Part-Aware Product Design Agent Using Deep Generative Network and Local Linear Embedding

Xingang Li
University of Arkansas
xl038@uark.edu

Charles Xie
Institute for Future Intelligence
charles@intofuture.org

Zhenghui Sha*
University of Arkansas
zsha@uark.edu

Abstract

In this study, we present a data-driven generative design approach that can augment human creativity in product shape design with the objective of improving system performance. The approach consists of two modules: 1) a 3D mesh generative design module that can generate part-aware 3D objects using variational auto-encoder (VAE), and 2) a low-fidelity evaluation module that can rapidly assess the engineering performance of 3D objects based on locally linear embedding (LLE). This approach has two unique features. First, it generates 3D meshes that can better capture surface details (e.g., smoothness and curvature) given individual parts' interconnection and constraints (i.e., part-aware), as opposed to generating holistic 3D shapes. Second, the LLE-based solver can assess the engineering performance of the generated 3D shapes to realize real-time evaluation. Our approach is applied to car design to reduce air drag for optimal aerodynamic performance.

1. Introduction and Motivation

With advances in Artificial intelligence (AI), AI has shown its capabilities in many "human" jobs, like speech translation, customer service, and even decision-making. Dellermann et al. [1] argue that, in the following decades, AI will not replace but rather collaborate with humans in most domains. They treat this human-AI collaboration as hybrid intelligence, which leverages the complementary strengths of human intelligence and AI. In the design field, AI has also greatly facilitated human designers' decision-making in different design processes. For example, researchers have successfully embedded intelligent agents into traditional computer-aided design (CAD) software (e.g., [2]) and some custom research platforms (e.g., [3,4]) in support of conceptual design and design optimization. This could save a large amount of human labor, thus significantly shorten the cycle and iteration of product design and development.

Among various AI techniques, generative design (GD) models using deep neural networks can be used as intelligent design agents, particularly in product shape design. GD is a term for a class of tools that can generate novel yet realistic designs by leveraging computational and manufacturing capabilities [5]. Deep GD models can produce a large amount of new 2D or 3D data [6–9] given a set of training data, which has shown promises in computer graphics and computer vision. In the design field, deep GD models, like variational auto-encoder (VAE) [10] and generative adversarial network (GAN) [11], have been used to assist human designers in generating novel and realistic designs [12–14] for design conceptualization. Building upon existing models, we develop a data-driven GD approach in this study for product shape generation based on deep neural networks. Our assumption is that existing product designs (e.g., cars, chairs, tables, etc.) on the market must have gone through a complete design cycle, so both their appearances or functionalities are optimized. Using a deep GD approach with existing designs as training data, it is expected that the generated design candidates would be promising ones. Also, learning-based GD methods have the potentials to reduce the high dependencies on design expertise because machines learn design knowledge in advance, which will assist the designer in realizing design automation.

However, realizing this idea in engineering design is challenging. In engineering design, a product is a system that consists of interconnected components. Traditionally, the design of such systems starts from the system requirements analysis and is driven by a top-down hierarchical decomposition, followed by the design of subsystems and components. Each component in a system is first designed separately and finally integrated into a complete system and validated against system-level requirements. Most existing GD [15–20] are focused on generating holistic 3D shapes without considering the structural dependencies or relations between components (e.g., a car is treated as a whole piece instead of dividing it into the car body, mirrors, etc.). Even if efforts have been taken to generate part-aware 3D shapes [6, 21, 22], they ignore the evaluation of the engineering performance of the generated 3D shapes. However, assessing the

* Corresponding author

URI: https://hdl.handle.net/10125/71258
978-0-9981331-4-0
(CC BY-NC-ND 4.0)

engineering performance of a product is essential in engineering design.

Nowadays, most industries use computer-aided engineering (CAE) tools, such as finite element analysis (FEA) and computational fluid dynamics (CFD) software, to evaluate the engineering performance of a preliminary design before the physical prototyping and testing. Nonetheless, the engineering evaluation is costly. For instance, the assessment of the aerodynamic performance of a 3D car model using CFD software could take hours. Therefore, it is impractical to evaluate every single design candidate, let alone the vast number of design alternatives obtained from GD models. A fast engineering evaluation method is needed in realizing GD in engineering design.

To address these challenges, this study develops a new GD approach for engineering systems design that integrates fast engineering evaluation and deep generative models that allows the generation of part-aware 3D meshes. We validate and demonstrate the effectiveness of our approach through an aerodynamic car design problem. The remainder of this paper is organized as below. Section 2 gives a literature review of the relevant research. The details of the proposed approach are introduced in Section 3. Section 4 presents the results and discussion, and the paper is concluded in Section 5, in which we also summarize the closing insights and future work.

2. Literature Review

The review presented in this section is relevant to the literature in the fields of intelligent design agents, deep generative models, 3D shape synthesis, and data-driven CFD evaluation methods.

2.1. Intelligent design agents

Rules-driven parametric design tools have introduced the generative design module to enable automatic design exploration. Users usually set the constraints and requirements for their designs, and those tools can then run hundreds of simulations to generate various designs for users to select. There are also learning-based/data-driven design platforms. For example, Hu and Taylor [3] developed a CAD platform with an intelligent tutoring system. The system can first learn all possible ways to design a 3D model and then instruct users to draw 3D models.

In rules-driven parametric design tools, setting proper constraints and requirements for the design requires a high degree of domain expertise, which is not friendly to novice designers or fast design evaluation. In contrast, data-driven approaches learn from existing knowledge or data in advance and then can allow users with less experience to create designs. Our approach applies a deep generative model as an intelligent agent, which learns existing designs on the market to generate a large number of designs for users.

2.2. Deep generative models

Deep generative models aim at synthesizing new samples using the distribution learned from the existing data. The strategy of deep generative models is trying to approximate a distribution as similar to the true data distribution as possible by using a multi-layer of neural networks. GAN [11] and VAE [10] are the two most widely used deep generative models.

GAN consists of two parts: a generator and a discriminator. The discriminator tries to distinguish the data generated by the generator from the training data. In contrast, the generator aims to fool the discriminator with data that are highly similar to the training data. They compete with each other in the training process, driving the generator to produce data as identical to the training data as possible. GANs have achieved success in 2D and 3D visualizations and reconstructions [7,8]. However, since the discriminator judges the generated data based on distance metrics, the generator can synthesize unrealistic data (e.g., a face with a displaced mouth).

The basic idea of VAE is to find a hidden representation of the training data using low-dimensional latent variables. Those latent variables contain information like specific structural and semantic properties of the training data. Compared to GANs, the training of VAEs is faster and easier via backpropagation, thus gaining increasing popularity [23]. VAEs have also been successfully applied in both 2D images [9] and 3D models [6].

2.3. 3D shape synthesis

The increasing availability of large 3D shape datasets, like ShapeNet [24], provides a large amount of training. the deep generative methods have been applied in object detection [25,26], classification [27,28], and semantic segmentation [29,30]. They can also generate diverse 3D objects in various representations, such as point cloud [15,16], voxels [17–19], and meshes [20,31]. Compared to point cloud and voxels, meshes can better capture the geometric details (e.g., smoothness, curvature) of 3D objects without consuming large storage space. So, the mesh representation is more suitable for engineering design applications (e.g., the representation of automobiles).

However, 3D meshes from open-source datasets are usually non-manifold triangles that may contain holes, inverted faces, and self-intersection. Those data are not suitable for the input of deep generative models, and thus, it is necessary to construct a consistent representation of such unstructured data. Umetani [12]

proposes a novel parameterization method leveraging depth map and shrink wrapping [32]. He deforms a cube surface to approximate the surface of each 3D car shape resulting in quad meshes with consistent connectivity. However, that method is hard to deal with highly concave shapes, so the author needs to manually delete parts, e.g., mirrors and spoilers, and only keep the car body. Umetani and Bickel [13] further extend [12] by deforming a PolyCube that is similar to a shape in a coarse resolution, which enables the method to parameterize a wider variety of shapes. However, the method is incapable of dealing with shapes containing several parts. Gao et al. [6] propose a part-aware mesh generation approach. With a set of 3D shapes with part-level labels, the authors apply the non-rigid registration [33] to deform a box mesh to approximate each part. The method can finally generate part-aware 3D meshes with finer-scale geometry. Since creating 3D shapes in a part-aware manner addresses component-level dependencies and the generated 3D meshes contain geometric details for the ease of engineering evaluation, we adopt the two-level VAE structure used in [6] to build our 3D mesh generative design approach.

2.4. Data-driven CFD evaluation

Engineering performance evaluation is critical to engineering design and optimization, but it is usually computationally expensive. For example, the CFD evaluation of the aerodynamic performance of a 3D automobile model could take hours. To accelerate the CFD evaluation process, OmniAD [34] proposed a spherical harmonics based method using data from pictures that capture the falling motion of objects to predict the movement of an object moving within the air. Similarly, Baque et al. [35] applied graph convolution neural networks to predict the pressure distribution on free-form 3D models. These methods focus on the prediction of the motion of objects or the airflow around a moving object, which might be difficult for a designer to understand without specific expertise and knowledge.

On the other hand, Umetani and Bickel [13] present a method using Gaussian Process to predict airflow around a 3D object, which can predict the drag coefficient in real-time. In [36], Gunpinar et al. apply PCA and regression methods to build a mathematical model to obtain the prediction of drag coefficients of silhouettes of cars. Such predictions are straightforward to show the aerodynamic performance of a moving object and are easily understood by the designer. While Badias et al. [37] use the LLE algorithm [38] to estimate the airflow around a car, we propose to use LLE to predict the drag coefficients of newly generated car models from the 3D mesh

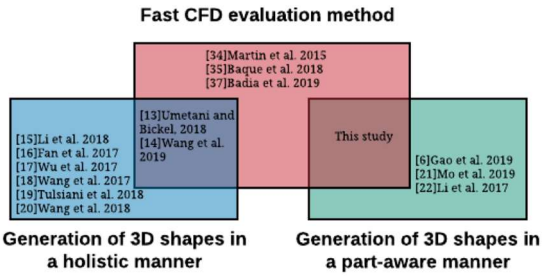

Figure 1. Synthesis of the research gap

generative design module. The reasons for adopting LLE in this study are 1) LLE has the potentials to estimate the drag coefficient of a car model with good accuracy; 2) the estimation of the drag coefficient of a car model using LLE is fast.

2.5. The research gap

As shown in Figure 1, there is a lack of research in the integration of the generation of part-aware 3D shapes and fast CFD evaluation method for GD. This study provides solutions to enhancing human-AI collaboration in the engineering design field beyond the conceptual design phase. The knowledge that the deep generative model learns from the existing designs can augment designers' creativity and engineering decision-making by exploring more design alternatives with the awareness of their engineering performance. To the best of our knowledge, this is the first study of integrating a deep generative design model that can produce part-aware 3D shapes with a fast CFD evaluation method. The validity of our approach is demonstrated in the aerodynamic automobile design.

3. The Research Approach

In this section, we present our overall approach. As shown in Figure 2, the proposed design approach is

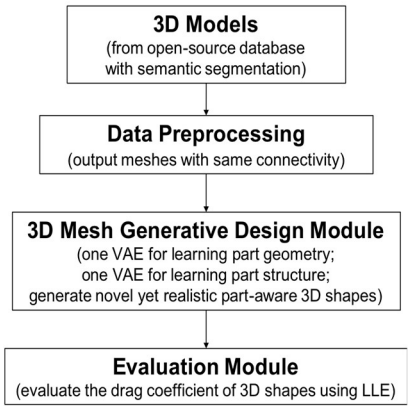

Figure 2. The part-aware product design approach

mainly composed of two modules: a generative design module and an evaluation module, in addition to a data collection and preprocessing process.

For the generative design module, we adopt a deep generative method based on VAE to learn the design concepts and geometries from existing products to generate a variety of novel designs. The data-driven generative method can take advantage of existing successful designs instead of designing from scratch. Inspired by SDM-NET [6], we apply two VAEs, namely PartVAE and SPVAE. PartVAE is for learning individual part geometries, and SPVAE is for learning the part geometries and the part structure of the 3D models. For the 3D shape representation, we chose the mesh for capturing surface details of automobile parts (e.g., body and mirrors) with less storage space. For the evaluation module, we apply LLE [38] to predict the drag coefficient of a 3D car model in real-time.

3.1. Data preprocessing

3D models from open databases, like Shapenet [24], are often unstructured. Also, some models may contain interior structures (e.g., seats, steering wheel in a car) that are not desired since we focus on the external geometry. Such data cannot be directly used in generative models without preprocessing. Meshes are usually voxelized, transferred to point-cloud, or re-meshed according to the structures of different generative models. Additionally, we need to segment a holistic 3D model into several parts so that each part can be learned in the generative module.

After segmentation, we first calculate the bounding box of each part. A unit cube mesh with 19.2k triangles (9602 vertices) is then used to represent each part and deformed to fit the bounding box of this part, as suggested in [6]. This step forms a coarse representation of a 3D model composing of several deformable boxes. We then apply the non-rigid registration [33] to transform each deformable box to approximate its corresponding part producing a mesh with fine geometric details. Non-rigid registration is widely used in computer graphics to map one point set to another. Figure 3 shows an example of the preprocessing of a car model that is segmented into one car body, two mirrors, and four wheels.

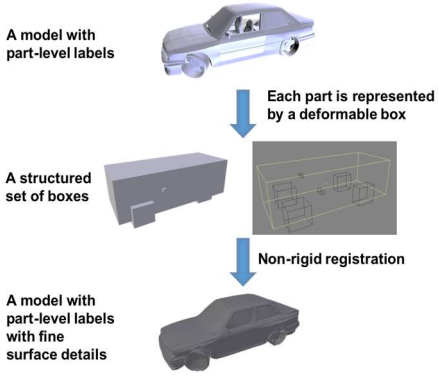

Figure 3. An example of the data preprocessing of 3D car models

3.2. 3D mesh generative design module

The generative design module is used to learn part geometry and structure information (i.e., how all parts are connected) so that it can generate new part-aware designs. We adopt a two-level VAE structure, namely PartVAE and SPVAE, from SDM-NET [6]. In what follows, we introduce how the adopted VAE structure can generate part-aware 3D shapes.

3.2.1. Encoding of a part. A vector $\boldsymbol{v}$ is used to encode the part geometry and the structures of all parts. The vector is composed of seven sections $\boldsymbol{v_1}$-$\boldsymbol{v_7}$.

- $\boldsymbol{v_1} \in \{0,1\}$ shows if this part exists or not;
- $\boldsymbol{v_2} \in \{0,1\}^n$ is a n-dimension vector to show which parts this part supports;
- $\boldsymbol{v_3} \in \{0,1\}^n$ is a n-dimension vector to show that this part is supported by which part;
- $\boldsymbol{v_4} \in \mathbb{R}^3$ is the x, y and z coordinates of the geometric center of the bounding box;
- $\boldsymbol{v_5} \in \{0,1\}$ shows if the symmetric part of this part exists or not;
- $\boldsymbol{v_6} \in \mathbb{R}^4$ shows the coefficients a, b, c and d of the mathematical representation of the symmetric plane $ax + by + cz + d = 0$;
- $\boldsymbol{v_7} \in \mathbb{R}^{64}$ is the latent vector for encoding the geometry of this part,

where n indicates the number of parts that a model consists of. For example, $n = 7$ because a car model is segmented into one body, two mirrors, and four wheels. The structure of parts is encoded by recording the information of each part using $\boldsymbol{v_2}$, $\boldsymbol{v_3}$ for support relations, and $\boldsymbol{v_5}$, $\boldsymbol{v_6}$ for symmetry relation. It should be noted that the method doesn't require the models to be symmetric, but symmetry is beneficial to machine learning of geometric structures.

3.2.2. PartVAE. PartVAE uses a convolutional VAE architecture. It consists of an encoder and a decoder, and it takes a matrix with a dimension of 9602×9 as input. As mentioned in Section 3.1, 9602 is the number of the vertices of the bounding box mesh. Each row of the matrix is a vector with a dimension of 9 that records the information (i.e., rotation axis, rotation angle, and scaling factor) of the 1-ring neighborhood of each vertex. The encoder consists of two convolutional layers and a fully connected layer. The output of the second convolution layer is reshaped by

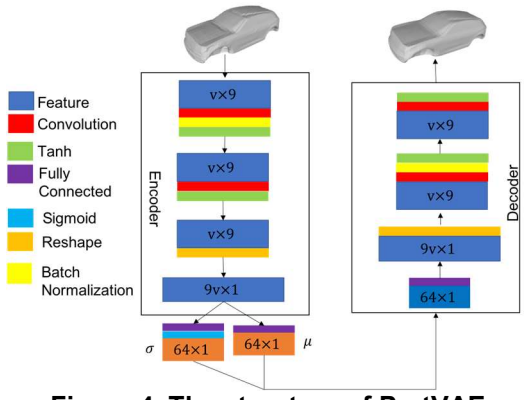

Figure 4. The structure of PartVAE

the fully connected layer to a 64-dimensional latent vector. The decoder has a mirrored structure of the encoder. Figure 4 shows the structure details. PartVAE is trained through minimizing the following loss:

$$L_{PartVAE} = L_{recon} + \lambda_1 L_{KL} + \lambda_2 L_{Reg}, \quad (1)$$

where L_{recon} denotes the mean square error reconstruction loss for better reconstruction, L_{KL} is the KL divergence to promote Gaussian distribution in the latent space, and L_{Reg} is the squared l_2 norm of the network parameters to avoid overfitting. λ_1 and λ_2 are weights for corresponding loss terms.

3.2.3. SPVAE. SPVAE is for joint learning of the part structure and part geometry, which makes sure the geometry of the generated shape is compatible with the structure. SPVAE takes as input the concatenated feature vector of all parts of a model, containing information of the part geometry and the structures of parts. In the encoder, the vector goes through three fully connected layers of dimension 1024, 512, and 256, respectively, and is translated into a latent vector with a dimension of 128. The decoder consists of four fully connected layers with dimensions of 128, 256, 512, and 1024, respectively. The output of the decoder is a concatenated feature vector and can be translated into several parts. Those parts can then be merged together to form a holistic shape. Figure 5 shows the details of the structure of SPVAE. SPVAE minimizes the following loss function:

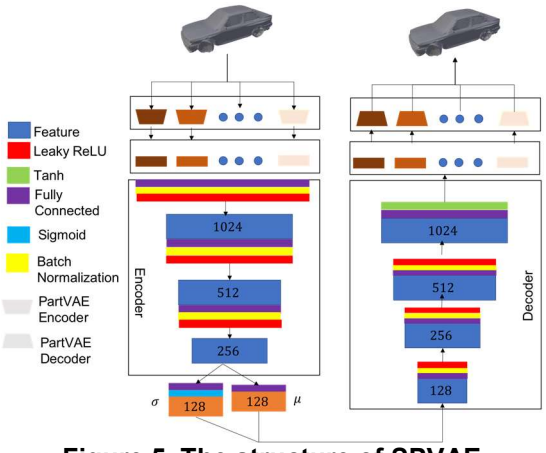

Figure 5. The structure of SPVAE

$$L_{SPVAE} = \alpha_1 D_{recon} + \alpha_2 D_{KL} + D_{Reg}. \quad (2)$$

where D_{recon} denotes the mean square error reconstruction loss, D_{KL} is the KL divergence, and D_{Reg} is the squared l_2 norm. α_1 and α_2 are weight parameters for corresponding loss terms. After the PartVAE and the SPVAE are trained, a 128-dimension latent space is learned in the SPVAE, which enables the network to perform generation tasks, like shape interpolation and random shape generation.

3.3. Evaluation module

Aerodynamic performance analysis of cars can be very time-consuming using traditional CFD tools. To tackle the challenge and accelerate the design evaluation process, we propose to apply the Locally Linear Embedding (LLE) algorithm [38] to approximate the drag coefficients of cars. In car design, the drag coefficient is one of the most important performance metrics, which primarily affects a car model's fuel efficiency. Generally, with the balance of other considerations, a smaller drag coefficient is more favorable. In the following sections, we discuss how we apply LLE and how we prepare the data of car models for fast evaluation.

3.3.1. Application of LLE. LLE holds the assumption that the data (e.g., car shapes) are located in a manifold space and the fact that the manifold is homeomorphic to flat data space. This can be understood using an analogy that a Swiss Roll shape can be extended to a flat piece that is homeomorphic to the Swiss Roll. Then LLE assumes that every data point can be approximated by linear interpolation of its nearest neighboring data in a reasonably accurate manner. The number of the nearest neighbors M is a modifiable parameter that can be defined by users. This linear interpolation is expressed by Equation (3).

$$\boldsymbol{\nu_m} = \sum_{\boldsymbol{\nu_i} \in S_m} W_{mi} \boldsymbol{\nu_i} \quad (3)$$

where $\boldsymbol{\nu_m}$ denotes a vector representing a data point, W_{mi} denotes the unknown weights, $\boldsymbol{\nu_i}$ denotes vectors representing all the neighbors of $\boldsymbol{\nu_m}$, and S_m is the set of the M-nearest neighbors of $\boldsymbol{\nu_m}$.

$$\phi(\boldsymbol{W}) = \sum_{m=1}^{M} \left\| \boldsymbol{\nu_m} - \sum_{i=1}^{M} W_{mi} \boldsymbol{\nu_i} \right\|_2^2 \quad (4)$$

The unknown weights W_{mi} will be obtained by minimizing equation (4). W_{mi} is zero if $\boldsymbol{v_i} \notin S_m$, i.e., car i is not one of the M nearest neighbors of car m. LLE also tries to project the set of high dimensional vectors to low dimensional space while conserving the manifold structure. If a car shape can be represented by a vector $\boldsymbol{v_m} \in \mathbb{R}^D$ and the LLE algorithm can then be applied to project $\boldsymbol{v_m}$ to a lower-dimensional space $\mathbb{R}^d$ with $d \ll D$. Then a new vector $\boldsymbol{x} \in \mathbb{R}^d$ can be found for each car $\boldsymbol{v} \in \mathbb{R}^D$. LLE states that $\boldsymbol{x}$ can be calculated by

$$(x_1, \ldots, x_M) = \arg\min \sum_{m=1}^{M} \left\| x_m - \sum_{i=1}^{M} W_{mi} x_i \right\|_2^2 \qquad (5)$$

where W_{mi} representing the neighboring relation between data points remains unchanged. When an original database (manifold) is built, a new vector can be easily embedded onto the manifold to find its neighbors, and the corresponding weighs W_i. The key idea of employing LLE to approximate the drag coefficients of a car is that we assume that the drag coefficient of a car model can be obtained through a linear combination of the drag coefficients of its neighboring cars in the manifold of 3D car shapes. The drag coefficient of a new car can be approximated using Equation (6).

$$\mathrm{C_d^{new}} \approx \sum_{i=1}^{M} W_i C_d^i \qquad (6)$$

where C_d^i represents the drag coefficient of the ith neighbor of the new car model and W_i is the corresponding weight.

With Equation (6), an approximation of the drag coefficient of a car can be obtained, which gives the designer a quick reference to the aerodynamic performance of the car. Results in Section 4 shows that the evaluation time is significantly reduced with a little loss in the accuracy of the value of the drag coefficient.

3.3.2. Characterization of car models. 3D car models should be represented by vectors so that they can be used as data points in LLE. We randomly select 60 car models from the training dataset (1161 car models) to form the original database for the LLE method. We manually categorize those models into four groups: small cars (e.g., sedans, coupes.); big cars (e.g., SUVs, miniVANs), classic cars; and sports cars based on the similarity of their appearance. Then the bounding box of each car model is calculated. The biggest size of the bounding box is $0.9(L) \times 0.3(H) \times 0.4(W)\ m^3$. We resize each model with a factor of 5^3 to match the size of all 60 car models to approximately the size of real cars.

A vector with only values of 0 and 1 is used to represent each car model. We first embed each car model into a grid of $50 \times 20 \times 20$ points uniformly distributed on a volume of $5 \times 2 \times 2\ m^3$, setting the mass center of each car model to the center of the grid. We then apply the signed distance field [37] to get the signed distance of each point to the surface of a car model. The distance value is positive, zero, or negative when a point is outside, on, or inside the surface, respectively. We then use the following presence function to transform each signed distance to 0 or 1.

$$\phi(x) = \begin{cases} 1, if\ x \in \Omega \\ 0, if\ x \notin \Omega \end{cases} \qquad (7)$$

where Ω represents the space inside and on the boundary of a car model, and x represents each point in the grid. In this way, we can characterize a car model to a vector $\boldsymbol{r} \in \mathbb{R}^D$, where $D = 50 \times 20 \times 20 = 20000$. Finally, we get a matrix $\boldsymbol{A} \in \mathbb{R}^{60 \times D}$, of which every row is a vector $\boldsymbol{r}$ representing a car model.

3.3.3. CFD evaluation of car models. We apply CFD software, ANSYS Fluent, to get the drag coefficients of 60 car models. The size of the enclosure (simulation for wind tunnel) is set as $3L \times 2H \times 2W$, where L, H, W represent the length, height, and width of the bounding box of the biggest car model, respectively. Figure 6 shows an example of the setting of a car model. The side with green color on the right-hand side is set as the inlet, and its opposite side is set as the outlet. The car model is placed on the ground and $0.5L$ away from the inlet side. As suggested in [36], the boundary conditions set for all surfaces are as follows for all car models: a) Inlet: constant velocity of $40m/s$; b) Outlet: 0 Pa constant pressure; and 3) Remaining surfaces: non-slip. We choose the standard $k-\epsilon$ model for the turbulence model, the SIMPLE algorithm for the pressure-velocity coupling, and 10^{-3} for the convergence criterion for all flow variables. We run the calculation until the result converges and calculate the drag coefficient of each car model. The result is available in Section 4.

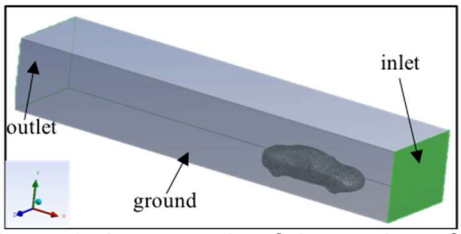

Figure 6. An example of the setting of a 3D car model for CFD analysis

3.3.4. Fast evaluation of the drag coefficient of a new car model. After we obtain the drag coefficients of all 60 car models, we build an original database that consists of the vector representations (Matrix $\boldsymbol{A}$) and the drag coefficients of 60 car models. When a new car

model comes, it is first characterized to a vector $\boldsymbol{r}$ using the method described in Section 3.3.2, and then it is embedded into the manifold. The neighbors of the vector and the corresponding weight for each neighbor can then be calculated. Based on our assumption, the drag coefficient of the new car can be calculated by the linear interpolation of the drag coefficients of its neighbors using Equation (6).

4. Results and discussion

4.1. Data preprocessing

The original dataset consists of 1824 car models with consistent segmentation from SDM-NET [6]. In this study, only daily commute cars (e.g., sedans, SUVs) are considered, and we exclude certain car models, including buses and Formula One cars. This gives us a total of 1161 car models, which are then preprocessed for the 3D mesh generative module. The most time-consuming part of the preprocessing is the non-rigid registration. It takes about 2 minutes for one part in a computer with a Windows operation system, an i7 8700 CPU, and 8GB RAM (*Computer 1*).

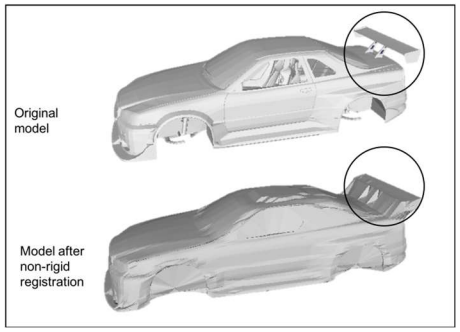

Figure 7. Registration of a car body

While the non-rigid registration method can produce watertight mesh for each part with fine geometry details, it still has some limitations in handling the shapes with non-genus-zero topology. Genus is a topological property, and non-genus-zero topology can be simply considered as a surface with at least one hole. In this case, the method manages to keep the outer geometry without maintaining the holes. Figure 7 shows an example of the registration of the body of a 3D car model. The method preserves the external geometry of the car body without maintaining the topology of the spoiler marked by the cycles.

4.2. 3D mesh generative design module

The training dataset of 1161 car models is randomly split into the training data (75%) and validation data (25%). The Chamfer distance is used to measure the reconstruction loss on the validation data. We set $\lambda_1 = 1.0,\ \lambda_2 = 0.01, \alpha_1 = 1.0, \alpha_2 =$ 0.5 for the weight parameters, as suggested in [6]. The ADAM optimization method [39] is used for both VAEs. We train the PartVAE 20000 iterations and SPVAE 40000 iterations separately. The batch size is set at 128, and the initial learning rate is 0.001, which decays every 2000 steps with a decay rate of 0.8. The training of both PartVAE and SPVAE using 1161 car models takes around 100 hours in a computer with a Linux operation system, an i7 6850K CPU, 64GB RAM, and two GTX 1080Ti GPUs (*Computer 2*).

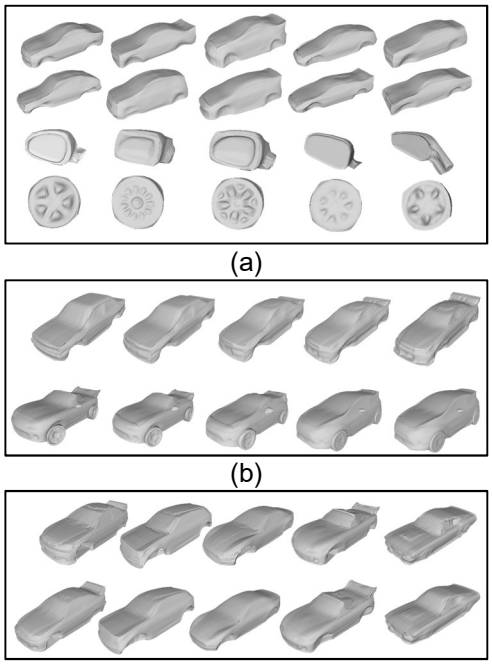

Figure 8. (a) Parts from random generation. (b) Car bodies and merged car models from shape interpolation. (c) Car bodies from shape reconstruction

When the training is done, the networks can perform shape reconstruction, random shape generation, and shape interpolation for separate parts. Random generation can be done by randomly sampling latent vectors from the learned latent space (Gaussian distribution) and then decoding them into 3D parts. We can get a latent vector by having a part go through the encoder and then decode the vector to a part similar to the original part, which shows the process of shape reconstruction. It takes about 0.9 seconds to generate one part in *Computer 2* for random generation, for instance. After we got separate parts, we can merge seven parts into a holistic car.

Figure 8(a) shows several parts from random generation. The model can generate varied shapes for different car parts that look reasonable in terms of visual appearance. Figure 8(b) shows some car bodies

and merged car models from shape interpolation. The first and the last column are the shapes to be interpolated. The in-between columns are the linear interpolated shapes. We can observe a gradual transition of the geometry between the first and the last shapes. Figure 8(c) shows several car bodies from shape reconstruction. The first row is the original models, and the second row is their corresponding reconstructed models. Theoretically, we can sample as many latent vectors as possible from the latent space for random generation. We can also perform shape interpolation between every pair of car models from the training dataset with any number of in-between interpolations. Thus, we can generate thousands of novel car models through the trained networks.

The results show that the deep generative model can effectively generate novel yet realistic car models in terms of appearance with part-aware information. It can provide the designer with many good references for the design. It can also enable the designer to design a car shape in a down-top systematic way, which aligns with the tradition in engineering design.

4.3. Evaluation module

We get the vector representations and drag coefficients of 60 cars to build the original database. These car models are categorized into four types: Small cars, 20; Big cars, 19; Classic cars, 8; Sports cars, 13, each of which should cluster together if the assumption of LLE works for car shapes.

LLE can project high dimensional vectors on a manifold to low dimensional space while preserving the manifold structure. The number of the nearest neighbors M is a customized parameter. Through trial-and-error, it is found that when M =6, the lowest reconstruction error is obtained. We then project those vectors to a two-dimension space, which is shown in Figure 9. Three clusters (i.e., big cars, small cars, and sports cars) can be observed with several exceptions that might have transitional shapes (e.g., some

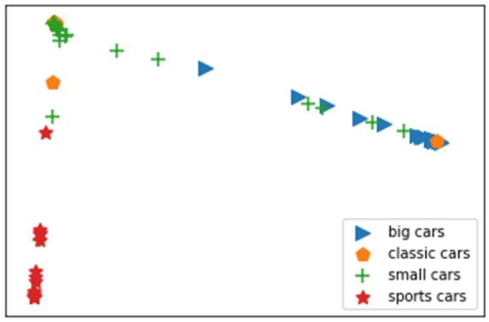

Figure 9. Projection of high dimensional vectors of car shapes to $\mathbb{R}^2$

Table 1. Drag coefficients of 60 car models

No.	Cd	Type	No.	Cd	Type	No.	Cd	Type
1	0.58	big	21	0.43	classic	41	0.34	small
2	0.68	big	22	0.65	classic	42	0.33	small
3	0.45	big	23	0.45	classic	43	0.25	small
4	0.52	big	24	0.44	classic	44	0.32	small
5	0.4	big	25	0.64	classic	45	0.36	small
6	0.37	big	26	0.67	classic	46	0.3	small
7	0.45	big	27	0.41	classic	47	0.3	small
8	0.49	big	28	0.35	small	48	0.33	sport
9	0.39	big	29	0.39	small	49	0.35	sport
10	0.55	big	30	0.29	small	50	0.72	sport
11	0.44	big	31	0.33	small	51	0.48	sport
12	0.44	big	32	0.34	small	52	0.37	sport
13	0.52	big	33	0.34	small	53	0.51	sport
14	0.52	big	34	0.54	small	54	0.49	sport
15	0.35	big	35	0.39	small	55	0.61	sport
16	0.41	big	36	0.38	small	56	0.6	sport
17	0.34	big	37	0.31	small	57	0.49	sport
18	0.4	big	38	0.33	small	58	0.71	sport
19	0.32	big	39	0.29	small	59	0.53	sport
20	0.24	classic	40	0.28	small	60	0.34	sport

Crossover SUVs are similar to both sedans and SUVs) between two clusters. Classic cars whose shapes do not follow any specific pattern are mixed in other clusters. This indicates that the proposed vectorization method captures the key features of different car models, and LLE can properly cluster them.

The drag coefficients of the 60 car models are obtained using ANSYS Fluent in a high-performance computer (*Computer 3*) with an Intel Xeon Silver 4114 CPU. It takes about 30 minutes to evaluate a car model when 16 processors are used in parallel. Table 1 shows the results, including both Cd (drag coefficients) and car types.

To test the accuracy of the approximation method using LLE, we randomly select 20 merged car models from the generative design module and compare their drag coefficients using LLE against those obtained from Ansys Fluent by computing the relative percent error (δ). We achieve an overall average percent error ($\bar{\delta}$) of 11.5%, with a standard deviation (s) of 8.3%. Figure 10 shows the results of δ of each car model and $\bar{\delta}$ in four groups. The reason why the accuracy is not ideal could be that the sample size of 60 cars is relatively small, which cannot fully represent the manifold of car shapes. Thus, the estimation of a new car model's drag coefficient does not refer to the proper neighbors in the manifold. In addition, LLE leads to relatively larger errors towards certain car models (e.g., 29.6% for a classic car), which could bias a designer's decision. To address this limitation, we plan to improve the accuracy by trying different ways, like increasing the size of the original database in future work. But with the sacrifice on the accuracy, the time of evaluation is significantly saved (from hours to seconds) so that designers are able to have a quick reference to rule out those underperformed design recommendations and select several promising

designs for further development and optimization. This could greatly accelerate the entire product design and development process.

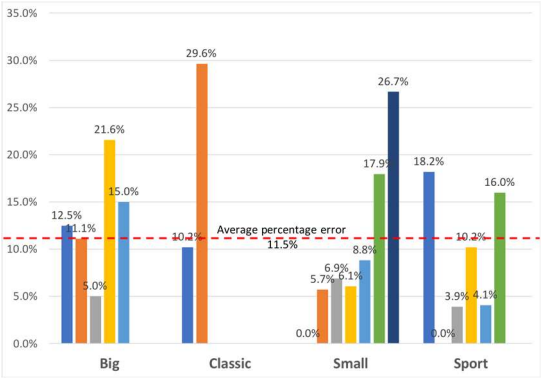

Figure 10. Results of the relative percent error of each car model and overall average percent error in four groups

5. Conclusion

Although the generative design has gained a lot of attention recently in human-AI collaboration, research on the integration of part-aware generative methods and fast evaluation for engineering design is still lacking. In this paper, we introduced an approach that integrates a part-aware deep generative model based on VAE with a fast engineering performance evaluation method using LLE. The approach enables the user to explore thousands of part-aware 3D models automatically generated by the GD-based design agent and can quickly get their engineering performance at the cost of losing a certain degree of accuracy. We acknowledge the limitation of the fast evaluation method in terms of the accuracy given the current sample size, but this method significantly reduces the evaluation time, thus facilitates the design process.

In the future, we plan to improve this method by increasing the size of the original database, adjusting the density of the point grid used for the vectorization of car models. LLE is only applied to one performance metric (i.e., drag coefficient) of a car in the case study, but we believe that the method can be applied to other product evaluations if the performance metrics are related to the geometry of a product. The scope could also be enlarged by evaluating more performance metrics of a system so that research about tradeoff between different performance metrics can be carried out. We also plan to add a shape optimization module to the approach to enable optimal 3D shape design in support of the design decision-making, considering the tradeoff between part performance and system-level performance for the reason that the combination of optimal parts might not lead to an optimal system. We also consider developing a GUI to support human-AI interactive design exploration and optimization.

6. Acknowledgment

The authors gratefully acknowledge the financial support from the NSF grant DUE-1918847. We also appreciate Dr. Miaoqing Huang for granting us access to *Computer 2* with GPUs. Editorial support provided by Laxmi P. Poudel is also much appreciated.

7. References

[1] Dellermann, D., Calma, A., Lipusch, N., Weber, T., Weigel, S., and Ebel, P., 2019, "The Future of Human-AI Collaboration: A Taxonomy of Design Knowledge for Hybrid Intelligence Systems," *Proceedings of the 52nd Hawaii International Conference on System Sciences*.

[2] Autodesk, "Generative Design in Autodesk" [Online]. Available: https://www.autodesk.com/solutions/generative-design. [Accessed: 19-Jun-2020].

[3] Hu, Y., and Taylor, M. E., "A Computer-Aided Design Intelligent Tutoring System Teaching Strategic Flexibility."

[4] Kumar, R., Ai, H., Beuth, J. L., and Rosé, C. P., 2010, "Socially Capable Conversational Tutors Can Be Effective in Collaborative Learning Situations," *International Conference on Intelligent Tutoring Systems*, Springer, pp. 156–164.

[5] Shea, K., Aish, R., and Gourtovaia, M., 2005, "Towards Integrated Performance-Driven Generative Design Tools," Autom. Constr., **14**(2), pp. 253–264.

[6] Gao, L., Yang, J., Wu, T., Yuan, Y. J., Fu, H., Lai, Y. K., and Zhang, H., 2019, "SDM-NET: Deep Generative Network for Structured Deformable Mesh," ACM Trans. Graph., **38**(6).

[7] Radford, A., Metz, L., and Chintala, S., 2015, "Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks," arXiv Prepr. arXiv1511.06434.

[8] Berthelot, D., Schumm, T., and Metz, L., 2017, "Began: Boundary Equilibrium Generative Adversarial Networks," arXiv Prepr. arXiv1703.10717.

[9] Rezende, D. J., Mohamed, S., and Wierstra, D., 2014, "Stochastic Backpropagation and Approximate Inference in Deep Generative Models," arXiv Prepr. arXiv1401.4082.

[10] Kingma, D. P., and Welling, M., 2013, "Auto-Encoding Variational Bayes," arXiv Prepr. arXiv1312.6114.

[11] Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y., 2014, "Generative Adversarial Nets," *Advances in Neural Information Processing Systems*, pp. 2672–2680.

[12] Umetani, N., 2017, "Exploring Generative 3D

Shapes Using Autoencoder Networks," *SIGGRAPH Asia 2017 Technical Briefs*, ACM, p. 24.

[13] Umetani, N., and Bickel, B., 2018, "Learning Three-Dimensional Flow for Interactive Aerodynamic Design," ACM Trans. Graph., **37**(4), p. 89.

[14] Wang, J., and He, Y., 2019, "Physics-Aware 3D Mesh Synthesis," *2019 International Conference on 3D Vision (3DV)*, IEEE, pp. 502–512.

[15] Li, K., Pham, T., Zhan, H., and Reid, I., 2018, "Efficient Dense Point Cloud Object Reconstruction Using Deformation Vector Fields," *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 497–513.

[16] Fan, H., Su, H., and Guibas, L. J., 2017, "A Point Set Generation Network for 3d Object Reconstruction from a Single Image," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 605–613.

[17] Wu, J., Wang, Y., Xue, T., Sun, X., Freeman, B., and Tenenbaum, J., 2017, "Marrnet: 3d Shape Reconstruction via 2.5 d Sketches," *Advances in Neural Information Processing Systems*, pp. 540–550.

[18] Wang, P.-S., Liu, Y., Guo, Y.-X., Sun, C.-Y., and Tong, X., 2017, "O-Cnn: Octree-Based Convolutional Neural Networks for 3d Shape Analysis," ACM Trans. Graph., **36**(4), pp. 1–11.

[19] Tulsiani, S., Efros, A. A., and Malik, J., 2018, "Multi-View Consistency as Supervisory Signal for Learning Shape and Pose Prediction," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2897–2905.

[20] Wang, N., Zhang, Y., Li, Z., Fu, Y., Liu, W., and Jiang, Y.-G., 2018, "Pixel2mesh: Generating 3d Mesh Models from Single Rgb Images," *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 52–67.

[21] Mo, K., Guerrero, P., Yi, L., Su, H., Wonka, P., Mitra, N., and Guibas, L. J., 2019, "Structurenet: Hierarchical Graph Networks for 3d Shape Generation," arXiv Prepr. arXiv1908.00575.

[22] Li, J., Xu, K., Chaudhuri, S., Yumer, E., Zhang, H., and Guibas, L., 2017, "Grass: Generative Recursive Autoencoders for Shape Structures," ACM Trans. Graph., **36**(4), pp. 1–14.

[23] Doersch, C., 2016, "Tutorial on Variational Autoencoders," arXiv Prepr. arXiv1606.05908.

[24] Chang, A. X., Funkhouser, T., Guibas, L., Hanrahan, P., Huang, Q., Li, Z., Savarese, S., Savva, M., Song, S., and Su, H., 2015, "Shapenet: An Information-Rich 3d Model Repository," arXiv Prepr. arXiv1512.03012.

[25] Zhu, C., He, Y., and Savvides, M., 2019, "Feature Selective Anchor-Free Module for Single-Shot Object Detection," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 840–849.

[26] He, Y., Zhu, C., Wang, J., Savvides, M., and Zhang, X., 2019, "Bounding Box Regression with Uncertainty for Accurate Object Detection," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2888–2897.

[27] Qi, C. R., Su, H., Nießner, M., Dai, A., Yan, M., and Guibas, L. J., 2016, "Volumetric and Multi-View Cnns for Object Classification on 3d Data," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5648–5656.

[28] Krizhevsky, A., Sutskever, I., and Hinton, G. E., 2012, "Imagenet Classification with Deep Convolutional Neural Networks," *Advances in Neural Information Processing Systems*, pp. 1097–1105.

[29] Qi, C. R., Yi, L., Su, H., and Guibas, L. J., 2017, "Pointnet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space," *Advances in Neural Information Processing Systems*, pp. 5099–5108.

[30] He, K., Gkioxari, G., Dollár, P., and Girshick, R., 2017, "Mask R-Cnn," *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2961–2969.

[31] Groueix, T., Fisher, M., Kim, V. G., Russell, B. C., and Aubry, M., 2018, "A Papier-Mâché Approach to Learning 3d Surface Generation," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 216–224.

[32] Kobbelt, L. P., Vorsatz, J., Labsik, U., and Seidel, H., 1999, "A Shrink Wrapping Approach to Remeshing Polygonal Surfaces," *Computer Graphics Forum*, Wiley Online Library, pp. 119–130.

[33] Zollhöfer, M., Nießner, M., Izadi, S., Rehmann, C., Zach, C., Fisher, M., Wu, C., Fitzgibbon, A., Loop, C., and Theobalt, C., 2014, "Real-Time Non-Rigid Reconstruction Using an RGB-D Camera," ACM Trans. Graph., **33**(4), pp. 1–12.

[34] Martin, T., Umetani, N., and Bickel, B., 2015, "OmniAD: Data-Driven Omni-Directional Aerodynamics," ACM Trans. Graph., **34**(4), pp. 1–12.

[35] Baque, P., Remelli, E., Fleuret, F., and Fua, P., 2018, "Geodesic Convolutional Shape Optimization," arXiv Prepr. arXiv1802.04016.

[36] Mustafa, E. G. U. C. C., and Gunpinarc, O. S., "A Generative Design and Drag Coefficient Prediction System for Sedan Car Side Silhouettes Based on Computational Fluid Dynamics."

[37] Badías, A., Curtit, S., González, D., Alfaro, I., Chinesta, F., and Cueto, E., "An Augmented Reality Platform for Interactive Aerodynamic Design and Analysis," Int. J. Numer. Methods Eng.

[38] Roweis, S. T., and Saul, L. K., 2000, "Nonlinear Dimensionality Reduction by Locally Linear Embedding," Science (80-.)., **290**(5500), pp. 2323–2326.

[39] Kingma, D. P., and Ba, J., 2014, "Adam: A Method for Stochastic Optimization," arXiv Prepr. arXiv1412.6980.