each representative, three designs are shown in a row with decreasing similarity along the columns b-d. For each design, we show

the corresponding loads as brown arrows. In each row, we measure the relative distance from the representative (a) to each design in columns b-d using AE code (AD) and chamfer distance (CD)

**Table 4** Advantages and disadvantages of different metrics. For each attribute column, values are put into three categories: low (L), medium (M), and high (H). Since the actual values depend on the computational

hardware, this categorization may be more useful. Since the metrics CD and EMD do not yield or use *geometric features*, they are ranked low in this attribute (third column)

Metric	Clustering accuracy	Quality of geometric features	Metric calculation ease	Clustering ease
(Voxel)	L	L	H	H
(Voxel, NMF)	M	M	M	H
(Voxel, PCA)	M	M	M	H
(Voxel, t-SNE)	H	L	M	H
(Voxel, UMAP)	H	M	M	H
(Pointcloud, CD)	H	L	L	L
(Pointcloud, EMD)	H	L	L	L
(AE code, ED)	H	H	M	M

in contrast, performs well in both scenarios: measuring similar distances as reference metrics and identifying geometrically similar designs. Empirically, we observe that the PCAE creates a feature vector with distances correlated to the loss function it is trained on. This allows conversion of any reasonable metric, which is defined as a pairwise function of geometries, to a Euclidean distance in a feature vector (latent code). Since the latent code is better suited for integrating geometric features with performance features, one can simultaneously explore designs based on geometry and performance. This analysis shows the need for our proposed method, along with the clustering performance measures, to validate if the *geometric features* are as meaningful as the reference metrics.

With an effective geometric feature vector such as the latent code of a PCAE, we can now combine *geometric features* with performance features for design exploration. It would be interesting to see if this can discover design clusters with unique properties, either geometric- or performance-wise, in real-world engineering applications. Furthermore, one can identify distinct concepts in a design database. This would considerably improve the applicability of topology optimization in the product development process.

**Acknowledgements** Nivesh Dommaraju developed the data and software used in this project with the help of Mariusz Bujny.

**Author Contributions** Fabian Duddeck, Stefan Menzel, and Markus Olhofer are the advisors for the doctoral project of the first author Nivesh Dommaraju, who is grateful for their support and research insights for this work.

**Funding** Open Access funding enabled and organized by Projekt DEAL. This project has received funding from Honda Research Institute Europe GmbH, Germany.

**Availability of Data and Material** Data and material are owned by Technical University of Munich and Honda Research Institute Europe

GmbH. Most of the software used can be replicated by using the open-source libraries on clustering and deep learning. The datasets discussed in this paper are available at zenodo [53].

## Declarations

**Competing interests** No competing interest exists.

**Open Access** This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

## References

1. Bendsøe MP, Sigmund O (2004) Topology optimization. Springer, Berlin. <https://doi.org/10.1007/978-3-662-05086-6>
2. Liu K, Tovar A (2014) An efficient 3D topology optimization code written in Matlab. Struct Multidiscipl Optim 50(6):1175–1196. <https://doi.org/10.1007/s00158-014-1107-x>
3. Bujny M, Aulig N, Olhofer M, Duddeck F (2017) Identification of optimal topologies for crashworthiness with the evolutionary level set method. Int J Crashworthiness 23(4):395–416. <https://doi.org/10.1080/13588265.2017.1331493>
4. Raponi E, Bujny M, Olhofer M, Aulig N, Boria S, Duddeck F (2019) Kriging-assisted topology optimization of crash structures. Comput Methods Appl Mech Eng 348:730–752. <https://doi.org/10.1016/j.cma.2019.02.002>
5. Zeng D, Duddeck F (2017) Improved hybrid cellular automata for crashworthiness optimization of thin-walled structures. Struct Multidiscipl Optim 56(1):101–115. <https://doi.org/10.1007/s00158-017-1650-3>
6. Duddeck F, Hunkeler S, Lozano P, Wehrle E, Zeng D (2016) Topology optimization for crashworthiness of thin-walled structures under axial impact using hybrid cellular automata.

- Struct Multidiscipl Optim 54(3):415–428. <https://doi.org/10.1007/s00158-016-1445-y>
7. Borrvall T, Petersson J (2002) Topology optimization of fluids in Stokes flow. *Int J Numer Methods Fluids* 41(1):77–107. <https://doi.org/10.1002/flid.426>
  8. Sanogo S, Messine F (2018) Topology optimization in electromagnetism using SIMP method. *COMPEL - The International Journal for Computation and Mathematics in Electrical and Electronic Engineering* 37(6):2138–2157. <https://doi.org/10.1108/compe-04-2017-0170>
  9. Yoon GH, Jensen JS, Sigmund O (2007) Topology optimization of acoustic-structure interaction problems using a mixed finite element formulation. *Int J Numer Methods Eng* 70(9):1049–1075. <https://doi.org/10.1002/nme.1900>
  10. Matejka J, Glueck M, Bradner E, Hashemi A, Grossman T, Fitzmaurice G (2018) Dream Lens. In: Proceedings of the 2018 CHI conference on human factors in computing systems. ACM, pp 1–12. <https://doi.org/10.1145/3173574.3173943>
  11. Ramnath S, Aulig N, Bujny M, Menzel S, Gandikota I, Horner K (2019) Load Case Preference Patterns based on Parameterized Pareto-Optimal Vehicle Design Concept Optimization. In: 12th European LS-DYNA Conference, pp 1–9
  12. Fender G, Marburg S, Duddeck F (2016) Identification of a set of candidate solutions for optimal positioning of damping layers. *SAE Int J Passen Cars Mechan Syst* 9(3):987–994. <https://doi.org/10.4271/2016-01-1778>
  13. Sigmund O, Petersson J (1998) Numerical instabilities in topology optimization: a survey on procedures dealing with checkerboards, mesh-dependencies and local minima. *Struct Optim* 16(1):68–75. <https://doi.org/10.1007/BF01214002>
  14. Ferreira JC, Fonseca CM, Gaspar-Cunha A (2007) Methodology to select solutions from the pareto-optimal set: a comparative study. In: Proceedings of the 9th annual conference on genetic and evolutionary computation - GECCO'07. ACM Press, pp 789–796. <https://doi.org/10.1145/1276958.1277117>
  15. Singh HK, Ray T, Rodemann T, Olhofer M (2019) Identifying solutions of interest for practical many-objective problems using recursive expected marginal utility. In: Proceedings of the genetic and evolutionary computation conference companion. ACM, pp 1734–1741. <https://doi.org/10.1145/3319619.3326804>
  16. Cvetkovic D, Parmee IC (2002) Preferences and their application in evolutionary multiobjective optimization. *IEEE Trans Evolution Comput* 6(1):42–57. <https://doi.org/10.1109/4235.985691>
  17. Hagg A, Asteroth A, Bäck T (2018) Prototype Discovery Using Quality-Diversity. In: Parallel Problem Solving from Nature – PPSN XV. Springer International Publishing, Berlin, pp 500–511. [https://doi.org/10.1007/978-3-319-99253-2\\_40](https://doi.org/10.1007/978-3-319-99253-2_40)
  18. Dommaraju N, Bujny M, Menzel S, Olhofer M, Duddeck F (2019) Identifying lAutoencoder Networks. In: 2019 International Conference on Data Mining Workshops (ICDMW). IEEE, pp 761–768. <https://doi.org/10.1109/icdmw.2019.00113>
  19. Oh S, Jung Y, Kim S, Lee I, Kang N (2019) Deep generative design: Integration of topology optimization and generative models. *Journal of Mechanical Design*, 141(11). <https://doi.org/10.1115/1.4044229>
  20. Yousaf MS, Bujny M, Zurbrugg N, Detwiler D, Duddeck F (2021) Similarity control in topology optimization under static and crash loading scenarios. *Eng Optim* 53(9):1523–1538. <https://doi.org/10.1080/0305215x.2020.1806257>
  21. López GL, Negrón APP, Jiménez ADA, Rodríguez JR (2016) Paredes RI. Comparative analysis of shape descriptors for 3D objects. *Multimed Tools Appl* 76(5):6993–7040. <https://doi.org/10.1007/s11042-016-3330-5>
  22. Feldman D, Schmidt M, Sohler C (2020) Turning Big Data Into Tiny Data: Constant-size Coresets for  $k$ -Means, PCA, and Projective Clustering. *SIAM J Comput* 49(3):601–657. <https://doi.org/10.1137/18m1209854>
  23. Bozakov Z, Graening L, Hasler S, Wersing H, Menzel S (2008) Unsupervised extraction of design components for a 3D parts-based representation. In: 2008 IEEE international joint conference on neural networks (IEEE world congress on computational intelligence). IEEE, pp 2009–2016. <https://doi.org/10.1109/ijcnn.2008.4634074>
  24. van der Maaten L, Hinton G (2008) Visualizing data using t-SNE. *J Mach Learn Res* 9(86):2579–2625
  25. McInnes L, Healy J, Saul N, Großberger L (2018) UMAP: Uniform Manifold Approximation and Projection. *J Open Source Softw* 3(29):861. <https://doi.org/10.21105/joss.00861>
  26. Charles RQ, Su H, Kaichun M, Guibas LJ (2017) PointNet: deep learning on point sets for 3D classification and segmentation. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). IEEE, pp 77–85. <https://doi.org/10.1109/cvpr.2017.16>
  27. Achlioptas P, Diamanti O, Mitliagkas I, Guibas L (2018) Learning Representations and Generative Models for 3D Point Clouds. In: proceedings of the 35th international conference on machine learning, vol 80. PMLR, pp 40–49
  28. Chang AX, Funkhouser T, Guibas L, Hanrahan P, Huang Q, Li Z et al (2015) ShapeNet: An Information-Rich 3D Model Repository. Stanford University — Princeton University — Toyota Technological Institute at Chicago. arXiv:1512.03012 [cs.GR]
  29. Wang PS, Liu Y, Guo YX, Sun CY, Tong X (2017) O-CNN Octree-Based convolutional neural networks for 3D shape analysis. *ACM Trans Graph* 36(4):1–11. <https://doi.org/10.1145/3072959.3073608>
  30. Ioannidou A, Chatzilari E, Nikolopoulos S, Kompatsiaris I (2017) Deep learning advances in computer vision with 3D data. *ACM Comput Surv* 50(2):1–38. <https://doi.org/10.1145/3042064>
  31. Peyrè G, Transport CuturiM. (2019) Computational optimal with applications to data science. *Foundations and Trends®*, in Machine Learning. 11(5-6):355–607. <https://doi.org/10.1561/2200000073>
  32. Ulu E, Zhang R, Kara LB (2015) A data-driven investigation and estimation of optimal topologies under variable loading configurations. *Comput Methods Biomechan Biomed Eng Imaging Visual* 4(2):61–72. <https://doi.org/10.1080/21681163.2015.1030775>
  33. Jolliffe IT (2002) Principal component analysis. Springer, Berlin. <https://doi.org/10.1007/b98835>
  34. Lee DD, Seung HS (1999) Learning the parts of objects by non-negative matrix factorization. *Nature* 401(6755):788–791. <https://doi.org/10.1038/44565>
  35. Umetani N (2017) Exploring generative 3D shapes using autoencoder networks. In: SIGGRAPH Asia 2017 Technical Briefs on - SA '17. 24. ACM Press, pp 1–4. <https://doi.org/10.1145/3145749.3145758>
  36. Yang Y, Feng C, Shen Y, Tian D (2018) FoldingNet: Point Cloud Auto-Encoder via Deep Grid Deformation. In: 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition. IEEE, pp 206–215. <https://doi.org/10.1109/cvpr.2018.00029>
  37. Rios T, Van SteinB, Menzel S, Back T, Sendhoff B, Wollstadt P (2020) Feature Visualization for 3D Point Cloud Autoencoders. In: Proceedings of the international joint conference on neural networks, pp 1–9. <https://doi.org/10.1109/IJCNN48605.2020.9207326>



38. Schober P, Boer C, Schwarte LA (2018) Correlation coefficients: appropriate use and interpretation. *Anesthesia Analgesia* 126(5):1763–1768. <https://doi.org/10.1213/ane.00000000000002864>
39. Godbole S, Sarawagi S (2004) Discriminative Methods for Multi-labeled Classification. In: *Advances in knowledge discovery and data mining*. Springer, Berlin, pp 22–30. [https://doi.org/10.1007/978-3-540-24775-3\\_5](https://doi.org/10.1007/978-3-540-24775-3_5)
40. Vinh NX, Epps J, Bailey J (2010) Information theoretic measures for clusterings comparison: variants, properties, normalization and correction for chance. *J Machine Learn Res* 11:2837–2854
41. Zhang W, Zhou J, Zhu Y, Guo X (2017) Structural complexity control in topology optimization via moving morphable component (MMC) approach. *Struct Multidiscipl Optim* 56(3):535–552. <https://doi.org/10.1007/s00158-017-1736-y>
42. Lewiner T, Lopes H, Vieira AW, Tavares G (2003) Efficient implementation of marching cubes' cases with topological guarantees. *J Graph Tools* 8(2):1–15. <https://doi.org/10.1080/10867651.2003.10487582>
43. van der Walt S, Schönberger JL, Nunez-Iglesias J, Boulogne F, Warner JD, Yager N et al (2014) scikit-image: image processing in Python. *PeerJ* e453:2. <https://doi.org/10.7717/peerj.453>
44. Dawson-Haggerty et al (2019) Trimesh (3.2.0). Available from: <http://trimsh.org>
45. Weisstein EW (1999) Triangle point picking. *MathWorld—A Wolfram Web Resource*. Available from: <https://mathworld.wolfram.com/TrianglePointPicking.html>
46. Zhang W, Yuan J, Zhang J, Guo X (2016) A new topology optimization approach based on Moving Morphable Components (MMC) and the ersatz material model. *Struct Multidiscipl Optim* 53(6):1243–1260. <https://doi.org/10.1007/s00158-015-1372-3>
47. Lei X, Liu C, Du Z, Zhang W, Guo X (2018) Machine Learning-Driven Real-Time Topology Optimization Under Moving Morphable Component-Based Framework. *Journal of Applied Mechanics*, 86(1). <https://doi.org/10.1115/1.4041319>
48. Bai J, Zuo W (2020) Hollow structural design in topology optimization via moving morphable component method. *Struct Multidiscipl Optim* 61(1):187–205. <https://doi.org/10.1007/s00158-019-02353-0>
49. Wein F, Dunning PD, Norato JA (2020) A review on feature-mapping methods for structural optimization. *Struct Multidiscipl Optim* 62(4):1597–1638. <https://doi.org/10.1007/s00158-020-02649-6>
50. Bujny M (2020) Level set topology optimization for crashworthiness using evolutionary algorithms and machine learning. *Technical University of Munich, Munich*. Available from: <https://nbn-resolving.org/urn/resolver.pl?urn:nbn:de:bvb:91-diss-20201009-1540709-1-0>
51. Bujny M, Olhofer M, Aulig N, Duddeck F (2021) Topology optimization of 3D-printed joints under crash loads using evolutionary algorithms. *Struct Multidiscipl Optim* 64(6):4181–4206. <https://doi.org/10.1007/s00158-021-03053-4>
52. Dommaraju N, Bujny M, Menzel S, Olhofer M, Duddeck F (2020) Simultaneous Exploration of Geometric Features and Performance in Design Optimization. In: *16th International LS-DYNA Conference*, p 12
53. Dommaraju N, Bujny M, Menzel S, Olhofer M, Duddeck F (2022) Diverse Topologies for Evaluation of Geometric Similarity Metrics [Data set]. *Zenodo*. <https://doi.org/10.5281/zenodo.6323250>

**Publisher's note** Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.



**Nivesh Dommaraju** received his B.Tech. degree in mechanical engineering from Indian Institute of Technology, Kharagpur in 2012. Until 2015, he worked as a maintenance engineer in automobile manufacturing at Tata Motors, India. Pursuing his interest in computer-aided engineering, he earned an M.Sc. in computational engineering at Friedrich-Alexander University, Germany. He is currently pursuing a Ph. D. in cooperative topology optimization at Technical University of Munich in collaboration with Honda Research Institute Europe, Germany. His research interests include computational mechanics, multi-objective design optimization, and artificial intelligence.



**Mariusz Bujny** received his B.Sc. in Mechanical Engineering from the Warsaw University of Technology, Poland, and B.A. in Economics from the University of Warsaw in 2013. In 2015, he earned M.Sc. in Computational Science and Engineering from the Technical University of Munich (TUM), Germany. In 2020, he completed a doctorate at the Chair of Computational Mechanics, TUM, conducted in cooperation with the Honda Research Institute Europe (HRI-EU), Offenbach/Main, Germany. Since 2018, he is a researcher at HRI-EU, where he is currently a Senior Scientist in the Optimization and Creativity Group, shaping international projects in the domains of optimization and vehicle crashworthiness. His research interests include applications of non-gradient techniques and machine learning in topology optimization.



**Stefan Menzel** received the Dipl.-Ing. degree in civil engineering from RWTH Aachen, Aachen, Germany, in 1998 and the Ph.D. degree in civil engineering from Technical University Darmstadt, Darmstadt, Germany, in 2004. Since 2004, he is with the Honda Research Institute Europe, Offenbach, Germany, where he is currently a Chief Scientist with the Optimization and Creativity Group. His current research interests include evolutionary optimization with special focus on adaptive representations, machine learning for knowledge transfer and multidisciplinary optimization for real-world applications.



**Markus Olhofer** is Chief Scientist at the Honda Research Institute Europe since 2010 and responsible for the Complex Systems Optimisation and Analysis Group of the institute. Markus Olhofer graduated in Electrical Engineering with a thesis on face detection and image recognition with a Diplom Ingenieur degree in 1997. After that he worked as PhD student at the Institute for Neural Computation (Institut für Neuroinformatik), Ruhr-

Universität Bochum and received his PhD in 2000 from that university. He joined the Future Technology Research Division at Honda R&D Europe (Deutschland) GmbH in 1998 and since 2001 he works at the Honda Research Institute Europe GmbH.



**Fabian Duddeck** studied civil engineering at the Technical University of Munich (TUM) finishing with a PhD (1997) and a Habilitation in mechanics (2001). Then, he worked at BMW/Munich until 2005 when he became Reader of Computational Mechanics at Queen Mary University of London (until 2020). In addition, he was Maître de Conférences at École des Ponts ParisTech (2006-2016). Since 2010, he is professor for Computational Mechanics at TUM leading a research group on optimization of structures

for aerospace and automotive problems (e.g., crashworthiness). His group is working, e.g., on multifidelity optimization, nonlinear model order reduction, isogeometric analysis, uncertainty, material modeling, and machine learning for mechanical problems.

## Affiliations

Nivesh Dommaraju<sup>1</sup>  · Mariusz Bujny<sup>2</sup>  · Stefan Menzel<sup>2</sup> · Markus Olhofer<sup>2</sup> · Fabian Duddeck<sup>1</sup> 

Mariusz Bujny  
mariusz.bujny@honda-ri.de

Stefan Menzel  
stefan.menzel@honda-ri.de

Markus Olhofer  
markus.olhofer@honda-ri.de

Fabian Duddeck  
duddeck@tum.de

<sup>1</sup> Technical University of Munich, 80333 Munich, Germany

<sup>2</sup> Honda Research Institute Europe GmbH,  
63073 Offenbach/Main, Germany