A Predictive and Generative Design Approach for 3D Mesh Shapes Using Target-Embedding Variational Autoencoder

Xingang Li
Walker Department of Mechanical Engineering
University of Texas at Austin
Austin, TX 78712, USA
xingang.li@utexas.edu
ASME Student Member

Charles Xie
Institute for Future Intelligence
Natick, MA 01760, USA
charles@intofuture.org

Zhenghui Sha[1]
Walker Department of Mechanical Engineering
University of Texas at Austin
Austin, TX 78712, USA
zsha@austin.utexas.edu
ASME Member

ABSTRACT

In this paper, we present a predictive and generative design approach to supporting the conceptual design of product shapes in 3D meshes. We develop a target-embedding variational autoencoder (TEVAE) neural network architecture, which consists of two modules: 1) a training module with two encoders and one decoder (E^2D network); and 2) an application module performing the generative design of new 3D shapes and the prediction of a 3D shape from its silhouette. We demonstrate the utility and effectiveness of the proposed approach in the design of 3D car body and mugs. The results show that our approach can generate a large number of novel 3D shapes and successfully predict a 3D shape based on a single

[1] Corresponding author: zsha@austin.utexas.edu

silhouette sketch. The resulting 3D shapes are watertight polygon meshes with high-quality surface details, which have better visualization than voxels and point clouds, and are ready for downstream engineering evaluation (e.g., drag coefficient) and prototyping (e.g., 3D printing).

1. INTRODUCTION

Sketching plays an essential role in sparking creative ideas to explore emerging design concepts [1]. For example, in car design, characteristic contour lines are often used to represent silhouettes in supporting conceptual design of car body shapes [2, 3], which complements many other ideation approaches, such as freehand sketches, design analogies, and prototypes. Compared to freehand sketches, silhouettes regularize the sketching process, and thereby makes sketching easier and more manageable. This is particularly useful for designers who lack professional sketching skills. However, silhouettes, as a specific type of 2D sketches, are often ambiguous and lack of geometric details. In later design stages, such as embodiment design, a 3D computer-aided design (CAD) model is often required to more accurately evaluate the engineering performance of a design concept. 3D shapes can also provide better visualization and thus help designers better understand the design, inspiring them to develop new shapes and refine geometric details. Therefore, the question is: can we build a system to predict and automatically generate 3D shapes just based on silhouettes?

Such a system will yield several benefits. First, it automates the 2D-to-3D reconstruction process, thereby saving labor and time. Designers can allocate more resources for better design iteration and ideation. Second, all silhouettes created during the conceptual design stage can be evaluated against the desired engineering performance in 3D form. So, designs that would have better performance will not be ruled out too

early when performance-driven decisions (i.e., rational decisions) are not yet obtained. Third, ordinary people would not be discouraged to show their design ideas merely due to the lack of CAD experience or sketching skills. This may have significant educational implications for training novice designers and facilitate the democratization of design innovation. Lastly, enterprises may use this system to enable user interface soliciting consumer preferences for design customization.

However, automatically reconstructing 3D shapes directly from 2D sketches is a challenge because it is an ill-defined problem due to insufficient and imperfect information from simple strokes [4]. To tackle this challenge, inspired by the target-embedding autoencoder (TEA) network [5, 6], we propose a novel target-embedding variational autoencoder (TEVAE) (see Fig. 1(a)). The TEVAE architecture consists of two modules: 1) A training module with an E^2D network that has two encoders and one decoder. 2) An application module performing two functions: *generative design* function, such as shape interpolation and random generation of new 3D shapes; and *predictive design* function (i.e., 3D shape prediction from silhouette sketches). The integration of generative and predictive functions is beneficial in that it makes the structure of the neural network compact, thus saving training costs. To demonstrate the utility and generalizability of the proposed approach, we apply it to two case studies in the design of 3D car body and mugs.

The contributions of this paper are summarized below.

a) To the best of our knowledge, this is the first attempt of developing a system integrating both predictive and generative functions of 3D mesh shapes from silhouettes. The TEA has a classic autoencoder that can perform pseudo

generative tasks, but essentially, is not a generative model. Our TEVAE applies variational autoencoders (VAEs) and becomes a true generative model. It can also learn a continuous and smooth latent representation of the data.

b) Predicting 3D shapes from silhouettes is more challenging because a silhouette sketch provides less information (e.g., depth and normal maps) than traditional freehand sketches with inner contour lines. To that end, we introduce an intermediate step (e.g., extrusion or rotation) to first convert the silhouette to a 3D primitive shape. This transforms the original 2D-to-3D problem to a 3D-to-3D problem, which promotes a stable training process and the generation of reliable and viable 3D shapes.

c) Building upon a graph convolutional mesh VAE [7], our approach can directly output high-quality 3D mesh shapes that are more storage-efficient for high-resolution 3D structure, compared to point clouds [8] and voxels [9]. 3D meshes also facilitates engineering analyses because they are compatible with existing computer-aided engineering (CAE) software[2].

d) A data automation program is developed for training data pairs of 3D shapes that can be used in any TEA-like neural networks for supervised learning problems.

2. LITERATURE REVIEW

In this section, we review the existing research that is most relevant to our work.

2.1 Learning-Based Sketch-to-3D Generation Methods

[2] "Meshes" are used for 3D representation here. In CAE software, there is a concept called "meshing". Meshing is a process that breaks down the continuous geometric space of an object into a discrete number of shape elements. All 3D representations including meshes and native CAD data format (e.g., IGES, DWG, and STL) that can be directly input to CAE software have to go through the meshing process for analysis.

Point clouds and voxels have been widely used as 3D representations for sketch-to-3D generation [8-10]. These methods need to postprocess the resulting 3D shapes to meshes for better visualization, which still suffer from low surface quality. There are studies attempting to directly produce high-quality mesh shapes. For example, concurrent with the development of our approach, Guillard et al. [11] propose a pipeline to reconstruct and edit 3D shapes from 2D sketches. They train an encoder/decoder architecture to regress surface meshes from freehand sketches. The method applies a differentiable rendering technique to iteratively refine the resulting 3D shapes. Similarly, Xiang et al. [12] integrate a differentiable rendering approach to an end-to-end learning framework for predicting 3D mesh shapes from line drawings.

All methods above are promising and have inspired us to explore a more challenging task, i.e., to predict a 3D shape from a simple silhouette sketch. Our approach is similar to [10, 11, 13, 14] in that we only need one single sketch as input, but we create a new neural network architecture that can predict a 3D shape from a single silhouette and simultaneously generate novel 3D shapes. The direct output shapes are 3D meshes with high-quality surface details, thus, requiring no postprocessing.

2.2 Learning-Based Generative Design Methods

Learning-based generative design (GD) methods have been primarily developed based on two techniques, generative adversarial networks (GANs) [15] and variational autoencoders (VAEs) [16]. There are several approaches for 2D designs [17-19], but they are not appropriate for design applications that require 3D models. In 3D applications, Shu et al. [20] present a method that combines GAN and the physics-based virtual environment introduced in [18] to generate high-performance 3D aircraft models. Zhang

et al. [21] propose a method using a VAE, a physics-based simulator, and a functional design optimizer to synthesize 3D aircrafts with prescribed engineering performance. Building upon [17], Yoo et al. [22] develop a deep learning-based CAD/CAE framework that can automatically generate 3D car wheels from 2D images. Gunpinar et al. [3] apply a spatial simulated annealing algorithm to generate various silhouettes of cars, which are then extruded to 3D car models. Those models can be further refined by sweeping a predefined cross-section sketch. However, simply extrusion does not guarantee satisfactory outcomes, and the resulting 3D car models look unreal.

2.3 Target-Embedding Representation Learning

Girdhar et al. propose a TL-embedding network [6] that is composed of a T-network for training and an L-network for testing. The T-network contains an autoencoder (encoder-decoder) network and a CNN. After training, the L-network can be used to predict 3D shapes in voxel from images. Similarly, Mostajabi et al. [23] uses an autoencoder and a CNN to perform the semantic segmentation task of images. Dalca et al. [24] apply a similar network structure as [6, 23], consisting of a prior generative model to generate paired data (biomedical images and anatomical regions) to solve the scarcity of labeled image data for anatomical segmentation tasks. Jarrett and Schaar [5] categorize these studies as supervised representation learning methods. They observe that when the dimension of the target data space is higher or similar to the feature data space, a target-embedding autoencoder (TEA) can be more effective than a feature-embedding autoencoder (FEA). The authors verify that the TEA structure will guarantee the learning stability by using a mathematical proof of a simple linear TEA and showing the empirical

145 results from a complex non-linear TEA. Inspired by those existing works, we construct
146 the target-embedding variational autoencoder (TEVAE) architecture.

147 **3. APPROACH**

148 The proposed target-embedding variational autoencoder (TEVAE) architecture,
149 shown in Fig. 1(a), consists of two modules: a training module and an application
150 module.

151 **3.1 The E^2D Network and the Two-Stage Training**

152 The key component of the training module is the E^2D network that consists of
153 two encoders and one decoder, which is constructed by concatenating an encoder (labeled
154 as $Enc_2(\cdot)$) to a mesh VAE (an encoder-decoder network, labeled as $Enc_1(\cdot)$ and $Dec(\cdot)$)
155 [7]. The $Enc_1(\cdot)$ maps target shapes (S_t, i.e., the original authentic 3D mesh shapes) to a
156 low-dimensional latent space, and the $Dec(\cdot)$ maps latent vectors from that latent space
157 to 3D mesh shapes. With the same network structure as $Enc_1(\cdot)$, $Enc_2(\cdot)$ takes source
158 shapes (S_s, the 3D mesh shapes extruded from silhouette sketches of the target shapes) as
159 the input and maps them to the same dimensional latent space as the mesh VAE.

160 We adopt the same loss function developed in [7] to train $Enc_1(\cdot)$ and $Dec(\cdot)$.
161 For $Enc_2(\cdot)$, we create a new loss function L_2 as below.

$$L_2 = \alpha D_{Regress} + D_R \tag{1}$$

162 where α is the weight for the regression loss and

$$D_{Regress} = \frac{1}{2M}\sum_{i=1}^{M} ||{\mu_2}^i - {\mu_1}^i||_2^2 \tag{2}$$

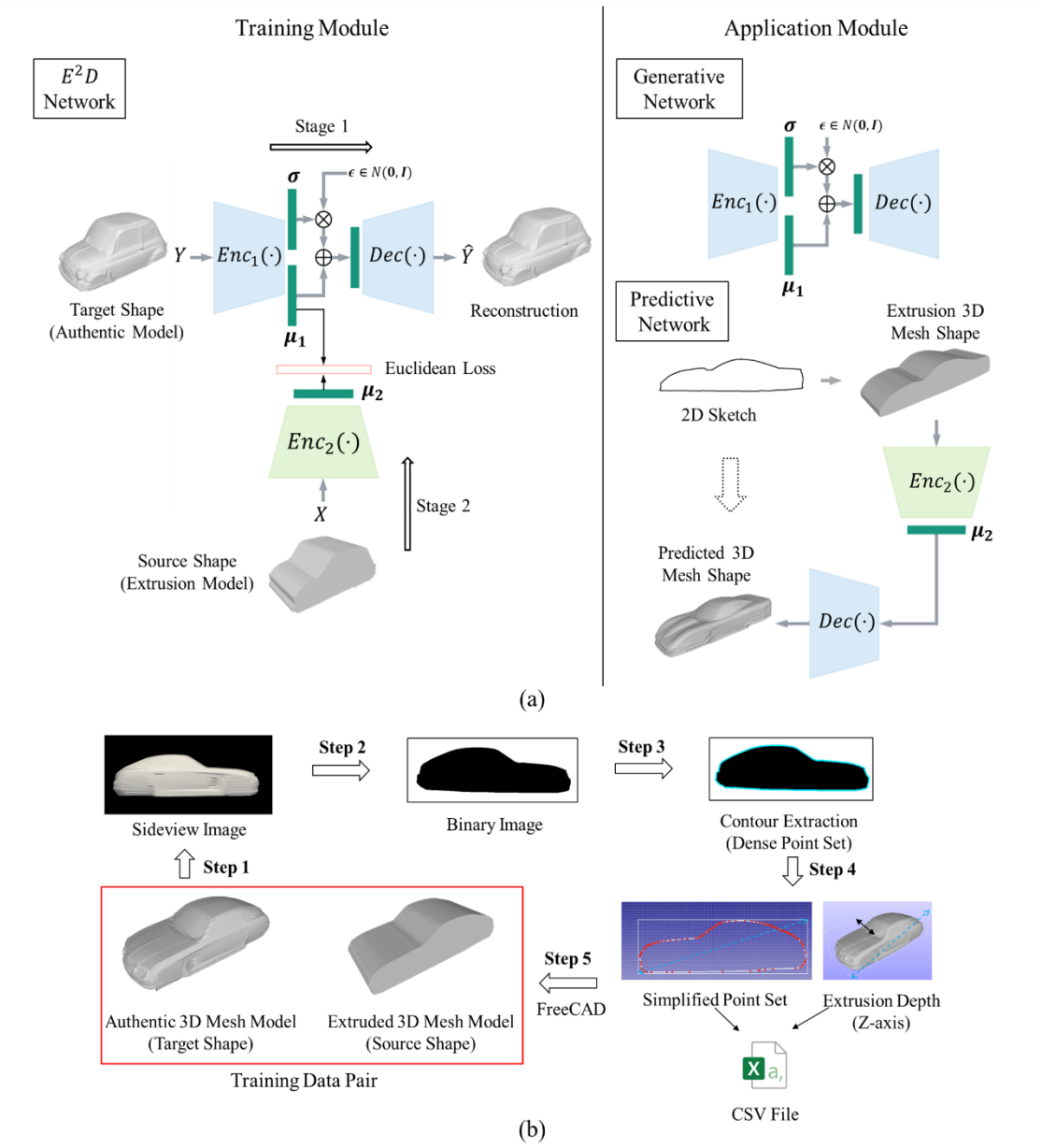


Figure 1: (a) The proposed approach using target-embedding variational autoencoder (TEVAE); (b) The preparation of data pairs

163

164 denotes the Euclidean loss, where $\mu_2{}^i$ is the output of $Enc_2(\cdot)$ using X^i as input and $\mu_1{}^i$ is

165 the mean vector obtained from the latent space of the mesh VAE using the input of Y^i.

166 D_R is the regularization loss applied to improve the generalization ability of the $Enc_2(\cdot)$.

We apply a two-stage training strategy [23] to jointly train the mesh VAE and $Enc_2(\cdot)$ from scratch. In Stage 1, the mesh VAE is trained independently. In Stage 2, we fix all the learning parameters of the mesh VAE and train $Enc_2(\cdot)$ by minimizing L_2.

3.2 The Predictive Network and Generative Network

After the E^2D network is trained, we connect $Enc_2(\cdot)$ to $Dec(\cdot)$ to form the predictive network. It can take a 3D extrusion mesh shape as input and output a 3D mesh shape that is similar to the input shape but has finer geometric details, making it authentic and aesthetic. We use the trained mesh VAE as the generative network, which can perform generative design tasks, including shape reconstruction, interpolation, and random generation.

3.3 Preparation of Data Pairs

Data pairs $\{S_s^i, S_t^i\}_{i=1}^N$ are needed to train the E^2D network. Fig. 1(b) shows the process of obtaining one training data pair using a car body as an example. From the sideview image of an authentic 3D car model, we extract its contour points, from which we can obtain an extrusion model using the FreeCAD Python API. We develop a set of Python scripts that fully automate the whole process, which are made open-source for the community[3]. We process $N = 1240$ car models obtained from [25] and $N = 203$ mug models from [26]. For car models, we keep only car bodies by removing all the other parts, such as mirrors, wheels, and spoilers.

3.4 Shape Preprocessing and Feature Representation

The E^2D network requires input mesh shapes in both the source shape set ($\{S_s^i\}_{i=1}^N$) and the target shape set ($\{S_t^i\}_{i=1}^N$) to have the same topology (i.e., the same

[3] https://github.com/Xingang1990/TEVAE

number of vertices and the same mesh connectivity). However, the mesh typologies between the two datasets can be different. For simplicity, we use a uniform topology for both datasets and the non-rigid registration method [27] is applied to meet this requirement. Nonrigid registration is a widely used technique in the computer graphics field to map one point set (e.g., point cloud, mesh) to another. A uniform unit cube mesh with 19.2k triangles (9602 vertices) is used to register all mesh shapes. This makes all shapes have the same topology as the cube mesh, but remain the same as their original shapes and geometric details.

The As-Consistent-As-Possible (ACAP) method [28] is applied to extract features of a 3D shape to input to the E^2D network. We deform the aforementioned uniform cube mesh to a target 3D mesh shape by multiplying deformation matrices, from which nine unique numbers can be extracted for each vertex of the mesh shape. Thus, a shape with v vertices can be represented by a feature matrix $M_f \in \mathbb{R}^{v\times 9}$, where $v = 9602$ in our implementation. We can get the feature representations of the source shape dataset $X = \{X^k\}_{k=1}^N$ and the target shape dataset $Y = \{Y^l\}_{l=1}^N$, where $N = 1240$ for the car models and $N = 203$ for the mug models, and $\{X^i, Y^i\}$ forms the input feature of one data pair.

More details of the approach and the training of the E^2D network are provided in the [Supplementary Material].

4. CASE STUDIES AND RESULTS

4.1 Implementation of the Two-Stage Training

For training the mesh VAE in Stage 1, the input target shape dataset $Y = \{Y^l\}_{l=1}^N$ is randomly divided into the training set (80%) and the test set (20%). For training the $Enc_2(\cdot)$ network in Stage 2, we also do an 80-20 split of the source shape dataset $X =$

212 $\{X^k\}_{k=1}^N$, and meanwhile use the data pair to make sure the i^{th} target shape (S_t^i)

213 corresponds to the i^{th} source shape (S_s^i) in both the training and test sets.

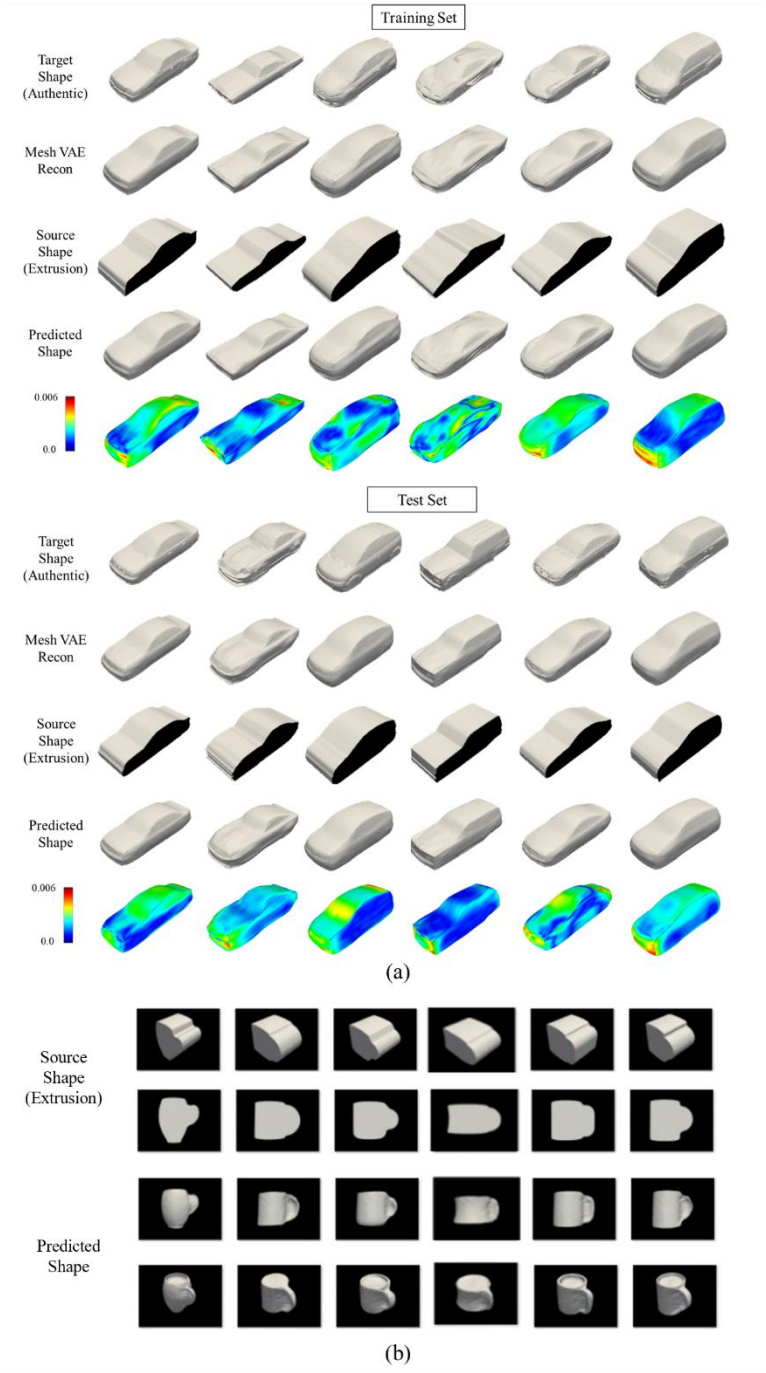


Figure 2: (a) Results of car bodies: predicted shapes (in the fourth row) and reconstructed shapes (in the second row); (b) The prediction results of mugs

214

4.2 The Predictive Network

The predictive network aims to predict a 3D shape from an input silhouette sketch. We conducted experiments on the prediction of the training set and the test set. The results of the car models are shown in Fig. 2(a). For the result of the training set, the first row shows the target shapes, and the following rows are their corresponding reconstruction shapes from the mesh VAE, extrusion shapes (with silhouettes marked in dark), and the predicted shapes, respectively. The results indicate that, given an input extrusion shape from the corresponding silhouette sketch, the predictive network is capable of predicting an authentic 3D shape, as illustrated in the fourth row. It should be noted that even though we are targeting shapes (ground truth) in the first row, the best results that can be achieved from the predictive network are the reconstruction shapes in the second row. The reconstruction shapes and the corresponding predicted shapes look identical in terms of visual appearance, but are different in geometric details. To show the difference, we compute the Hausdorff distance between those shapes and visualize the distance values in the fifth row. Similar results are also observed for the shapes in the test set, which indicates a good generalization of the network because the test set shapes are unseen data for the network. This is particularly important in real-world applications, where user input often does not resemble existing shapes in a training dataset.

The prediction results of the mug models are shown in Fig. 2(b). The first two rows are the source shapes that are obtained from extruding the silhouettes, while the last two rows are the corresponding predicted shapes. Mugs are generally non-extrudable from side-view silhouettes, so the extrusion shapes look more like toast instead of mugs. However, our approach can still predict authentic mug shapes. Please note that, besides

238 extruding, other 3D modeling techniques, such as revolving and sweeping, can also be
239 used to obtain 3D primitive shapes. Extruding is adopted in this study for ease of
240 implementation. In addition, it provides us with basic geometric features for 3D shape
241 prediction.

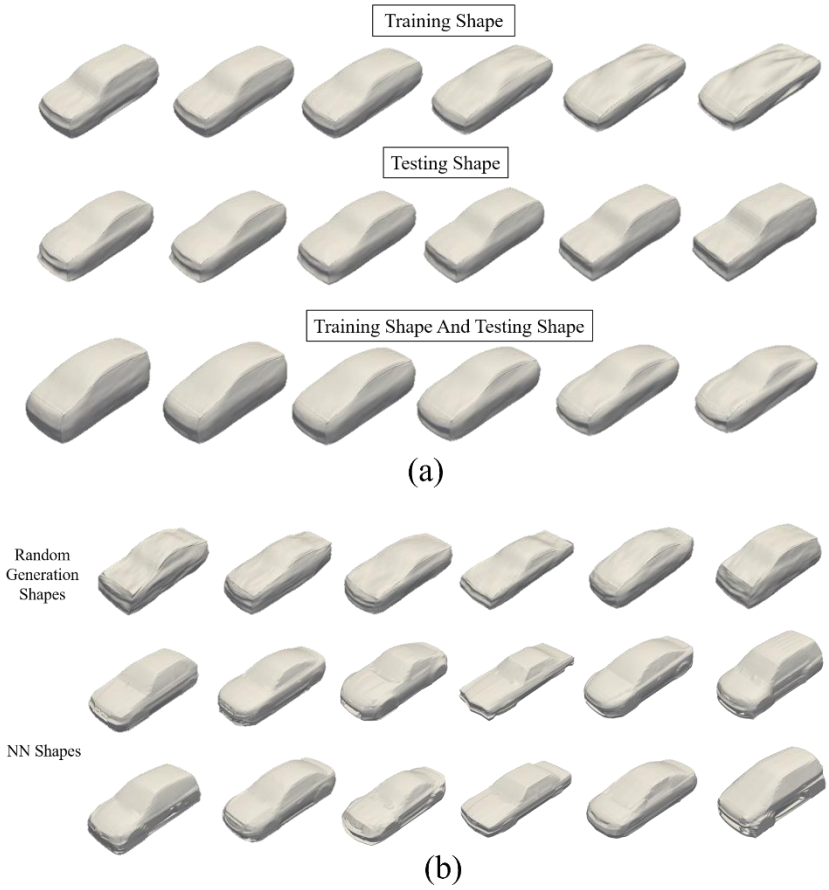


Figure 3: (a) The results of shape interpolation for three cases; (b) The results of random generation shapes along with two nearest neighbor (NN) shapes

242

243 **4.3 The Generative Network**

244 For the generative network, different generative operations, such as shape
245 reconstruction, interpolation, and random generation, can be performed. The
246 reconstructed 3D shapes are already shown in the second row for both the training set and
247 the testing set in Fig. 2(a). For shape interpolation, new 3D shapes are synthesized by

linearly interpolating two target 3D shapes through their encoded latent vectors. We demonstrate the results of shape interpolation in three cases using the case study of car models (see Fig. 3(a)): 1) interpolation between two training shapes, 2) between two test shapes, and 3) between a training shape and a test shape. In each case, the first and the last columns are the shapes to be interpolated, and the in-between columns are linearly interpolated shapes. It can be observed that there is a gradual transition of the shape geometry between the two target shapes.

For random shape generation, latent vectors are randomly sampled from the latent space of the Mesh VAE, and decoded by the trained $Dec(\cdot)$ to 3D mesh shapes. Fig. 3(b) shows that the generative network can generate novel car models (in the first row) that are not seen in the original dataset. This is validated by finding their nearest neighbors (NNs) (the second and third rows) in the original dataset based on the Hausdorff distance. A quick visual comparison between the randomly generated car models and their NNs tells the differences, and they are indeed new shapes.

5. CONCLUSION

To tackle the challenge of predicting a 3D shape from a silhouette sketch, we present a novel target-embedding variational autoencoder (TEVAE) network that enables a 2D-to-3D design approach. Our approach can effectively predict a 3D shape from a silhouette sketch. The predicted 3D shape is consistent with the input sketch, and is authentic with rich geometric details. Such a design transformation could greatly shorten the iteration between the design ideation to CAD modeling. The approach can also generate novel 3D shapes, and thus could better inspire designers for their creative work.

The resulting 3D shapes are represented in meshes, which are ready for downstream engineering analyses, evaluation, and prototype (e.g., 3D printing).

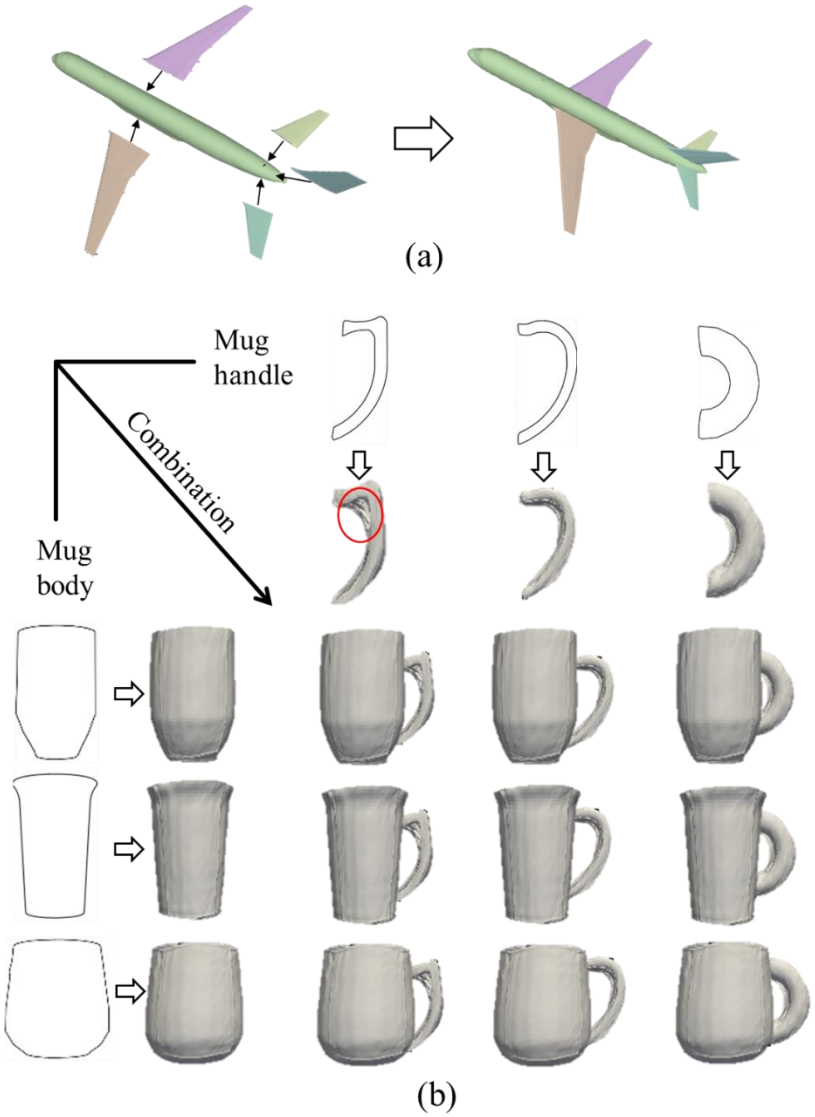


Figure 4: Complex geometries (e.g., a toy plane) or non-genus-zero shapes (e.g., a mug) can be partitioned to several genus-zero shapes. Then, the proposed approach can be applied to each component for shape exploration and synthesis

Quantity yields quality, and this can be achieved by broadening the initial pool of concept ideas [29]. We believe that the presented approach can help designers explore the design space more efficiently and stimulate creative design ideas in early design stages.

From the methodology point of view, this new generative design approach is general enough to be applied in many applications where 3D shape modeling and rendering are necessary. As long as the sketch can provide a major perspective view of an object, like the frontview of a human body and the sideview of a bottle, the corresponding authentic 3D shape can be predicted and novel shape concepts can be generated. In addition, our approach is friendly to ordinary people who have few professional sketching skills, since it only requires a simple silhouette of an object as input.

There are a few limitations in the current study that the authors would like to share. First, the current model only handles genus-zero shapes and ignores any through holes (e.g., the hole between the body and the handle of a mug in Fig. 2(b)) in the original shape due to the non-rigid registration [27]. However, many design artifacts are usually non-genus-zero (e.g., a mug with a through hole between the body and the handle) or have more complex geometry consisting of many components, e.g., a plane model can have a body, two wings, and three tails, etc., as shown in Fig. 4(a).

To address this limitation, a part-aware method [26, 30] may be a potential solution. We perform a quick experiment using a part-aware mug design problem (see Fig. 4(b)). In this particular application, users can first draw an outline sketch for an individual component (e.g., a mug body or a handle). Then, the corresponding 3D mesh shape can be predicted and new shapes can be generated. Lastly, the resulting individual components can be combined to a holistic structure allowing non-genus-zero topology. However, the part-aware strategy could not work for parts that are non-genus-zero and unable to be further decomposed into genus-zero components, e.g., a chair back with hollow-out structures and holes. In this experiment, we applied a cube mesh template to

299 register mug models using non-rigid registration [27] as introduced previously. However,
300 we observed that artifacts with a large curvature could not be perfectly registered (e.g.,
301 the mug handle in Fig. 4(b) highlighted by a circle). This issue can be alleviated by using
302 different templates of 3D primitives, e.g., a sphere or a cylinder.

303 In addition to the part-aware method, other methods based on new 3D
304 representations could also be applied to address the first limitation. For example,
305 primitive-based methods can use a set of primitive surfaces to represent a 3D shape [31-
306 34]. Implicit 3D representation (e.g., signed distance fields [35, 36]) can characterize 3D
307 surfaces implicitly, and the resulting 3D geometries can be converted to mesh
308 representation. These methods can capture the topology changes of 3D shapes without
309 using a template mesh for data registration, thus deserving our future exploration.

310 Second, the constructed 3D shapes are consistent with the designer's sketch in
311 terms of sideview, but they might not be the same as what the designer has in mind. To
312 address this limitation, we plan to integrate interactive modeling techniques to build a
313 graphical user interface (GUI) for users to adjust the generated 3D shapes according to
314 their preferences.

315 Third, the generative network performs well in shape reconstruction and shape
316 interpolation, but the success rate of random shape generation is lower than one-third due
317 to the sparsity of the training data. Therefore, the random shape generation function is not
318 fully reliable in practice for now. This problem could be solved by obtaining more 3D
319 shape data using data augmentation methods, such as the one presented in the study of
320 [13], to improve the diversity and quality of the training dataset. These limitations
321 motivate us to further improve the current model in the future.

ACKNOWLEDGMENT

The authors appreciate the financial support from the NSF through the grant DUE-1918847. We also thank Dr. Miaoqing Huang for granting us access to the computer with GPUs to train the E^2D network.

FUNDING

This study is supported by the U.S. National Science Foundation (NSF) DUE through grant #1918847.

REFERENCES

[1] Pratt, M. J., Anderson, B. D., and Ranger, T., 2005. “Towards the standardized exchange of parameterized feature-based cad models”. *Computer-Aided Design*, 37(12), pp. 1251–1265.

[2] Reid, T. N., Gonzalez, R. D., and Papalambros, P. Y., 2010. “Quantification of perceived environmental friendliness for vehicle silhouette design”. *Journal of Mechanical Design*, 132(10), 101010.

[3] Gunpinar, E., Ovur, S. E., and Gunpinar, S., 2019. “A user-centered side silhouette generation system for sedan cars based on shape templates”. *Optimization and Engineering*, 20(3), pp. 683–723.

[4] Schmidt, R., Khan, A., Kurtenbach, G., and Singh, K., 2009. “On expert performance in 3d curve-drawing tasks”. *In Proceedings of the 6th euro graphics symposium on sketch-based interfaces and modeling*, pp. 133–140.

[5] Jarrett, D., and van der Schaar, M., 2020. “Target- embedding autoencoders for supervised representation learning”. *arXiv preprint* arXiv:2001.08345.

[6] Girdhar, R., Fouhey, D. F., Rodriguez, M., and Gupta, A., 2016. “Learning a predictable and generative vector representation for objects”. *In European Conference on Computer Vision*, Springer, pp. 484–499.

[7] Yuan, Y.-J., Lai, Y.-K., Yang, J., Duan, Q., Fu, H., and Gao, L., 2020. “Mesh variational autoencoders with edge contraction pooling”. *In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pp. 274–275.

[8] Lun, Z., Gadelha, M., Kalogerakis, E., Maji, S., and Wang, R., 2017. “3d shape reconstruction from sketches via multi-view convolutional networks”. *In 2017 International Conference on 3D Vision (3DV), IEEE*, pp. 67–77.

[9] Jin, A., Fu, Q., and Deng, Z., 2020. “Contour-based 3d modeling through joint embedding of shapes and contours”. *In Symposium on Interactive 3D Graphics and Games*, pp. 1–10.

[10] Nozawa, N., Shum, H. P., Feng, Q., Ho, E. S., and Morishima, S., 2021. “3d car shape reconstruction from a contour sketch using gan and lazy learning”. *The Visual Computer*, pp. 1–14.

[11] Guillard, B., Remelli, E., Yvernay, P., and Fua, P., 2021. “Sketch2mesh: Reconstructing and editing 3d shapes from sketches”. *arXiv preprint* arXiv:2104.00482.

[12] Xiang, N., Wang, R., Jiang, T., Wang, L., Li, Y., Yang, X., and Zhang, J., 2020. “Sketch-based modeling with a differentiable renderer”. *Computer Animation and Virtual Worlds*, 31(4-5), p. e1939.

[13] Nozawa, N., Shum, H. P., Ho, E. S., and Morishima, S., 2020. “Single sketch image based 3d car shape reconstruction with deep learning and lazy learning.”. *In VISIGRAPP (1: GRAPP)*, pp. 179–190.

[14] Han, X., Gao, C., and Yu, Y., 2017. “Deepsketch2face: A deep learning based sketching system for 3d face and caricature modeling”. *ACM Transactions on graphics (TOG)*, 36(4), pp. 1–12.

[15] Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y., 2014. “Generative adversarial nets”. *In Advances in neural information processing systems*, pp. 2672–2680.

[16] Kingma, D. P., and Welling, M., 2013. "Auto-encoding variational bayes". *arXiv preprint* arXiv:1312.6114.
[17] Oh, S., Jung, Y., Kim, S., Lee, I., and Kang, N., 2019. "Deep generative design: Integration of topology optimization and generative models". *Journal of Mechanical Design*, 141(11).
[18] Dering, M., Cunningham, J., Desai, R., Yukish, M. A., Simpson, T. W., and Tucker, C. S., 2018. "A physics-based virtual environment for enhancing the quality of deep generative designs". *In ASME 2018 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, American Society of Mechanical Engineers Digital Collection.
[19] Fujita, K., Minowa, K., Nomaguchi, Y., Yamasaki, S., and Yaji, K., 2021. "Design concept generation with variational deep embedding over comprehensive optimization". *In International Design Engineering Technical Conferences and Computers and Information in Engineering Conference,* Vol. 85390, American Society of Mechanical Engineers, p. V03BT03A038.
[20] Shu, D., Cunningham, J., Stump, G., Miller, S. W., Yukish, M. A., Simpson, T. W., and Tucker, C. S., 2020. "3d design using generative adversarial networks and physics-based validation". *Journal of Mechanical Design*, 142(7).
[21] Zhang, W., Yang, Z., Jiang, H., Nigam, S., Yamakawa, S., Furuhata, T., Shimada, K., and Kara, L. B., 2019. "3d shape synthesis for conceptual design and optimization using variational autoencoders". *In International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, Vol. 59186, American Society of Mechanical Engineers, p. V02AT03A017.
[22] Yoo, S., Lee, S., Kim, S., Hwang, K. H., Park, J. H., and Kang, N., 2020. "Integrating deep learning into cad/cae system: Case study on road wheel design automation". *arXiv preprint* arXiv:2006.02138.
[23] Mostajabi, M., Maire, M., and Shakhnarovich, G., 2018. "Regularizing deep networks by modeling and predicting label structure". *In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5629–5638.
[24] Dalca, A. V., Guttag, J., and Sabuncu, M. R., 2018. "Anatomical priors in convolutional networks for unsupervised biomedical segmentation". *In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 9290–9299.
[25] Umetani, N., 2017. "Exploring generative 3d shapes using autoencoder networks". *In SIGGRAPH Asia 2017 technical briefs*. pp. 1–4.
[26] 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 Transactions on Graphics (TOG),* 38(6), pp. 1–15.
[27] 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 Transactions on Graphics (ToG),* 33(4), pp. 1–12.
[28] Gao, L., Lai, Y.-K., Yang, J., Ling-Xiao, Z., Xia, S., and Kobbelt, L., 2019. "Sparse data driven mesh deformation". *IEEE transactions on visualization and computer graphics.*
[29] Yang, M. C., 2003. "Concept generation and sketching: Correlations with design outcome". *In International Design Engineering Technical Conferences and Computers and Information in Engineering Conference,* Vol. 37017, pp. 829–834.

[30] Li, X., Xie, C., and Sha, Z., 2021. “Part-aware product design agent using deep generative network and local linear embedding”. *In Proceedings of the 54th Hawaii International Conference on System Sciences*, p. 5250.
[31] Paschalidou, D., Katharopoulos, A., Geiger, A., and Fidler, S., 2021. “Neural parts: Learning expressive 3d shape abstractions with invertible neural networks”. *In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 3204–3215.
[32] Vasu, S., Talabot, N., Lukoianov, A., Baque, P., Donier, J., and Fua, P., 2021. “Hybridsdf: Combining free form shapes and geometric primitives for effective shape manipulation”. *arXiv preprint* arXiv:2109.10767.
[33] Jones, R. K., Hanocka, R., and Ritchie, D., 2021. “The neurally-guided shape parser: A monte carlo method for hierarchical labeling of over-segmented 3d shapes”. *arXiv preprint* arXiv:2106.12026.
[34] Groueix, T., Fisher, M., Kim, V. G., Russell, B. C., and Aubry, M., 2018. “A papier-mâché approach to learning 3d surface generation”. *In Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 216–224.
[35] Chen, Z., and Zhang, H., 2019. “Learning implicit fields for generative shape modeling”. *In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 5939–5948.
[36] Park, J. J., Florence, P., Straub, J., Newcombe, R., and Lovegrove, S., 2019. “Deepsdf: Learning continuous signed distance functions for shape representation”. *In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 165–174.

443 **Figure Captions List**

444

Fig. 1 (a) The proposed approach using target-embedding variational autoencoder (TEVAE); (b) The preparation of data pairs

Fig. 2 (a) Results of car bodies: predicted shapes (in the fourth row) and reconstructed shapes (in the second row); (b) The prediction results of mugs

Fig. 3 (a) The results of shape interpolation for three cases; (b) The results of random generation shapes along with two nearest neighbor (NN) shapes

Fig. 4 Complex geometries (e.g., a toy plane) or non-genus-zero shapes (e.g., a mug) can be partitioned to several genus-zero shapes. Then, the proposed approach can be applied to each component for shape exploration and synthesis

445

Table Captions List

SUPPLEMENTARY MATERIALS

Part 1: The E^2D Network

We construct the E^2D network by concatenating an encoder ($Enc_2(\cdot)$) to a mesh VAE [1]. The $Enc_1(\cdot)$ has the following network structure: two graph convolutional (Conv) layers with a batch normalization (BN) layer and a *tanh* activation function following each graph Conv layer, one pooling layer, and a third graph Conv layer. The output of the last Conv layer is mapped to a 128-dimensional space that contains a mean vector ($\mu_1 \in \mathbb{R}^{128}$) and a deviation vector ($\sigma \in \mathbb{R}^{128}$) by two different fully-connected (FC) layers. The mean vector does not have an activation function and the deviation vector uses *sigmoid* as the activation function. The $Enc_2(\cdot)$ shares the same network structure as $Enc_1(\cdot)$ except that only one FC layer is used to form the latent vector ($\mu_2 \in \mathbb{R}^{128}$) because the standard deviation vector is not needed here. $Enc_1(\cdot)$ takes Y and $Enc_2(\cdot)$ takes X as input, where $X \in \mathbb{R}^{V\times 9}$ and $Y \in \mathbb{R}^{V\times 9}$ are feature representations of source shapes (S_s) and target shapes (S_t), respectively. V represents the number of vertices of a 3D mesh shape. S_s are 3D extrusion mesh models extruded from sideview sketches of the original authentic 3D mesh models (S_t).

The decoder ($Dec(\cdot)$) mirrors the encoders and it consists of an FC layer, a graph Conv layer and a de-pooling layer, followed by two graph Conv layers. Each Conv layer is connected to a BN layer and a *tanh* activation function except the third Conv layer. The $Dec(\cdot)$ takes the latent vector $z_1 = \mu_1 + \sigma\epsilon$ as input, where $\epsilon \in N(0, I)$ which is a standard multivariate Gaussian distribution. The output of $Dec(\cdot)$ is $\hat{Y} \in \mathbb{R}^{V\times 9}$ which has the same dimension as the input Y and can be used to reconstruct 3D mesh shapes.

The entire E^2D network is trained by minimizing the following loss function:

$$L_{total} = \lambda L_1 + L_2 \tag{1}$$

where L_2 is the loss function for $Enc_2(\cdot)$ which has been shown in the paper, λ is a weight parameter, and L_1 is the loss function for the mesh VAE. L_1 can be written as:

$$L_1 = \alpha_1 L_{Recon} + \alpha_2 L_{KL} + L_{Reg} \tag{2}$$

where α_1 and α_2 are the weights of different loss terms, and

$$L_{Recon} = \frac{1}{2M} \sum_{i=1}^{M} ||Y^i - \hat{Y}^i||_F^2 \tag{3}$$

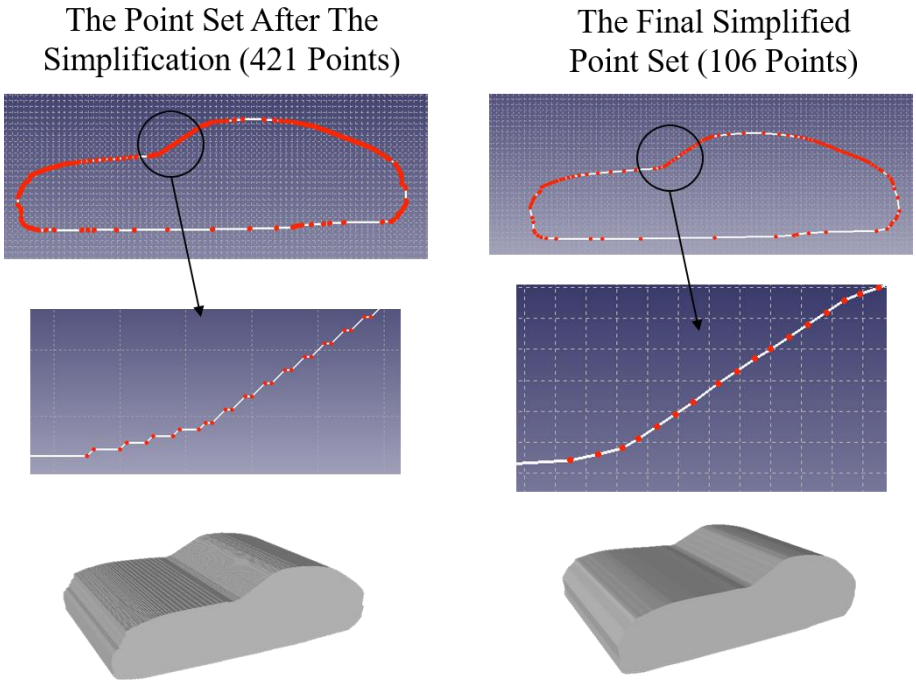
denotes the mean squared error (MSE) reconstruction loss, where Y^i and $\hat{Y}^i$ represent the input feature matrix of the i^{th} model and the corresponding output of the mesh VAE, respectively. $||\cdot||_F$ is the Frobenius norm of the matrix and M is the number of mesh shapes in the training dataset.

$$L_{KL} = D_{KL}\big(q(z_1||Y)||p(z_1)\big) \tag{4}$$

represents the Kullback–Leibler (KL) divergence loss to promote the Gaussian distribution in the latent space, where z_1 is the latent vector, $p(z_1)$ is the prior probability, $q(z_1||Y)$ is the posterior distribution given the feature matrix Y, and D_{KL} is the KL-divergence. L_{Reg} is the squared l_2 norm regularization loss of the parameters which is used to avoid overfitting to the training data to improve the generalization ability of the mesh VAE.

For the two-stage training in our application, in Stage 1, the mesh VAE is trained independently. The network is initialized at random and trained end-to-end by minimizing L_1 as shown in Supp. Equation (2). The 128-dimensional mean vectors μ_1 from the latent space of the trained mesh VAE, will be used in Stage 2 training. In Stage

2, we fix all the learning parameters of the trained mesh VAE and train $Enc_2(\cdot)$. $Enc_2(\cdot)$ is also initialized randomly. It is trained by minimizing L_2 as shown in Equations (1) and (2) of the paper.



Supplemental Figure 1: An example showing the comparison between resulting extrusion mesh models using a point set after the approximation method and the final simplified point set

Part 2: Data Pair Preparation

We present more details of the process of how we prepare training data pairs (see Fig 1(b) of the paper) using a car body as the example. Data can be processed as the following steps.

1) Obtain the sideview image from an authentic 3D car mesh shape.
2) Get the binary image from the sideview image.
3) Extract the contour of the binary image, which results in a dense point set.

4) Get a simplified point set using a simple contour approximation method provided in OpenCV-Python package.
5) Get an extrusion shape using the FreeCAD Python API. Then, the original authentic 3D mesh shape (target shape) and the extrusion shape (source shape) form one data pair.

An approximation method is used in Step 4 because the extrusion model directly from the original contour points (a dense point set) has zigzags on its surface, which affects the quality of the extruded shapes. The contour approximation method can keep more points where the geometry is more complex (e.g., bumper lines, windshield lines) and fewer points where the geometry is simpler (e.g., the bottom line). In our implementation, as shown in Supp. Fig. 1, this simplification process results in a point set that has around 400 points (varies a little for different car models), and we reduce the number of points by a factor of 1/4 to around 100 points for a further avoidance of zigzags on the models' surface, which can still preserve the contour shape well thanks to the simplification method. It can be observed that the resulting extrusion mesh model using the finally simplified point set has a smoother surface than that using the point set directly after the simplification. In addition, to make sure the extruded model has an equivalent scale as the original authentic model, for each point set, we calculate the diagonal length of its bounding box $Diag^{Pts}$ and the diagonal length of the bounding box of sideview of its corresponding original authentic mesh model $Diag^{S}$ as illustrated by blue dash lines in Fig. 1(b) of the paper. We then scale down the coordinates of the finally simplified point set by a factor of $Diag^{S}/Diag^{Pts}$. We also move the center of the

bounding box of the point set to the origin (0, 0). The top right image in Supp. Fig. 1 shows an example of the final simplified point set displayed in FreeCAD.

In Step 5, to obtain the extrusion model, we need to specify the extrusion depth in the orthogonal direction (z-axis) on the sideview. We use the z dimension size of the bounding box of the original authentic 3D mesh model as the extrusion depth. We store the x and y coordinates of the simplified point set and the extrusion depth in a CSV file in Step 4, which will be used as input for a Python script using the FreeCAD Python API to generate a 2D sideview sketch first and then extrude the sketch to an extrusion mesh model.

We develop a set of Python scripts that fully automate the whole process, and the scripts are made open-source for the community (*https://github.com/Xingang1990/TEVAE*).

Part 3: Feature Representation

We present more details of how we apply the as-consistent-as-possible (ACAP) algorithm [2] to obtain feature representations of 3D mesh shapes. Given a set of 3D mesh shapes with the same topology, each shape is represented by S_m, where $m \in [1, \dots, n]$. $p_{m,i} \in \mathbb{R}^3$ is denoted as the i^{th} vertex of the m^{th} shape S_m. The first shape S_1 is the reference shape. Let N_i represent the index set of 1-ring neighbors of the i^{th} vertex on a 3D shape. 1-ring neighbors are all adjacent vertices that are connected to a vertex with one edge. We can then get the deformation matrix $T_{m,i} \in \mathbb{R}^{3\times3}$ that represents the local shape deformation by Supp. Equation (5).

$$\arg\min_{T_{m,i}} \sum_{j\in N_i} c_{i,j} ||(p_{m,i} - p_{m,j}) - T_{m,i}(p_{1,i} - p_{1,j})||_2^2 \tag{5}$$

where $c_{i,j}$ is the cotangent weight. Using polar decomposition, we can then get $T_{m,i} = R_{m,i}S_{m,i}$, where $R_{m,i} \in \mathbb{R}^{3\times3}$ is an orthogonal matrix representing rotation deformation and $S_{m,i} \in \mathbb{R}^{3\times3}$ is a real symmetry matrix describing the scale and shear deformation. $\log R_{m,i}$ is a skew-symmetry matrix, from which we can extract 3 entries (i.e., the upper triangular matrix excluding the diagonal elements that are always zeros). Additionally, we can get 6 entries (i.e., the upper triangular matrix that includes diagonal elements) from $S_{m,i}$ since it is a symmetry matrix. Thus, for each vertex, 9 features can be obtained and concatenated to a 9-dimensional vector $q_{m,i}$. In a result, a mesh shape with V vertices can be represented by a feature matrix $M \in \mathbb{R}^{\mathrm{V}\times9}$. The first shape S_1 is a uniform cube with 9602 vertices, which is the same as the one used in the registration process, so $V = 9602$ in our applications.

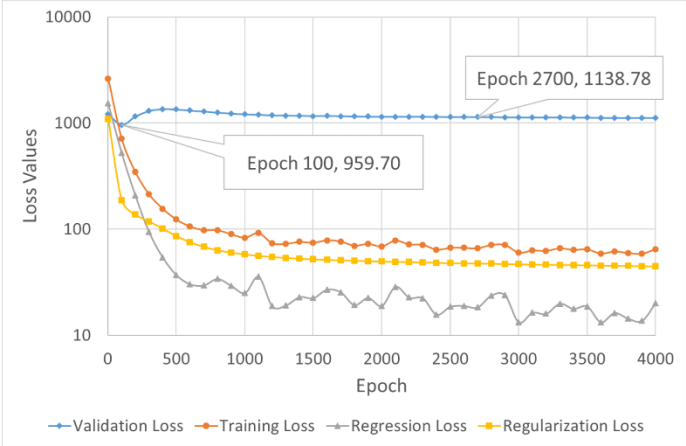
Part 4: Training Details of the Approach

We present the detail of the training process using the case study of car design as an example, and the training of the neural network for mug shapes generation follows the same procedure. We apply a two-stage training strategy to train the E^2D network. We set $\lambda = 1$ in the total loss function as shown in Supp. Equation (1), the value of which does not affect the training in our case *per se*, because we actually minimize L_1 and L_2 independently.

For training the mesh VAE in Stage 1 (i.e., $Enc_1(\cdot)$ and $Dec(\cdot)$), the input target shape dataset is randomly split into the training set (80%) and testing set (20%). The generalization ability of the mesh VAE (i.e., whether the trained model is overfitting to the training data or not) is evaluated by the validation loss on the testing data (unseen data). It is also worth investigating the impact of the L_{KL} term on the performance of the

generative network. By trial-and-error, we find that $\alpha_1 = 40$ and $\alpha_2 = 10$ for Supp. Equation (2) produces the best results in terms of the reconstruction loss, validation loss, KL-divergence loss, and the regularization loss. In minimizing the losses, the Adam optimizer is applied with a learning rate of $lr_1 = 0.0001$. The batch size is set as 32, and the training batches are randomly sampled from the training dataset. We train the mesh VAE 4000 epochs and save the best model that has the least validation loss.

(a)



(b)

Supplemental Figure 2: (a) The loss values from Stage 1 training of the car models; (b) The loss values from Stage 2 training of the car models

For training the $Enc_2(\cdot)$ network in Stage 2, we also do an 80-20 split, and meanwhile use the data pair to make sure the i^{th} target shape (S_t^i) corresponds to the i^{th} source shape (S_s^i) in both the training and testing sets. This ensures that the 20% testing shapes are always unseen data. We set $\alpha = 1$ in Equation (2) of the paper. The optimizer, the learning rate lr_2, and the number of epochs follow the same setting of Stage 1.

For stage 1, the loss values during the training process of car models are reported in Supp. Fig. 2(a). It can be observed that the network is learning, and all loss terms start converging at around epoch 1000. The regularization loss is also gradually reduced, which can prevent the network from overfitting to the training dataset, and thus improve the generalization ability of the network. For stage 2, the loss values in every 100 epochs are shown in Supp. Fig. 2(b). Note that the validation loss achieves the least value at epoch 100 and then converges to a higher loss value. However, in light of the total loss and the overall performance of the $Enc_2(\cdot)$ network, we select epoch 2700 as the best model, which has the second least validation loss, but with much lower regression loss and regularization loss compared to those at epoch 100.

REFERENCES FOR SUPPLEMENTARY MATERIALS

[1] Yuan, Y.-J., Lai, Y.-K., Yang, J., Duan, Q., Fu, H., and Gao, L., 2020. "Mesh variational autoencoders with edge contraction pooling". *In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pp. 274–275.

[2] Gao, L., Lai, Y.-K., Yang, J., Ling-Xiao, Z., Xia, S., and Kobbelt, L., 2019. "Sparse data driven mesh deformation". *IEEE transactions on visualization and computer graphics.*

644 **Figure Captions List for Supplementary Materials**

645

Supp. Fig. 1 An example showing the comparison between resulting extrusion mesh models using a point set after the approximation method and the final simplified point set

Supp. Fig. 2 (a) The loss values from Stage 1 training of the car models; (b) The loss values from Stage 2 training of the car models

646

647