Design Manifolds Capture the Intrinsic Complexity and Dimension of Design Spaces

Wei Chen[1]
Department of Mechanical Engineering,
University of Maryland,
College Park, MD 20742
e-mail: wchen459@umd.edu

Mark Fuge
Department of Mechanical Engineering,
University of Maryland,
College Park, MD 20742
e-mail: fuge@umd.edu

Jonah Chazan
Department of Computer Science,
University of Maryland,
College Park, MD 20742
e-mail: jchazan@umd.edu

This paper shows how to measure the intrinsic complexity and dimensionality of a design space. It assumes that high-dimensional design parameters actually lie in a much lower-dimensional space that represents semantic attributes—a design manifold. Past work has shown how to embed designs using techniques like autoencoders; in contrast, the method proposed in this paper first captures the inherent properties of a design space and then chooses appropriate embeddings based on the captured properties. We demonstrate this with both synthetic shapes of controllable complexity (using a generalization of the ellipse called the superformula) and real-world designs (glassware and airfoils). We evaluate multiple embeddings by measuring shape reconstruction error, pairwise distance preservation, and captured semantic attributes. By generating fundamental knowledge about the inherent complexity of a design space and how designs differ from one another, our approach allows us to improve design optimization, consumer preference learning, geometric modeling, and other design applications that rely on navigating complex design spaces. Ultimately, this deepens our understanding of design complexity in general. [DOI: 10.1115/1.4036134]

1 Introduction

Products differ among many design parameters. For example, a wine glass contour can be designed using coordinates of B-spline control points: with 20 B-spline control points, a glass would have at least 40 design parameters. However, we cannot arbitrarily set these parameters because the contour must still look like a wine glass. Therefore, high-dimensional design parameters actually lie on a lower-dimensional *design manifold* (Fig. 1) or *semantic space* that encodes semantic design attributes, such as roundness or slenderness. A manifold's *intrinsic dimension* is the minimal dimensionality we need to faithfully represent how those high-dimensional design parameters vary.

Past researchers, both within and beyond design, have proposed many algorithms for mapping high-dimensional spaces to lower-dimensional manifolds and back again—what we call a *design embedding*. But how do you ensure the embedding has captured all the geometric variability among a collection of designs, while not using more dimensions than necessary? How do you evaluate which embedding best captures a given design space? What properties should a good design embedding possess?

This paper answers those questions by proposing methods to study design embeddings. Specifically, our method measures the complexity of a high-dimensional design space and the quality of an embedding. We demonstrate our approach on synthetic *superformula* examples [1] with varying complexity and real-world glassware and airfoil examples.

While this paper focuses on mathematically understanding design spaces, our approach ultimately has implications for several important subfields of engineering design. In engineering optimization, the number of design variables severely impacts accuracy and convergence. In consumer preference models, high-dimensional design spaces complicate capturing human opinion inexpensively or accurately. In design interfaces, designers have difficulty exploring and manipulating high-dimensional design spaces. Because our approach automatically determines a design space's complexity and dimension, these subfields can assess (1) their task's fundamental difficulty and (2) how to best reduce that difficulty. Mathematically, our work deepens our understanding of design complexity in general by illuminating how design embeddings model a design space, how to judge the inherent complexity of a design space, and how to measure the ways designs differ from one another.

Our main contributions are:

(1) A method for embedding designs in the fewest dimensions is needed to control shape variations.
(2) A minimal set of performance metrics for embeddings, including reconstruction error, topology preservation, and whether the manifold captures key shape variations.
(3) Manifold benchmarks with controllable complexity (i.e., nonlinearity, dimensionality, and separability) for testing design embeddings.

2 Related Work

Past research has taken two different approaches for understanding design spaces and synthesizing new shapes or designs—

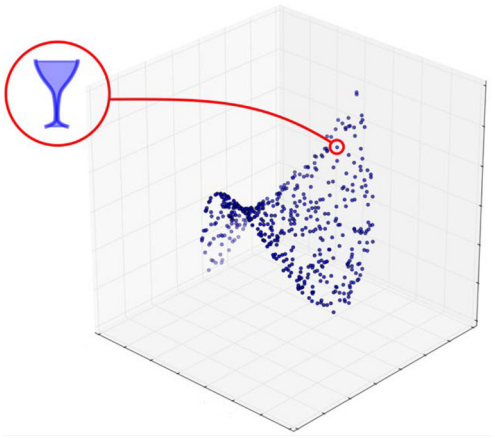

Fig. 1 Three-dimensional visualization of high-dimensional design space showing that design parameters actually lie on a two-dimensional manifold

[1]Corresponding author.

Contributed by the Design Theory and Methodology Committee of ASME for publication in the Journal of Mechanical Design. Manuscript received September 16, 2016; final manuscript received February 7, 2017; published online March 23, 2017. Assoc. Editor: Harrison M. Kim.

knowledge-driven methods and *data-driven methods* [2]. Knowledge-driven methods generate new shapes or designs via explicit rules. One representative example is procedural modeling, which creates 3D models and textures of objects (such as buildings and cities) from sets of rules [3,4]. Another example is computational design synthesis (CDS), which synthesizes designs (such as gearboxes and bicycle frames) based on topological or parametric rules or constraints [5–9]. However, such explicit rules are hard to specify or generalize to diverse design objectives. Data-driven methods, by contrast, learn the representation of geometric structure from examples, adjusting the model to match provided design data. We refer the readers to the survey by Xu et al. [2] for an overview of data-driven shape-processing techniques.

Our work uses a data-driven approach to learn the inherent complexity and the semantic representation of a shape collection and synthesize shapes by exploring that semantic representation. Generally, there are three main data-driven approaches: (1) assembly based modeling, where parts from an existing shape database are assembled onto a new shape [10–14]; (2) statistical-based modeling, where a probability distribution is fitted to shapes in the design space, and plausible shapes are generated based on that distribution [12,15,16]; and (3) shape editing, where a low-dimensional representation is learned from high-dimensional design parameters, and designers create shapes by exploring that low-dimensional representation.

This paper falls into that third category. Many approaches use manifold learning techniques such as multidimensional scaling (MDS) to map the design space to a low-dimensional embedding space [17]. However, such approaches generally construct only one-way mappings (high to low). Some methods can also directly learn a two-way mapping between the design space and the embedding space, such as autoencoders, which uses neural networks to project designs from high to low dimension and back again [18,19]. Another way is to associate shapes with their semantic attributes by crowd-sourcing and then learn a mapping from the semantic attributes to the shapes, such that new shapes can be generated by editing these semantic attributes [20].

3 How Our Contributions Relate to Prior Work

Our hypothesis is that while shapes are represented in a high-dimensional design space, they actually vary in a lower-dimensional manifold, or surface within the larger design space [21,22]. One would then navigate a shape space by moving across this manifold—like an ant walking across a surface of a sphere—while still visiting all the valid designs. This raises several questions: How does one find the manifold (if it exists)? How does one "jump back" to the high-dimensional space (raw geometry) once one has wandered around on the manifold? How does one choose a "useful" manifold, and how does one evaluate that "usefulness?"

A common issue for previous shape synthesis methods is that they often do not address the inherent properties (e.g., intrinsic dimension and nonlinearity) of the geometric design space before embedding, choosing instead to set parameters (such as dimensionality) manually. This causes problems during embedding and shape synthesis because this may not adequately capture shape variability or may use unnecessary dimensions along which designs do not vary. Unlike past work, this paper directly investigates the inherent complexity of a design space under the assumption that different parts of that space may differ in complexity. We discover information such as discontinuities in the design space and the intrinsic dimension of each segment. We then adjust the embeddings and design manifolds based on that information.

Another issue is that given various embeddings (constructed by MDS, autoencoders, etc.), how does one choose the embedding that best enables a smooth and accurate exploration of the space? Reconstruction error is one common metric for evaluating embeddings. It measures how accurately the model can reconstruct input data as it embeds data from high dimension to low and back again. However, sometimes embeddings can excel at reconstruction but ultimately place the data in a lower-dimensional space with unexpected and unintuitive topological structures, such as linear filaments in Deep Autoencoders [23]. While these structures may aid accurate reconstruction, they make it difficult for users to explore the embedded space. In this paper, we propose evaluation metrics that measure three properties of an embedding: (1) its reconstruction accuracy, (2) how well it preserves a design space's topology, and (3) whether it captures the design space's principal semantic attributes. In practice, there is often a tradeoff between these metrics, and our approach allows a designer to visualize and decide among embeddings with respect to that tradeoff.

Past research has also studied various theoretical bounds on the performance of different dimensionality reduction techniques [24–26]. However, current analytical results apply largely to cases where the mapping is linear and performance is measured in terms of Euclidean distances. In this paper, we consider a broader class of manifolds that include nonlinear mappings and non-Euclidean distances. We direct interested readers to refer Sec. S1, which is available under the "Supplemental Materials" tab for this paper on the ASME Digital Collection, for a review and discussion of the available theoretical results.

4 Samples and Data

Before we discuss our approach, we need to introduce some concrete benchmarks of design spaces that we will demonstrate and validate our method over. By design space, we mean any M-dimensional vector ($\mathbf{x} \in \mathbb{R}^M$) that controls a given design's form or function—e.g., its shape, material, power, etc. To create a high-dimensional design space $\mathcal{X}$, we generate a set of design parameters or shape representations $\mathbf{X} \in \mathcal{X}$. For ease of explanation and visualization, this paper uses designs described by 2D curved contours, however, the proposed methods extend to nongeometric design spaces as well, as we show in Sec. S8, which is available under the "Supplemental Materials" tab for this paper on the ASME Digital Collection. Specifically, we uniformly sample points along the shape contours and use their coordinates as $\mathbf{X}$, where $x_j^{(i)}$ and $y_j^{(i)}$ are the x and y coordinates of the jth point on the contour of the ith sample

$$X = \begin{bmatrix} x_1^{(1)} & y_1^{(1)} & \cdots & x_n^{(1)} & y_n^{(1)} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ x_1^{(m)} & y_1^{(m)} & \cdots & x_n^{(m)} & y_n^{(m)} \end{bmatrix}$$

The sample shapes come from two sources: (1) the *superformula* [1] as a synthetic example whose design space complexity we can directly control and (2) glassware and airfoil contours as real-world examples.

4.1 Synthetic Benchmark. Since this paper's goal is to capture a design space's inherent properties, we first need a benchmark dataset whose properties we can directly control; this allows us to measure performance with respect to a known ground truth. Since, to the best of our knowledge, no such benchmark exists for design embeddings, we created one using a generalization of the ellipse—called the *superformula* (see Fig. 2 for examples)—that allows us to control the following properties: nonlinearity, number of separate manifolds, intrinsic dimension, and manifold separability/intersection. Two-dimensional superformula shapes have a multidimensional parameter space in $\mathbb{R}^6$ with parameters (a, b, m,

Fig. 2 Examples of superformula shapes

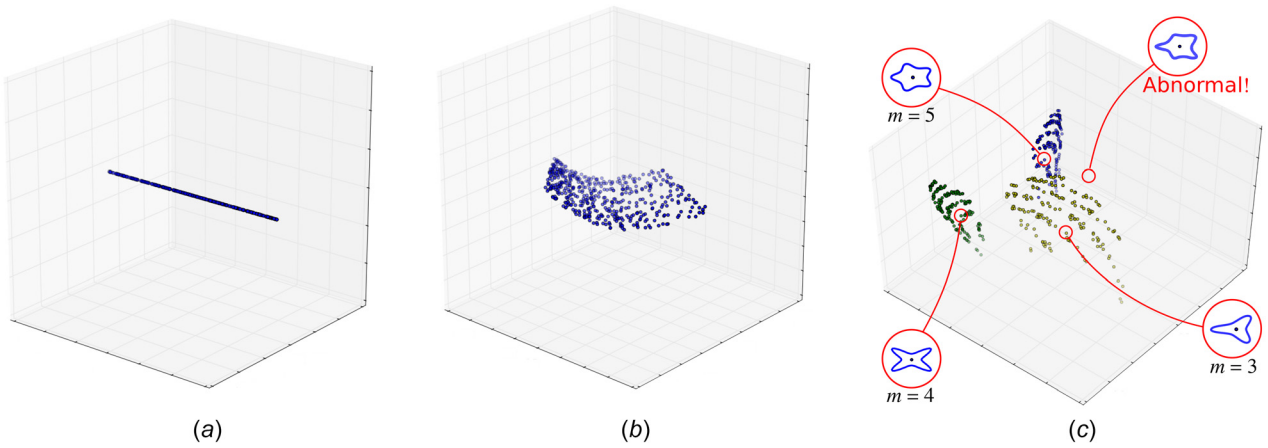

Fig. 3 Three-dimensional visualization of the superformula design space created by a linear mapping from the high-dimensional design space $\mathcal{X}$ to a three-dimensional space, solely for visualization. Each point represents a design. (*a*) Linear design space ($d = 1$) varying s, (*b*) nonlinear space ($d = 2$) varying s and n_3, and (*c*) design space with multiple shape categories.

n_1, n_2, n_3) [1] in Eq. (1). With a radius (r) and an angle (θ), the superformula is expressed in polar coordinates as

$$r(\theta) = \left(\left| \frac{\cos\left(\frac{m\theta}{4}\right)}{a} \right|^{n_2} + \left| \frac{\sin\left(\frac{m\theta}{4}\right)}{b} \right|^{n_3} \right)^{-\frac{1}{n_1}} \tag{1}$$

Thus, the Cartesian coordinates X are

$$(x, y) = (r(\theta)\cos\theta, r(\theta)\sin\theta)$$

4.1.1 Controlling Linearity. By tuning the parameters in Eq. (1), we can vary the nonlinearity of X; for example, by changing the aspect ratio s of the shapes, we can linearly vary X

$$(x, y) = (s \cdot r(\theta)\cos\theta, r(\theta)\sin\theta) \tag{2}$$

We can control the linearity of X by tuning the linear switch s in Eq. (2) or the nonlinear switches $\{a, b, m, n_1, n_2, n_3\}$ in Eq. (1). Figures 3(*a*) and 3(*b*) are examples of controlling linearity of the design space. A design space's linearity is reflected by the curvature of the manifold—higher nonlinearity results in higher curvature.

4.1.2 Controlling the Intrinsic Dimension. To artificially control the intrinsic dimension M of the design space, we construct M-dimensional subspaces of the superformula parameter space by varying M parameters choosing from $\{s, a, b, m, n_1, n_2, n_3\}$ and keeping other parameters fixed, as shown in Figs. 3(*a*) (1D) and 3(*b*) (2D).

4.1.3 Controlling the Number of Shape Categories. A collection of designs does not always consist of just one category of shapes. For example, a collection may contain not only glassware, but also bottles, which have very different contours to glassware and likely have different manifold properties. A naïve embedding, which lumps glasses and bottles together, should perform poorly here, and thus, we need our synthetic benchmark to create similar cases with distinct manifolds. It is possible to have multiple categories of superformula shapes by discretely changing the value of m in Eq. (1). Because the parameter m controls the period of the right-hand side function in Eq. (1), we can use it to set the superformula to an m-pointed-star, as shown in Fig. 3(*c*). The discrete change of m forms separate clusters or submanifolds in the design space.

Another benefit of this formulation is that we can control the separability of the submanifold, since we can make these clusters intersect one another (e.g., see Fig. 7(*b*)). We can generate intersecting clusters by setting the ranges of varying parameters n_2 and n_3, such that all the clusters contain ellipses or circles. Because this superformula benchmark can generate design manifolds with many different properties, we believe it should be useful not only for benchmarking and improving future design embedding techniques but also for evaluating manifold learning techniques in general.

4.2 Real-World Data

4.2.1 Glassware. Glassware is a good real-world example because (1) shape representation using B-splines smoothly fits the glass contours and (2) we can interpret the semantic attributes of glassware—e.g., roundness, slenderness, type of drink, etc. We use 128 glass images, including wine, beer, champagne, and cocktail glasses. We fit each glass contour with a B-spline and build the design space $\mathcal{X}$ using the coordinates of points uniformly sampled across the B-spline curves.

4.2.2 Airfoils. An airfoil is the cross-sectional shape of a wing or blade (of a propeller, rotor, or turbine). Like glassware, we can also represent airfoils via 2D contours and they have discernible semantic attributes that allow us to verify different embeddings. A good airfoil shape embedding can aid airfoil optimization; for example, the proposed method can provide a continuous space with the fewest number of necessary dimensions for an airfoil optimization algorithm to optimize over, improving convergence speed and accuracy. Our airfoil samples are from the UIUC Airfoil Coordinates Database, which provides the Cartesian coordinates for nearly 1600 airfoils.[2] We use linear interpolation to ensure that each airfoil has the same number of coordinates in the design space $\mathcal{X}$.

5 Methodology

First, we try to achieve good design embeddings by understanding the complexity or properties of the design space—for example, detecting the number of submanifolds and what their dimension might be. Based on those results, we then apply embeddings of appropriate complexity and type to the different submanifolds. We will review the high-level details of our methodology, however, some of the specific mathematical, algorithmic, and computational

[2] http://m-selig.ae.illinois.edu/ads/coord_database.html

complexity calculations have been omitted due to brevity; interested readers can review the online supplemental material, which is available under the "Supplemental Materials" tab for this paper on the ASME Digital Collection, as well as the full source code needed to reproduce all the details in this paper.[3]

5.1 Preprocessing. Our raw data may come from shapes with various height and width or have inconsistent or very high dimensionality. These issues obstruct manifold learning and embedding. We apply two preprocessing steps to mitigate these issues.

5.1.1 Shape Correspondence. We ensure that the coordinates for all the designs correspond to consistent cardinality and areas in space. This step is important for our techniques to work well, however, the choice of correspondence technique is not central to the contributions of this paper. See Sec. S2, which is available under the "Supplemental Materials" tab for this paper on the ASME Digital Collection, for full details.

5.1.2 Linear Dimension Reduction. Before performing nonlinear dimension reduction, we first use principal component analysis (PCA) as a linear mapping ($f' : \mathcal{X} \in \mathbb{R}^{D} \rightarrow \mathcal{X}' \in \mathbb{R}^{D'}$) to reduce $\mathcal{X}$ such that it retains 99.5% of the variance. See Sec. S3, which is available under the "Supplemental Materials" tab for this paper on the ASME Digital Collection, for full details.

5.2 Learning Design Space Properties. If designs *do* lie on manifolds, we first need to know the number, intrinsic dimension, and complexity of those manifolds for two reasons. First, knowing the intrinsic dimension M sets the dimensionality d of the semantic space $\mathcal{F}$. Setting $d < M$ makes it impossible to completely capture all the semantic attributes of designs, while $d > M$ introduces unnecessary dimensions along which designs do not vary—this unnecessarily impedes exploration or optimization.

Second, separating different categories of designs helps exclude invalid designs. To illustrate this point, we can first look into the case when there are multiple manifolds in a design space, as shown in Fig. 3(*c*). Suppose we treat these multiple manifolds as one and perform embedding and shape synthesis. We have to use a three-dimensional semantic space to completely capture the variation between the designs. Since there are no design samples in between two manifolds—what we call a *design cavity*, we do not know whether a design from that area is valid. Consequently, in that area, we might synthesize a new shape which looks like a weird hybrid of two designs from two different manifolds, like the abnormal shape in Fig. 3(*c*). In contrast, if we separate these manifolds and then do embedding and shape synthesis on each manifold/design category, we can avoid generating invalid new designs.[4]

For the above reasons, we apply clustering and intrinsic dimension estimation over the dimensionality-reduced design space $\mathcal{X}'$ before embedding designs.

5.2.1 Clustering. In this section, we introduce an adaptive clustering method that automatically captures the number of groups given a collection of designs and then separates these groups. This method uses spectral clustering, where data are clustered using eigenvectors of an affinity matrix [27]. The affinity matrix gives pairwise similarity measures between data points. We adapted a method that is better suited for clustering manifolds, where the similarity measure considers both pairwise distances and principal angles between local tangent spaces [28]. To compute the local tangent spaces, we use an adaptive neighborhood selection algorithm to get the local data matrix. After computing the affinity matrix, we apply a method that automatically detects the number of groups and separate them using the computed affinity matrix [29].

Affinity matrix. To separate manifolds, we use a method based on *robust multiple manifolds structure learning* (RMMSL) [28]. It assumes that within each cluster, samples should not only be close to each other but also form a flat and smooth manifold. A manifold can be flat and smooth if it has small curvature everywhere. The method thus constructs an affinity matrix which incorporates both pairwise distance and a curvature measurement, which is computed via principal angles between local tangent spaces. See Sec. S4, which is available under the "Supplemental Materials" tab for this paper on the ASME Digital Collection, for further algorithmic and computational complexity details.

Neighborhood selection. To compute the curvature measurement, we need the neighboring point set $N(\mathbf{x})$ for each sample point $\mathbf{x}$. We use the neighborhood contraction and expansion algorithm proposed by Zhang et al. [29]. The basic idea is that the neighborhood size k selected for each point $\mathbf{x}$ should not only reflect the local geometric structure of the manifold but also have enough overlap among nearby neighborhoods to allow local information propagation. See Sec. S5, which is available under the "Supplemental Materials" tab for this paper on the ASME Digital Collection, for further algorithmic and computational complexity details.

Group number estimation. Normally, given the correct number of manifolds C (i.e., group number), RMMSL has good performances in separating them [28]. However, it requires manually specifying the number C. To automatically detect the group number C, we apply a method based on *self-tuning spectral clustering* (STSC) [30]. This method automatically infers C by exploiting the structure of the eigenvectors V of the normalized affinity matrix A (i.e., the Laplacian matrix) using an iterative algorithm (with T number of iterations). See Sec. S6, which is available under the "Supplemental Materials" tab for this paper on the ASME Digital Collection, for further algorithmic and computational complexity details.

In sum, we first obtain the nearest neighbors $N(\mathbf{x}_i)$ for each sample $\mathbf{x}_i$, then apply RMMSL to compute the affinity matrix W. Finally, we use W as the affinity matrix for STSC to determine the group number C and assign samples to different groups. The general computational cost for all these steps is $O(N^3)$ when the sample size N is large; see Sec. S7, which is available under the "Supplemental Materials" tab for this paper on the ASME Digital Collection, for further discussion of when this might be cost-prohibitive.

5.2.2 Intrinsic Dimension Estimation. Following the manifold clustering procedure, we apply intrinsic dimension estimation over each manifold/design category. The estimator is based on the local dimension estimation method mentioned in RMMSL [28]. We first obtain the K nearest neighbors of each sample $\mathbf{x}_i$. We set the neighborhood size K using the adaptive method proposed in Ref. [31]. With the neighbors of $\mathbf{x}_i$ and a weight matrix S, we construct a local structure matrix

$$\mathbf{T}_i = (\mathbf{X}_i - \mathbf{x}_i\mathbf{1}^{\mathrm{T}})\mathbf{S}\mathbf{S}^{\mathrm{T}}(\mathbf{X}_i - \mathbf{x}_i\mathbf{1}^{\mathrm{T}})^{\mathrm{T}}$$

where $\mathbf{S}$ is a diagonal matrix and can be set as $S_{jj} = 1/(\sigma_n^2 + \sigma^2\alpha(\|\mathbf{x}_{i_j} - \mathbf{x}_i\|_2))$, σ_n^2 and σ^2 indicate the scales of the noise and the error, and $\alpha(\cdot)$ is a monotonically nondecreasing function with non-negative domain (e.g., a quadratic) [28].

The local intrinsic dimension d_i is estimated from the eigenvalues of $\mathbf{T}_i$—similar to dimensionality estimation using PCA. A category's overall intrinsic dimension is the mean intrinsic dimension of all the points in that category. In practice, point-wide intrinsic dimension can be noisy (and thus appear to change dimensionality often). We apply a kernel density smoother (KDE) to local dimensionality estimates to account for our assumption that the local dimensionality should not vary drastically within a neighborhood on a smooth manifold. For example, if the local dimensionality estimations for $\mathbf{x}_i$ and its five neighbors are {2, 3, 2, 2, 2, 2}, the second

[3] https://github.com/IDEALLab/design_embeddings_jmd_2016

[4] An astute reader may notice that designs "off-the-manifold" may be, in some sense, creative or innovative. We return to that discussion in Sec. 6.3.

estimation is likely to be incorrect. After applying KDE, the new estimations will be {2, 2, 2, 2, 2, 2}. We use the Epanechnikov kernel to limit density estimation to a local neighborhood and set the kernel bandwidth adaptively based on the distance between each sample and its Kth neighbor.

5.3 Embedding and Shape Synthesis. We used methods involving PCA, kernel PCA (with a radial basis function (RBF) kernel) [32], and stacked denoising autoencoders (SdA) [33], which all simultaneously learn a mapping f from the space $\mathcal{X}'$ to the semantic space $\mathcal{F}$ ($f : \mathcal{X}' \in \mathbb{R}^{D'} \rightarrow \mathcal{F} \in \mathbb{R}^{d}$) and a reverse mapping g from the $\mathcal{F}$ back to $\mathcal{X}'$ ($g : \mathcal{F} \in \mathbb{R}^{d} \rightarrow \mathcal{X}' \in \mathbb{R}^{D'}$), where $D' \geq d$.

PCA performs an orthogonal linear transformation on the input data. Kernel PCA extends PCA, achieving a nonlinear transformation via the kernel trick [34]. The SdA extends the stacked autoencoder [35], which is a multilayer artificial neural network that nonlinearly maps $\mathcal{X}'$ and $\mathcal{F}$ using nonlinear activation functions [36].

We split any design data into a training set and test set. We further split the training data via fivefold cross validation to optimize the hyperparameters using the *sequential model-based algorithm configuration* (SMAC) [37]. After optimizing any hyperparameters, the model trains each example with the entire training set. We test each model with the test set.

After mapping from design space $\mathcal{X}'$ to semantic space $\mathcal{F}$, we also need to map back to the original $\mathcal{X}$ from $\mathcal{F}$ —i.e., given certain semantic attributes, we want to generate new shapes. This is achieved by first applying mapping g and then applying g' ($g' : \mathcal{X}' \in \mathbb{R}^{D'} \rightarrow \mathcal{X} \in \mathbb{R}^{D}$), which is the inverse mapping of f'.

To visualize the mapping $g' : \mathcal{F} \rightarrow \mathcal{X}$, we uniformly sample from $\mathcal{F}$, map those samples back to $\mathcal{X}$ to synthesize their shapes and then plot them in Fig. 4(*b*). Note that although we only visualize a limited number of shapes generated from the semantic space $\mathcal{F}$, we can generate infinitely many shapes when continuously exploring in $\mathcal{F}$.

Every semantic space should have a boundary beyond which designs are not guaranteed to be valid. For example, at the top left of Fig. 4(*b*), the glass contours on the two sides intersect. We call these *infeasible shapes*, i.e., shapes that are unrealistic or invalid in the real-world. To limit $\mathcal{F}$ to only feasible shapes, we take the convex hull of the training samples in $\mathcal{F}$ as shown in Fig. 4(*a*) and set its boundary as the boundary for the *feasible semantic space*. We sample and synthesize shapes inside the feasible semantic space, as shown in Fig. 4(*c*). Because training samples all have feasible shapes, any designs lying between any two training samples should be valid if the semantic space preserves the original design space's topology (i.e., the space is not highly distorted). As a result, this method may not explore innovative, unusual designs. While this paper's main focus is understanding the complexity of an *existing* design space, we discuss how to find innovative designs in Sec. 6.3.

5.4 Evaluation. To evaluate embeddings, we consider these questions:

(1) Given known semantic attributes (e.g., roundness and slenderness), can the embedding precisely restore the original design parameters that created it?
(2) Do shapes in the semantic space ($\mathcal{F}$) vary similarly compared to the original shapes in the design space ($\mathcal{X}$)?
(3) Does the embedding precisely capture all the attributes that control the variation of shapes?

To answer these questions, we propose three metrics for comparing embeddings: (1) reconstruction error, (2) geodesic distance inconsistency (GDI), and (3) principal attributes.

5.4.1 Reconstruction Error. Reconstruction error measures how the actual design parameters $\mathbf{X}'$ differ from the design parameters of the input data once we project them onto the low-dimensional manifold and unproject back into high-dimensional space. We use the *symmetric mean absolute percentage error* (SMAPE) [38] to measure reconstruction error

$$\varepsilon = \frac{1}{mn} \sum_{i=1}^{m} \sum_{j=1}^{n} \frac{|r_j^{(i)} - s_j^{(i)}|}{|r_j^{(i)}| + |s_j^{(i)}|} \tag{3}$$

where m is the sample size, n is the number of design parameters for each sample, $r_j^{(i)}$ is the ith reconstructed design parameter for the jth sample, and $s_j^{(i)}$ is the ith original design parameter for the jth sample.

5.4.2 Pairwise Distance Preservation. To answer the second question, we can compare the pairwise distances of samples in the high-dimensional design space $\mathcal{X}$ versus the low-dimensional semantic space $\mathcal{F}$. Generally, as shown in Fig. 5, similar designs (A and B) have similar shape representations (e.g., Cartesian coordinates of shape outlines) in $\mathcal{X}$, thus are closer in $\mathcal{X}$ than dissimilar designs (A and C). We want such pairwise distances to be preserved in $\mathcal{F}$ (i.e., $d_{AB} < d_{AC}$) such that shapes will vary in the same manner as they do in $\mathcal{X}$. Since we assume our samples lie on a manifold in $\mathcal{X}$, we use the pairwise geodesic distances along the manifold as the pairwise distances to be preserved after the embedding.

Specifically, we construct a nearest neighbor graph G over the samples $\mathbf{X} \in \mathcal{X}$ to model the manifold structure. G is a weighted graph where the edge weight between neighbors is their Euclidean distance. The nearest neighbors are selected using the neighborhood contraction and expansion algorithm mentioned in the neighborhood selection section. This algorithm adaptively chooses the neighborhood size for each sample based on its local manifold geometry. The neighborhood size is large where the manifold is flat, and small where it is curvy. Thus, this method prevents "shortcuts" across high-curvature manifolds and maintains large

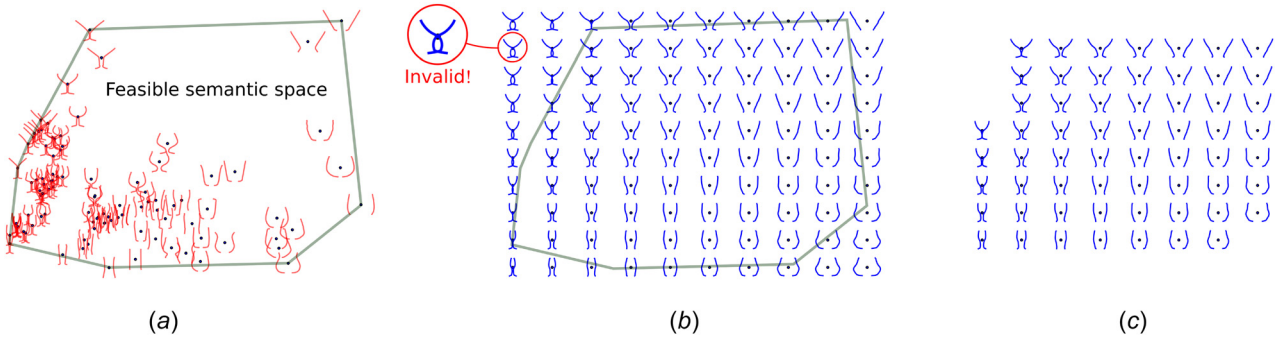

Fig. 4 Set boundary of the feasible semantic space: (*a*) create a convex hull of the training set in the semantic space, (*b*) copy the boundary of the convex hull to the grid of new designs generated from the semantic space, and (*c*) remove designs outside the boundary

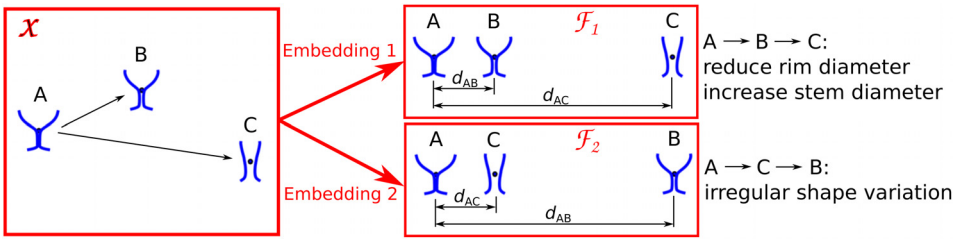

Fig. 5 Illustration of pairwise distance preservation. Similar designs (A and B) have similar shape representations in $\mathcal{X}$, thus are closer in $\mathcal{X}$ than dissimilar designs (A and C). We want such relation of pairwise distances to be preserved in $\mathcal{F}$ (i.e., $d_{AB} < d_{AC}$) such that shapes will vary in the same manner as they do in $\mathcal{X}$.

enough overlap among nearby neighborhoods. Then, we compute all the pairs shortest paths for the graph G and construct a geodesic distance matrix $\mathbf{D}_G$. We compare $\mathbf{D}_G$ with the pairwise Euclidean distance matrix $\mathbf{D}$ of the embedded samples $\mathbf{F} \in \mathcal{F}$ using Pearson's correlation coefficient. The *geodesic distance inconsistency* (GDI) can be expressed as

$$\text{GDI} = 1 - \rho(\mathbf{D}_G, \mathbf{D})^2 \tag{4}$$

where $\rho(\mathbf{D}_G, \mathbf{D})$ is the Pearson's correlation coefficient between $\mathbf{D}_G$ and $\mathbf{D}$. Lower GDI indicates that the embedding better preserves pairwise distances.

Normally, embedding methods like Isomap will have low GDIs because they explicitly optimize pairwise distances. However, they cannot simultaneously learn two-way mappings between the design space and the embedding space. Methods like autoencoders are able to learn two-way mappings, but they usually have higher GDIs because they minimize reconstruction error rather than preserve distances. There is often a tradeoff between reconstruction and distance preservation; embeddings that explicitly optimize both objectives would be an interesting topic for future research.

5.4.3 Principal Attributes. For superformula examples, we know what their correct semantic spaces should look like by looking at their parameter spaces. A *parameter space* $\mathcal{P}$ (e.g., see Fig. 6(*a*)) for the superformula contains the shapes generated by all the possible combinations of values for all the parameters used to modify the shapes. It is this space from which we randomly select training and testing samples. For example, if we fix parameters $\{a, b, m, n_2, n_3\}$ and vary $\{s, n_1\}$ in Eqs. (1) and (2), the superformula parameter space will have two dimensions (i.e., $\mathcal{P} \in \mathbb{R}^2$). Along the first dimension, some shape attribute (e.g., aspect ratio) changes with s, and along the second dimension another attribute (e.g., roundness) changes with n_1. We call these attributes the *principal attributes*. We can also vary these two attributes along a single dimension by simultaneously varying s and n_1 (e.g., let $s = \alpha t$ and $n_1 = \beta t$, where α and β are coefficients, and t is a variable). Figure 7(*a*) shows an example where two shape attributes vary along each dimension—the aspect ratio[5] changes along one dimension and roundness along the other, and the number of arms changes over both dimensions. Because the number of arms is not continuous, the clustering algorithm should separate shapes with different number of arms. And within each cluster, a good embedding should capture the other two attributes—the aspect ratio and roundness.

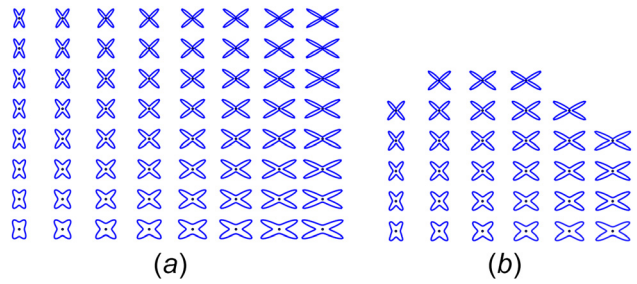

Fig. 6 An example comparing shapes generated from the semantic space $\mathcal{F}$ versus the superformula parameter space $\mathcal{P}$. If the embedding precisely captures the principal attributes, shapes from $\mathcal{F}$ should look like those from $\mathcal{P}$—with neither extra unexpected shape variation nor missing diversity. (*a*) Shapes in the superformula parameter space and (*b*) generated shapes in the semantic space.

A good embedding should precisely capture the right principal attributes, such that tuning these attributes creates diverse but valid shapes. By comparing the shapes generated from the semantic space $\mathcal{F}$ with those from the superformula parameter space $\mathcal{P}$, we can evaluate whether the embedding precisely captures all the attributes that control the variation of shapes. That is, shapes generated from $\mathcal{F}$ should look similar to those from $\mathcal{P}$—with neither unexpected shape variation nor missing diversity, as shown in Fig. 6.

6 Results and Discussion

We evaluated our method's performance at recovering various design space properties. We also compared how well different embedding approaches captured shape attributes.

6.1 Design Space Properties. We use the superformula examples to test the accuracy of our clustering algorithm and intrinsic dimension estimator. We conducted experiments with the number of clusters C set from 1 to 5, intrinsic dimension from 1 to 3, and three levels of linearity (curvature of the manifold). All the experiments correctly separated the superformula shape categories and estimated the correct intrinsic dimensions (Fig. 7). In cases where multiple manifolds intersect (Fig. 7(*b*)), it is improper to use metrics like the rand index to evaluate clustering accuracy, because samples at the intersection can be classified to any adjacent group. For example, shapes turn into ellipses at the intersection (Fig. 7(*a*)), so it does not matter whether they belong to the four- or five-pointed star group. Figure 7(*c*) shows that our approach captures this case.

Figure 8 demonstrates a case where two manifolds with *different intrinsic dimensions* intersect: one superformula category with an intrinsic dimension of one intersects with another category with a intrinsic dimension of two. Our method correctly separates the two categories and estimates the intrinsic dimension for each category, as shown in Fig. 8(*c*).

6.2 Principal Attributes. As mentioned above, a good embedding should precisely capture the right principal attributes. For superformula examples, we check this by comparing the shapes generated from the semantic space $\mathcal{F}$ with those from the superformula parameter space $\mathcal{P}$. For the example shown in Fig. 7,

[5]All the shapes shown in the figures are scaled to the same height. Thus, the aspect ratio will matter instead of the height or the width.

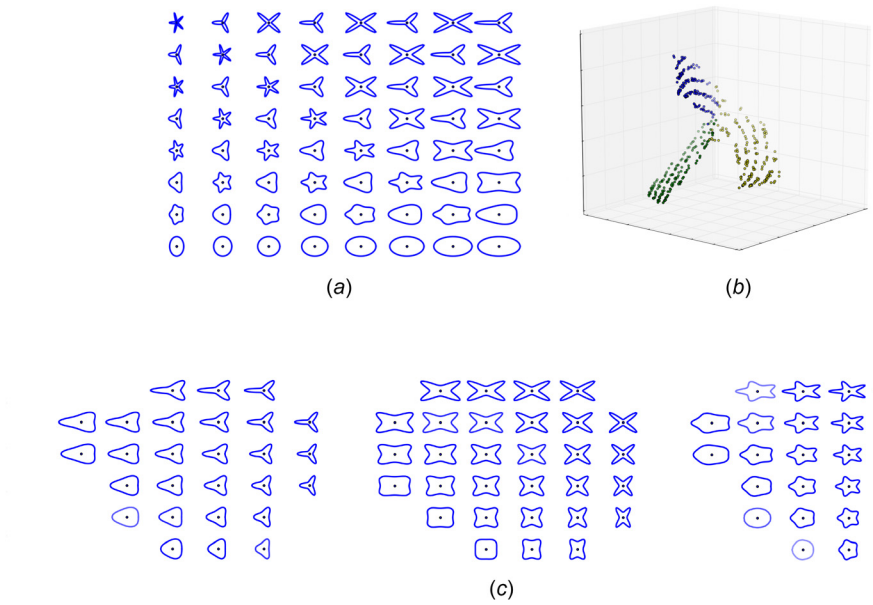

Fig. 7 Multiple superformula categories with intersection. Our approach correctly separates the three submanifolds, even though they all connect via a common seam. (*a*) Shapes in the superformula parameter space, (*b*) result of manifold clustering (as in Fig. 3, the design space $\mathcal{X}$ is visualized in three dimensions), and (*c*) generated shapes in semantic spaces. Since there are three categories, we have three separated semantic spaces.

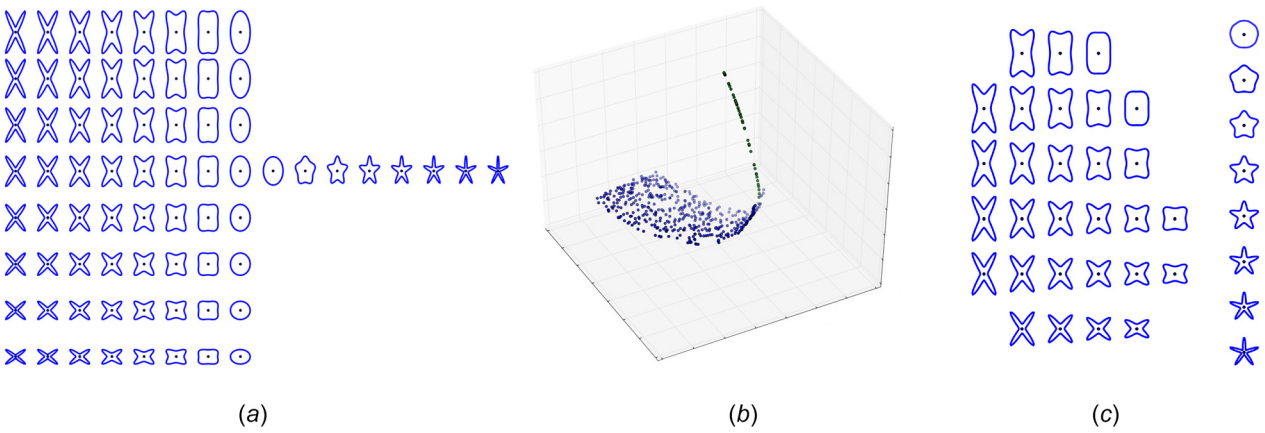

Fig. 8 Multiple superformula categories with intersection and different intrinsic dimensions. Our intrinsic dimension estimator automatically detects the appropriate dimensionality of the semantic space for each design category (*c*). (*a*) Shapes in the superformula parameter space, (*b*) result of manifold clustering (as in Fig. 3, the design space $\mathcal{X}$ is visualized in three dimensions), and (*c*) generated shapes in semantic spaces.

the clustering separated shapes with different number of arms, and each cluster's embedding correctly captured the two principal attributes—aspect ratio and roundness. In the example of Fig. 8, as expected, the embedding captured the aspect ratio and the roundness attributes of the four-pointed stars, and just the roundness for the five-pointed stars.

Figure 9 shows the synthesized glassware in a three-dimensional semantic space. This embedding captured three attributes—the rim diameter, the stem diameter, and the curvature. With the variation of these three attributes, we synthesized a collection of shapes that generally covers all the training samples—wine, beer, champagne, and cocktail glasses. Similarly, the airfoil embedding shown in Fig. 10 also captured three attributes—the upper and lower surface protrusion and the trailing edge direction.

For the airfoil example, the uncovered design manifold allows us to optimize airfoil shapes over this continuous low-dimensional space instead of the original geometric design space, using metamodeling techniques like Bayesian optimization. In our future work, we will compare the performance of metamodeling techniques with and without applying our dimensionality reduction method.

6.3 Are All Areas on the Manifold Equally Valid? The different shades in Figs. 7–10 represent for local density of the shape collection (i.e., training samples) distributed in the semantic space $\mathcal{F}$. At positions nearby the training samples (e.g., point A in Fig. 11), we plot the synthesized shape with a darker shade. This means we are more certain of its validity because it lies closer to real-world training samples. In contrast, in locations where training samples are sparse (e.g., point B in Fig. 11), we plot the synthesized shape with a lighter shade. This means we are less certain of its validity, and our model may create shapes that diverge from the training samples. In the semantic space where the training samples are absent—what we call a *design cavity*—new designs might be innovative or they may be unrealistic. For example, as

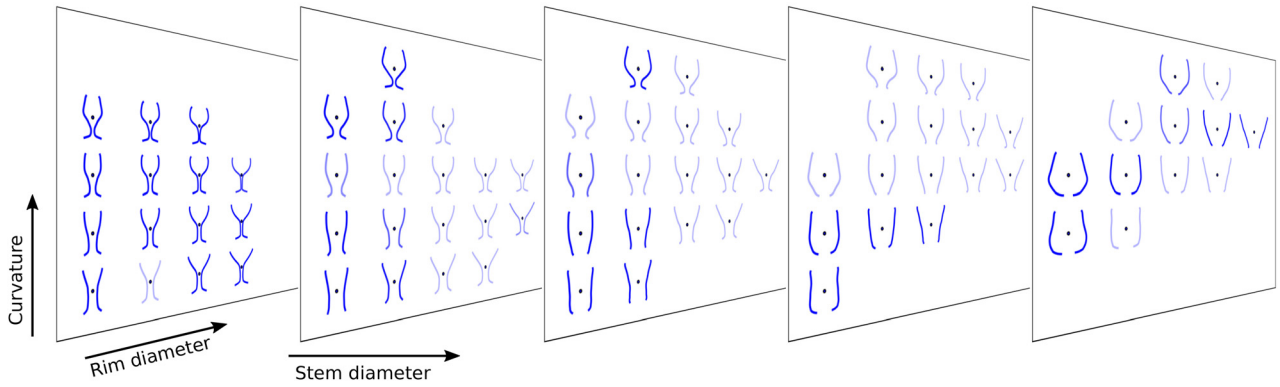

Fig. 9 Synthesized glassware shapes in a 3D semantic space. The embedding captured three shape attributes—the rim diameter, the stem diameter, and the curvature.

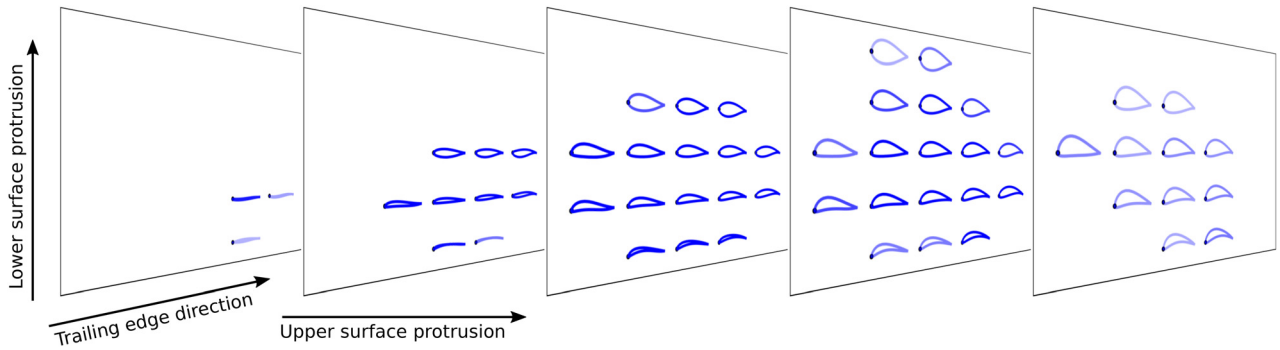

Fig. 10 Synthesized airfoil shapes in a 3D semantic space. The embedding captured three shape attributes—the upper and lower surface protrusion and the trailing edge direction.

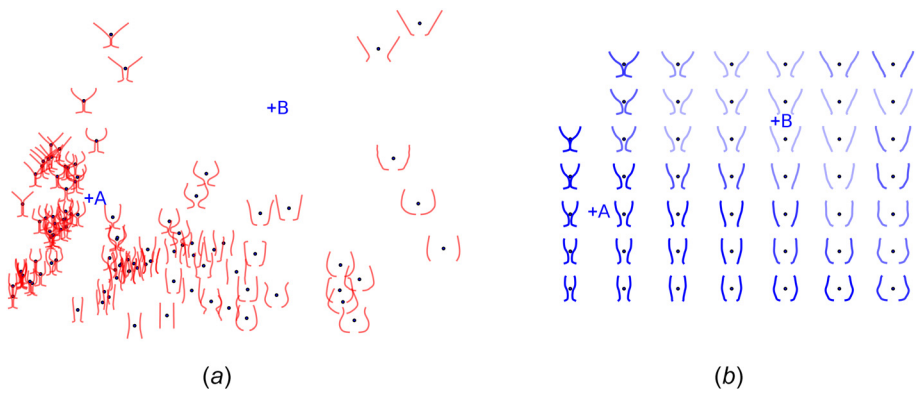

Fig. 11 Point A has high sample density and thus higher confidence that synthesized shapes will look similar to nearby real-world samples. In contrast, point B has low sample density and thus lower confidence but higher chance of generating an unusual or creative shape. Shade darkness correlates with higher local density. (*a*) Arrangement of training samples in the semantic space (for simplicity, this is a 2D projection of the 3D semantic space) and (*b*) synthesized shapes.

shown in Fig. 9, inside areas where shapes have light shades, we synthesized glassware that is not similar to any of our training samples. In this case, the model generated innovative designs rather than unrealistic ones, however, to the best of our knowledge, there is no formal mathematical way of differentiating those two cases automatically. Doing so would be a compelling topic of future research.

6.4 Sample Arrangement in Semantic Space. Figure 12 shows the comparison between results obtained from different embedding methods. For this superformula example, PCA has a high reconstruction error because the design space is nonlinear. This results in some abnormal shapes (i.e., shapes with attributes that do not exist in the parameter space in Fig. 12(*a*)) in the semantic space created by PCA (Fig. 12(*c*)). SdA and kernel PCA have similar reconstruction errors. However, the semantic space created by kernel PCA (Fig. 12(*d*)) better aligns with the original parameter space than that of SdA (Fig. 12(*e*)), because SdA generates abnormal shapes (Fig. 12(*e*), top right). Since these abnormal shapes have light shades, we know that they are inside the design cavity, while in fact if the embedding preserved the geodesic

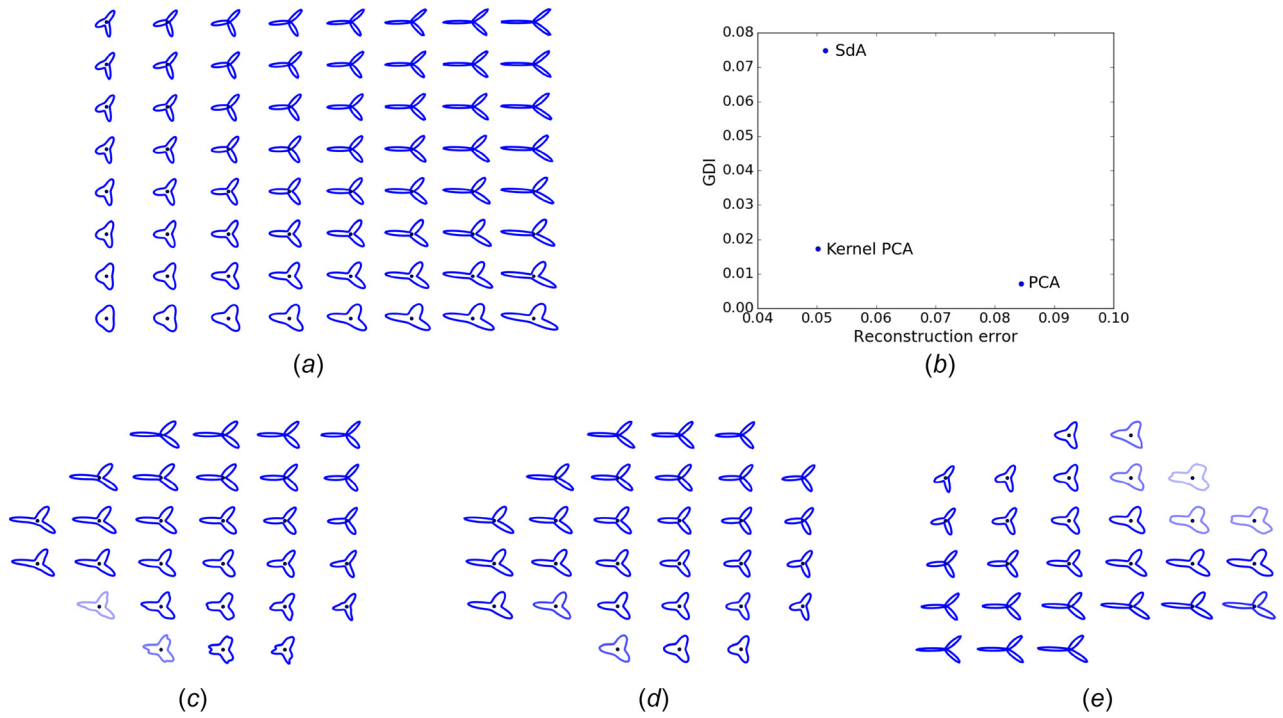

Fig. 12 Comparison of different embedding methods. The abnormal shapes generated by PCA and SdA are due to high reconstruction error and high GDI, respectively. (*a*) Shapes in the superformula parameter space, (*b*) reconstruction error and geodesic distance inconsistency, (*c*) embedding and shape synthesis result by PCA, (*d*) embedding and shape synthesis result by kernel PCA, and (*e*) embedding and shape synthesis result by a stacked denoising autoencoder.

distances between samples (i.e., low GDI), the sample arrangement should resemble that of PCA or kernel PCA, where there is no design cavity (Figs. 12(*c*) and 12(*d*)). Therefore, both high reconstruction errors (e.g., PCA in this example) and high GDI (e.g., SdA in this example) can create abnormal or invalid shapes.

7 Conclusions

We introduced an approach to learn the inherent properties of a design space and evaluate two-way mappings between a design space and a semantic space. By correctly identifying the design space properties such as the number of design categories and the intrinsic dimension, one can then create an embedding that precisely captures the principal attributes of each design category, assuming that the embedding is well chosen based on the reconstruction error and the pairwise distance preservation. This means the synthesized shapes will have no unexpected shape variation, missing diversity, or repeated shapes (brought about by unnecessary dimensions). We also introduced a benchmark for rigorously testing design embeddings that accounted for nonlinear, multiple (potentially intersecting) manifolds of controllable intrinsic dimension. We encourage others to use this benchmark to improve future design embeddings.

While this paper mainly addressed geometric design spaces, our approach would extend to any type of design space, including those that involve text, materials, or combinations with geometry (for an example of a combined material and geometry space, see Sec. S8, which is available under the "Supplemental Materials" tab for this paper on the ASME Digital Collection). It could apply to improving interfaces that help novices explore designs, aiding high-dimensional design optimization, and helping model consumer preferences in high-dimensional design spaces. Our work's main implication is that choosing a design embedding carries with it important choices about what you value in your semantic space: Should it reconstruct designs consistently? Should it preserve the topology of the design space? What semantic attributes should the semantic space capture? Choosing an embedding with the properties you want is not straightforward; our approach provides a principled way to compare and contrast embeddings—to help navigate those options and identify useful properties of both the embedding and your design space in general.

Acknowledgment

An earlier version of this work appeared in the ASME IDETC 2016 conference [39]. We thank the reviewers of both versions for many insightful comments that substantially improved the quality and rigor of the manuscript. We also acknowledge funding from the University of Maryland's Department of Mechanical Engineering and a Minta Martin Grant from the College of Engineering.

References

[1] Gielis, J., 2003, "A Generic Geometric Transformation That Unifies a Wide Range of Natural and Abstract Shapes," Am. J. Bot., **90**(3), pp. 333–338.
[2] Xu, K., Kim, V. G., Huang, Q., Mitra, N., and Kalogerakis, E., 2016, "Data-Driven Shape Analysis and Processing," SIGGRAPH ASIA 2016 Courses, (SA), Macau, Dec. 5–8, pp. 1–4.
[3] Müller, P., Wonka, P., Haegler, S., Ulmer, A., and Van Gool, L., 2006, "Procedural Modeling of Buildings," ACM Trans. Graphics, **25**(3), pp. 614–623.
[4] Talton, J. O., Lou, Y., Lesser, S., Duke, J., Měch, R., and Koltun, V., 2011, "Metropolis Procedural Modeling," ACM Trans. Graphics, **30**(2), pp. 1–14.
[5] Cagan, J., Campbell, M. I., Finger, S., and Tomiyama, T., 2005, "A Framework for Computational Design Synthesis: Model and Applications," ASME J. Comput. Inf. Sci. Eng., **5**(3), pp. 171–181.
[6] Wyatt, D. F., Wynn, D. C., Jarrett, J. P., and Clarkson, P. J., 2012, "Supporting Product Architecture Design Using Computational Design Synthesis With Network Structure Constraints," Res. Eng. Des., **23**(1), pp. 17–52.
[7] Königseder, C., and Shea, K., 2015, "Analyzing Generative Design Grammars," *Design Computing and Cognition'14*, Springer, Cham, Switzerland, pp. 363–381.
[8] Oberhauser, M., Sartorius, S., Gmeiner, T., and Shea, K., 2015, "Computational Design Synthesis of Aircraft Configurations With Shape Grammars," *Design Computing and Cognition'14*, Springer, Cham, Switzerland, pp. 21–39.
[9] Königseder, C., and Shea, K., 2016, "Comparing Strategies for Topologic and Parametric Rule Application in Automated Computational Design Synthesis," ASME J. Mech. Des., **138**(1), p. 011102.
[10] Chaudhuri, S., and Koltun, V., 2010, "Data-Driven Suggestions for Creativity Support in 3D Modeling," ACM Trans. Graphics, **29**(6), pp. 1–10.
[11] Chaudhuri, S., Kalogerakis, E., Guibas, L., and Koltun, V., 2011, "Probabilistic Reasoning for Assembly-Based 3D Modeling," ACM Trans. Graphics, **30**(4), pp. 1–10.

[12] Kalogerakis, E., Chaudhuri, S., Koller, D., and Koltun, V., 2012, "A Probabilistic Model for Component-Based Shape Synthesis," ACM Trans. Graphics, **31**(4), pp. 1–11.
[13] Chaudhuri, S., Kalogerakis, E., Giguere, S., and Funkhouser, T., 2013, "Attribit: Content Creation With Semantic Attributes," 26th Annual ACM Symposium on User Interface Software and Technology (UIST), St. Andrews, Scotland, Oct. 8–11, pp. 193–202.
[14] Guo, X., Lin, J., Xu, K., and Jin, X., 2014, "Creature Grammar for Creative Modeling of 3D Monsters," Graphical Models, **76**(5), pp. 376–389.
[15] Talton, J. O., Gibson, D., Yang, L., Hanrahan, P., and Koltun, V., 2009, "Exploratory Modeling With Collaborative Design Spaces," ACM Trans. Graphics, **28**(5), pp. 1–10.
[16] Fish, N., Averkiou, M., Van Kaick, O., Sorkine-Hornung, O., Cohen-Or, D., and Mitra, N. J., 2014, "Meta-Representation of Shape Families," ACM Trans. Graphics, **33**(4), pp. 1–11.
[17] Averkiou, M., Kim, V. G., Zheng, Y., and Mitra, N. J., 2014, "Shapesynth: Parameterizing Model Collections for Coupled Shape Exploration and Synthesis," Comput. Graphics Forum, **33**(2), pp. 125–134.
[18] Yumer, M. E., Asente, P., Mech, R., and Kara, L. B., 2015, "Procedural Modeling Using Autoencoder Networks," 28th ACM User Interface Software and Technology Symposium (UIST), Daegu, Kyungpook, Korea, Nov. 8–11, pp. 109–118.
[19] Burnap, A., Liu, Y., Pan, Y., Lee, H., Gonzalez, R., and Papalambros, P. Y., 2016, "Estimating and Exploring the Product From Design Space Using Deep Generative Models," ASME Paper No. DETC2016-60091.
[20] Yumer, M. E., Chaudhuri, S., Hodgins, J. K., and Kara, L. B., 2015, "Semantic Shape Editing Using Deformation Handles," ACM Trans. Graphics, **34**(4), pp. 1–12.
[21] Van der Maaten, L., Postma, E., and Van den Herik, H., 2009, "Dimensionality Reduction: A Comparative Review," Technical Report No. TiCC TR 2009-005.
[22] Bengio, Y., Courville, A., and Vincent, P., 2013, "Representation Learning: A Review and New Perspectives," IEEE Trans. Pattern Anal. Mach. Intell., **35**(8), pp. 1798–1828.
[23] Duvenaud, D. K., Rippel, O., Adams, R. P., and Ghahramani, Z., 2014, "Avoiding Pathologies in Very Deep Networks," arXiv:1402.5836.
[24] Larsen, K. G., and Nelson, J., 2014, "The Johnson–Lindenstrauss Lemma Is Optimal for Linear Dimensionality Reduction," preprint arXiv:1411.2404.
[25] Bartal, Y., Recht, B., and Schulman, L. J., 2011, "Dimensionality Reduction: Beyond the Johnson–Lindenstrauss Bound," Twenty-Second Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), San Francisco, CA, Jan. 23–25, pp. 868–887.
[26] Shawe-Taylor, J., Williams, C. K., Cristianini, N., and Kandola, J., 2005, "On the Eigenspectrum of the Gram Matrix and the Generalization Error of Kernel-PCA," IEEE Trans. Inf. Theory, **51**(7), pp. 2510–2522.
[27] Ng, A. Y., Jordan, M. I., and Weiss, Y., 2001, "On Spectral Clustering: Analysis and an Algorithm," *Advances in Neural Information Processing Systems*, MIT Press, Cambridge, MA, Vol. 14, pp. 849–856.
[28] Gong, D., Zhao, X., and Medioni, G., 2012, "Robust Multiple Manifolds Structure Learning," preprint arXiv:1206.4624.
[29] Zhang, Z., Wang, J., and Zha, H., 2012, "Adaptive Manifold Learning," IEEE Trans. Pattern Anal. Mach. Intell., **34**(2), pp. 253–265.
[30] Zelnik-Manor, L., and Perona, P., 2004, "Self-Tuning Spectral Clustering," *Advances in Neural Information Processing Systems*, MIT Press, Cambridge, MA, Vol. 17, pp. 1601–1608.
[31] Samko, O., Marshall, A. D., and Rosin, P. L., 2006, "Selection of the Optimal Parameter Value for the Isomap Algorithm," Pattern Recog. Lett., **27**(9), pp. 968–979.
[32] Schölkopf, B., Smola, A., and Müller, K.-R., 1998, "Nonlinear Component Analysis as a Kernel Eigenvalue Problem," Neural Comput., **10**(5), pp. 1299–1319.
[33] Vincent, P., Larochelle, H., Lajoie, I., Bengio, Y., and Manzagol, P.-A., 2010, "Stacked Denoising Autoencoders: Learning Useful Representations in a Deep Network With a Local Denoising Criterion," J. Mach. Learn. Res., **11**(2), pp. 3371–3408.
[34] Schölkopf, B., Smola, A., and Müller, K.-R., 1997, "Kernel Principal Component Analysis," International Conference on Artificial Neural Networks (ICANN), Lausanne, Switzerland, Oct. 8–10, pp. 583–588.
[35] Vincent, P., Larochelle, H., Bengio, Y., and Manzagol, P.-A., 2008, "Extracting and Composing Robust Features With Denoising Autoencoders," 25th International Conference on Machine Learning (ICML), Helsinki, Finland, July 5–9, pp. 1096–1103.
[36] Bengio, Y., Lamblin, P., Popovici, D., and Larochelle, H., 2007, "Greedy Layer-Wise Training of Deep Networks," *Advances in Neural Information Processing Systems* (NIPS), MIT Press, Cambridge, MA, Vol. 19, pp. 153–160.
[37] Hutter, F., Hoos, H. H., and Leyton-Brown, K., 2011, "Sequential Model-Based Optimization for General Algorithm Configuration," Lecture Notes in Computer Science (LNCS), Vol. 6683, Springer, Berlin, pp. 507–523.
[38] Makridakis, S., 1993, "Accuracy Measures: Theoretical and Practical Concerns," Int. J. Forecasting, **9**(4), pp. 527–529.
[39] Chen, W., Chazan, J., and Fuge, M., 2016, "How Designs Differ: Non-Linear Embeddings Illuminate Intrinsic Design Complexity," ASME Paper No. DETC2016-60112.