Learning Texture Generators for 3D Shape Collections from Internet Photo Sets

Rui Yu[1,2]
1996yurui@gmail.com
Yue Dong[2]
yuedong@microsoft.com
Pieter Peers[3]
ppeers@siggraph.org
Xin Tong[2]
xtong@microsoft.com

[1] University of Science and Technology of China
[2] Microsoft Research Asia
[3] College of William & Mary

Abstract

We present a method for decorating existing 3D shape collections by learning a texture generator from internet photo collections. We condition the StyleGAN [15] texture generation by injecting multiview silhouettes of a 3D shape with SPADE-IN [23]. To bridge the inherent domain gap between the multiview silhouettes from the shape collection and the distribution of silhouettes in the photo collection, we employ a mixture of silhouettes from both collections for training. Furthermore, we do not assume each exemplar in the photo collection is viewed from more than one vantage point, and leverage multiview discriminators to promote semantic view-consistency over the generated textures. We verify the efficacy of our design on three real-world 3D shape collections.

1 Introduction

Despite various novel 3D representations ranging from image-based to neural models, the classic disentangled shape+texture model is still the most commonly used in downstream applications. Image-based models represent a single instance of a real-world object which cannot be easily altered. Learned neural models only generate 3D shape or a tangled representation where both shape and texture are encoded in the learned features, and either lack detail or do not guarantee consistency over different views. Classic disentangled shape+texture models naturally appear consistent over different views due to the explicit mapping of the 2D texture on a 3D geometry. Parameterized 2D texture presentations are also compatible with existing mature application pipelines. However, generation/synthesis of disentangled shape and texture is not trivial and labor-intensive, even if either the shape or texture distributions is known. For example, numerous shape datasets have been collected for training deep networks. Yet, most of these shape datasets do not include textures or only contain very simplistic textures. Adding realistic textures to an existing shape dataset is currently difficult and labor-intensive.

In this paper we assume that a shape generator or collection of 3D shapes is provided a-priori, and we focus on the problem of generating textures for these shapes, where the user can control the texture content in a shape consistent way. Even this "simplified" problem is challenging as it is impractical to collect a large amount of high quality shape+texture exemplars without resorting to sophisticated 3D scanning methods. On the other hand, it is relatively trivial to gather a large collection of 2D projections of shape+texture in the form of photographs. However, unless specifically captured for this purpose, most image collections do not guarantee that all possible views for each object are contained in the dataset.

To this end, we present a framework for training a shape-aware texture generator from a set of photographs collected from uncontrolled sources (*e.g.*, internet photo repositories). To decouple the texture parameterization from the geometry complexity, we employ a canonical-view projection-based texture atlas and use StyleGAN [15] to generate a texture for each canonical view's chart. To generate shape-aware textures, we condition StyleGAN by SPADE-IN [23] with the corresponding silhouettes for each canonical view. At training time, we do not have direct access to the full texture, but only a sampling of 2D projections (*i.e.*, photographs). Therefore, during generator training, we map the textures to the shapes and render them for different views and check the visualizations against discriminators trained on the photo collections. However, a single view only encodes partial information of the texture. To ensure semantically consistent texture generation across views, we utilize a multi-discriminator architecture [16] where the different discriminators validate the rendering for different overlapping views. A key challenge is that the distribution of the condition (*i.e.*, the distribution of silhouettes from the shape collection or shape generator) needs to match distribution of the silhouettes in the photo sets. Even more, the distributions of shapes in the photographs is possibly inconsistent between the different views, making it difficult to obtain a consistent silhouette distribution. Ignoring this *silhouette-condition* gap can lead to a lower quality texture generator and even model collapse. We alleviate the impact of this silhouette-condition gap during generator training by drawing the condition from a mixed distribution of the source (*i.e.*, shape) and target (*i.e.*, photographs) domains' silhouettes, making the generator conditioned to both the silhouette of the source (*i.e.*, shape) and the target (*i.e.*, photograph) domain.

We validate our texture generator on different shape collections and internet photo sets, and show that our generator is able to produce high quality view-consistent textures.

2 Related work

Texture generation is a classic computer vision problem [29] and recent advances have leveraged the power of neural networks [8, 11, 27]. However, these classic methods are only concerned with unconditionally replicating the texture's spatial distribution. Conditional image generation can be used to generate textures conditioned on given constraints [14, 23, 28, 30]. However, such generation methods only operate in the 2D texture domain and do not consider the specifics of the underlying 3D shape. Texture suggestion [4, 22] properly applies existing textures to a 3D shape, but requires significant semantical labeling of both the shape and the dataset. Texture Fields [21] is a conditional texture generator that given a shape and an image of the object, produces the full texture; training Texture Fields require matching image-shape pairs. When applied to unconditional texture generation (*i.e.*, without the image condition), Texture Fields tends to produce either blurry textures (VAE-based) or artifacts (GAN-based). Our method is akin to unconditional Texture Fields, combining both the

goals of texture synthesis and texture suggestion, to generate texture distributions while ensuring semantic consistency with the underlying shape without the need for manual labeling or matching shape-image pairs.

Neural rendering [14, 19, 20, 23, 28, 32] shares the goal of generating view consistent imagery from image collections. A crucial difference is that neural rendering aims to retain the shape and texture of the objects in the training data, and typically encodes shape and texture together, making disentangled texture and shape control difficult. More importantly, while neural rendering methods offer view consistency, typically, it is not consistent with a single 3D shape over all views. Our method naturally ensures consistency over different views with a single shape and enforces shape and texture disentanglement by design.

Having an underlying 3D representation naturally provides view-consistency. Volume generators can also be trained from image collections [7, 12, 16], and the training photographs can be reprojected via the learned generator [17]. However, such an approach does not guarantee a semantically meaningful texturing of unseen surface points. Moreover, the quality of the reprojected image greatly depends on the volume resolution or number of primitives. Pavllo et al. [24] propose a GAN framework for generating both shape and texture jointly, and as a consequence it cannot cleanly disentangle shape and texture. Furthermore, this method relies on deforming a suitable template mesh that also defines the UV mapping. Instead of an explicit mapping, an implicit representation such as Neural Radiance Fields (NeRF) [18] improves texture quality, and extensions have shown separate control of shape and texture [25]. Similarly, Chan et al. [2] adopt the Sinusoidal Representation Network (SIREN) [26] to further improve quality. While such methods offer to explore the shape and texture distribution separately, it is difficult to decorate an a-priori provided 3D geometry (*i.e.*, with unknown corresponding latent code in the implicit representation).

3 Method

3.1 Overview

Our method takes as input a collection of 3D shapes, and a large set photographs of the same class of objects captured from a variety of viewpoints; we do not assume each object is seen from multiple viewpoints or that the objects in the 3D shape collection exactly match the objects in the photographs. Furthermore, we assume the availability of a foreground segmentation for each photograph (*e.g.*, obtained automatically [10] or manually).We assume all the 3D models are oriented consistently.

The goal is to train a texture generator that can produce semantically consistent textures for each shape in the collection, and where the user has control of the texture "style" (via a latent vector 'z') conditioned on the shape. We formulate the texture generation as a shape-conditioned StyleGAN generator. To bridge the difference between the output and training domains (*i.e.*, 2D texture space vs. photographs), we employ a multi-discriminator architecture [16] where each discriminator acts on a rendering of the texture and shape for a particular view range (Figure 1).

3.2 Texture Parameterization

Before detailing the training and design of the texture generator, we first need to specify the texture parameterization that maps the 2D texture domain to the 3D shape. A shared

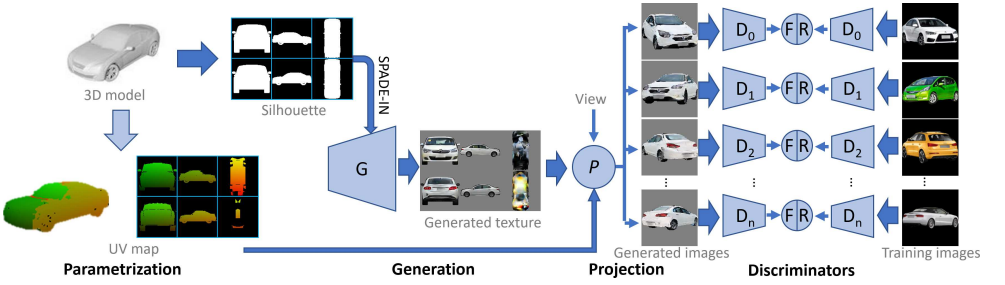

Figure 1: To train a shape-aware texture generator for a set of 3D shapes, we automatically parameterize the 3D models into a view-based texture atlas (bottom-left also shows the atlas mapped onto the target shape), and provide the shape-conditions by SPADE-IN the silhouettes. The shape+texture can then be visualized from different viewpoints, and tested against multiple discriminators, each trained on photograph subsets with similar viewpoints.

consistent texture parameterization over all shapes in the dataset simplifies the texture generation task. We posit that a good parameterization roughly follows the semantics of the shape (*e.g.*, identifiable regions such as a car tire are assigned to the same regions in the texture domain), and that the parameterization has a small number of contiguous texture charts. The former links semantic texture generation to its location in the texture, whereas the latter facilitates exploitation of local texture neighborhoods during synthesis. However, fully automatic semantic alignment for general shapes is challenging. Our solution exploits that we only desire a texture parameterization for a set of shapes of a narrow *common* class in which most semantically meaningful parts are in a similar location; we assume that the shapes are consistently oriented in the shape dataset. We therefore opt for using a view-based texture projection parameterization in which each chart in the texture is determined by projecting the mesh to a small set of predetermined viewpoints. In practice, we define a front, back, left, right, top and bottom viewpoint, and assign the projected coordinate as a vertex's texture coordinate. For vertices visible in multiple views, we assign it only to one of the projected views to ensure each surface point has a unique texture coordinate; surface points not visible in any projection will be ignored for texture synthesis. In our implementation we assign the vertex to the first chart in which it is visible; clearly this assignment is order sensitive and we manually assign the order per dataset. Besides its simplicity, our view-based texture parameterization has two practical advantages. First, it ensures that we have large contiguous charts making it easier to exploit neighborhood relations. Second, it allows us to encode the shape condition as the silhouettes for each view (and thus texture chart).

3.3 Texture Generator

As noted before, we follow the powerful StyleGAN [15] generator and discriminator network architecture. Unlike StyleGAN, we employ multiple discriminators for different views similar to MD-GAN [16]. We employ a separate StyleGAN generator for each texture chart. To promote consistency among the different chart textures, the different StyleGAN generators share the convolution weights and only differ in the trainable *"constant starting vector"*.

We use SPADE-IN [23] to provide the shape condition in the form of the silhouette for the corresponding view (used for chart parameterization). Unlike regular SPADE-IN, where the bias and scale of the instance normalization layer is only determined by the SPADE-IN

map (*i.e.*, the silhouette), we combine SPADE-IN with the style feature of the StyleGAN. Practically, we follow StyleGAN to generate the style feature vector w from the latent vector z. Next, we perform a element-wise multiplication between the style feature vector w and the silhouette map, and use the resulting multi-channel feature maps similarly as a regular style vector. Thus, the actual SPADE-IN signal is the original w augmented by the shape silhouette, encoding both texture (via the latent code z) and shape (via the silhouette).

To obtain successful texture generators based on the above network structure, three critical issues need to be addressed: inter-chart semantic consistency, background detection bias, and the distribution gap between the silhouettes in the shape collection versus the photo set.

Inter-Chart Semantic Consistency Although the charts are generated independently by different generators, these generators are trained together. Each discriminator checks a rendered image of the generated texture (mapped to a shape) against the distribution of exemplar images viewed from a similar viewpoint. If all projected views pass all the discriminators, we consider the texture generation successful. To ensure semantically consistent textures between charts, it is important that each view includes texels from multiple charts, such that each discriminator can provide gradients to multiple generators simultaneously, thereby enforcing multi-chart semantic consistency. Therefore, we include viewpoints that differ from the canonical chart-projection views (*e.g.*, midway between the left and front view). Without these overlapping views, each chart's texture is trained independently and while it produces plausible textures per chart, these are semantically decoupled from the other charts.

We follow Li et al. [16] to obtain rough viewpoint estimates from the silhouette image (via a view-estimation network trained on rendered silhouettes from the shape dataset) and partition the training images in different categories based on the viewpoint categories. In case of ambiguous silhouettes (*e.g.*, the front and back of a car look similar), we provide additional manual labeling to disambiguate both cases. Furthermore, we sample the distribution of viewpoints in each partition when rendering the synthesized textures for each discriminator to ensure a similar view distribution between rendered images and the photographs.

Background Detection Bias Unlike the renders of the textured shapes, the photographs also contain a background. To avoid biasing the discriminator to detect photographed backgrounds, we replace the background in each photograph by a solid color. While this resolves the issue of the discriminator leveraging the background, there is the danger that now the texture generator will ignore the silhouette and 'fill-in' texels within the silhouette with the background color. To avoid this, we apply a random (gray-scale) color to the background in the rendering/photograph such that the generator cannot predict the background color.

Silhouette Distribution Gap In general, the distribution of silhouettes in the shape collection does not match the distribution of silhouettes in the image collection. Furthermore, biases in the photo collection can even result in different silhouette distributions between the different 'views' of the data. This difference in silhouette distributions consequently also results in a difference in the generated images versus the photographs. As a consequence, the discriminator might memorize the silhouette shapes and leverage these to distinguish between real and synthesized images thereby ignoring the content of the generated textures.

To bridge this gap, we introduce a *proxy* conditional image generation problem that given a silhouette from the photo collection (and latent vector z) directly synthesizes an image. Unlike texture generation, the conditional image generation has been shown to have a well-defined equilibrium solution and can be successfully trained with existing GAN training strategies. Since our texture-charts are view-based, there is no real difference between the image and texture generator/discriminator. Hence, we can interleave the training of each chart's texture generator with the proxy condition (*i.e.*, image silhouette) and with the 3D

shape collection condition (*i.e.*, shape silhouette), and use the same generator/discriminator for both, to regularize the discriminator training to focus on content rather than silhouette.

Another potential gap between the photographs' silhouettes and the shape silhouettes is the distribution of size and position in the image. Therefore, we apply a random scale (0.7-0.9 scale factor) and a random shift (from 0 pixels to the image boundary) before passing an image to the discriminator.

4 Results

We use three different datasets to demonstrate the training of texture generators:

1. *Cars*: we use the 3D car models from ShapeNet [3], and manually remove shapes with esoteric or extreme shapes, retaining 552 shapes for texture generation training and 32 shapes for testing. We use 6 charts (front, back, left, right, top, and bottom view). In addition, we collected 16K real-world car photographs from the CompCars dataset [31]. All photographs are cropped to contain only the car and resampled to 128×128 resolution, and we use the default Mask R-CNN model [10] to extract the silhouettes. We use 5 discriminators: for the front and back views, as well as a single discriminator for the (left & right) side views, one for the (left & right) side front, and one for the (left & right) side back view. After assigning the photographs based on viewpoint, we obtain $1618, 1368, 2526, 6090$, and 4856 photographs for the *Front*, *Back*, *Side*, *Side-Front* and *Side-Back* views respectively.
2. *Faces*: we use the FFHQ dataset [15] which contains $60K$ training images. We fit a 3D face model and extract view information following [5] and further refine the face model parameters and viewpoints using the same training loss for individual face images. We randomly draw 3000 shapes for training and another 32 shapes for testing. Since all facial features can be seen from the frontal view, we only use a single frontal chart. Note, the 3D face model dataset does not contain teeth geometry, and thus our model does not generate teeth textures either. We use 3 discriminators: for the front, right, and left side views.
3. *Shoes*: we mine photographs of shoes from the Internet, and manually classify them into 8 view slots (*i.e.*, front, back, left, right, front-left, front-right, back-left, and back-left). The masks of the shoes are manually segmented. Our image dataset consists of 7K images uniformly distributed over all the view slots. We gathered 122 shoe geometries from an online repository [1] which where re-meshed to fix topological errors, and finally manually aligned. To further enrich the shape dataset, we randomly scale along each axis within the range of $[0.9, 1.1]$, resulting in 488 shoe shapes in total. The parameterization and discriminator follows the same setup as the *Cars* dataset. After grouping the photographs based on viewpoint, we obtain $1300, 1424, 2087, 1087$, and 962 photographs for the *Front*, *Back*, *Side*, *Side-Front*, and *Side-Back* views respectively.

Figure 2 shows 3D models of each of the above datasets visualized with textures generated by our generators. The resulting textured models can be visualized with any existing 3D rendering system and yields natural consistency over the different viewpoints. The explicit separation of shape and texture allows us to generate the same texture (*i.e.*, same z vector) adapted for each shape as well as generate different textures for the same shape (*i.e.*, varying the z vector). We refer to the supplemental document for more results.

Figure 2: Rendering results of synthesized textures conditioned on the shape. For each example, we vary the shape, shown for 4 different views, over the columns (*i.e.*, fixed z) and the texture over the rows (*i.e.*, different z).

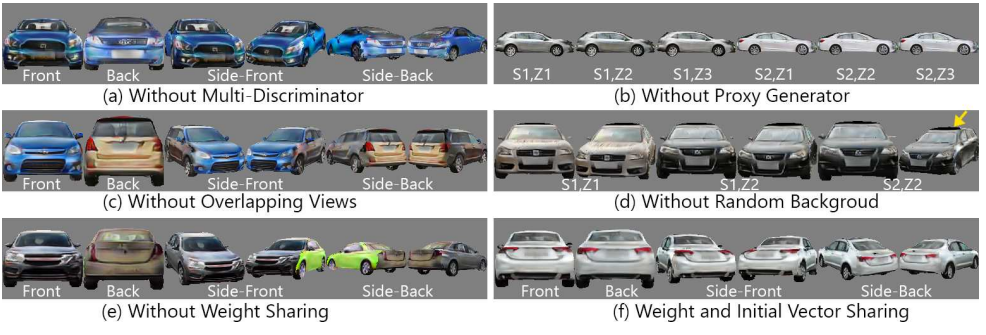

Figure 3: **Ablation Study Results** (a) Without using multiple discriminators the texture generator is unable to distinguish front and back. (b) Mode collapse occurs when not using the proxy image generation. (c) Inter-chart semantic consistency is lost when not overlapping the discriminator views. (d) A fixed background color (black) during training allows the generator to use this color to ignore the silhouette (yellow arrow). (e) Weight sharing is necessary to promote inter-chart consistency. (f) Without a learned constant initial vector the network cannot distinguish charts with similar silhouettes such as front and back.

5 Discussion

5.1 Ablation Study

We validate the effectiveness of our design decisions (*i.e.*, multiple discriminators, proxy image generation, overlapping view partitioning, random background, shifting/scaling, and weight sharing) in a set of ablation studies on the *Cars* dataset. Table 1 list the numeric results of these tests. We use FID [13] to measure the quality and diversity of the model, and GIQA [9] to measure the image quality of individual images to avoid bias by the silhouette

Table 1: Ablation study results for the *Cars* dataset (GIQA is scaled by 10^2).

	Ours	No Multi. Discr.	No Proxy	No Overlap	Fixed BG	No Shift /Scale	No Share	Full Share
GIQA ↑	**9.910**	9.828	9.803	9.813	9.895	9.885	9.848	9.837
FID ↓	**32.59**	33.75	33.02	33.71	36.23	33.65	34.00	32.80

(*i.e.*, shape) distribution gap. Both FID and GIQA are measured on all 3D shapes.

Multi-discriminator We employ multiple discriminators to better model the differences in distribution between the different views. To validate the necessity of using multiple discriminators, we train a texture generator using a single discriminator for all views (Table 1, 3rd column). Figure 3 (a) shows that the resulting texture discriminator fails to differentiate between the front and back of the car, and thus generates a front side texture for both.

Proxy Image Generation We introduce the proxy image generation to address the silhouette distribution gap between the image collection and the shape collection. Figure 3 (b) shows the results of the texture generator trained without interleaving the proxy image generation. The lack of variation for different z is indicative of a mode collapse and is confirmed in the numerical errors (Table 1, 4th column).

Overlapping views To promote inter-chart consistency, we employ overlapping discriminator views. Figure 3 (c) and Table 1, 5th column, shows the results of a generator trained with only 3 discriminators for the (non-overlapping) front, back, and side-views. The resulting synthesized textures exhibits clear disjunct charts with different colors and texture details.

Random Background To avoid that the generator ignores the silhouette, we randomly change the color of the background during training. Figure 3 (d) and Table 1, 6th row, shows the results of training a texture generator using only a black background. While the results look reasonable from a frontal view, we can see that at oblique angles that the texture generator 'fills in' the black background color inside the silhouettes.

Shift/Scaling In addition to using the proxy generator to bridge the distribution gap, we also randomly shift and scale the images. As a result, omitting this shifting and scaling results in a notable decrease in quality (Table 1, 7th row).

Weight Sharing We share the generator network weights, but with different constant vectors for each chart, to promote a consistent style (due to the shared weights) between the different charts while at the same time adapting to the specifics of each chart (via different constant vectors). To validate the effectiveness of this design decision, we train two additional texture generators: one with shared weights *and* constant initial vector, and one without sharing any weights (and initial vector). Figure 3 (e,f) and Table 1, columns 8 & 9, show that neither achieves the same quality. When sharing everything, we can see that the generator cannot distinguish between the front and back because they have the same silhouette. In contrast, the fully independent generators (without sharing any weights) produce inconsistent charts.

5.2 Comparison to Prior Work

To our knowledge, there does not exist a method that has the same goals as ours. Therefore, we compare our method to state-of-the-art prior works that shares some of our goals. First, we compare our method to multi-view neural view synthesis methods: HoloGAN [19], GRAF [25], and piGAN [2] on the Car shapes but with the CAR-LA dataset [6, 25]; a synthetic dataset commonly used for view synthesis. Figure 4(a) compares the generated image quality for randomly selected shapes from the dataset not used for training. In contrast to

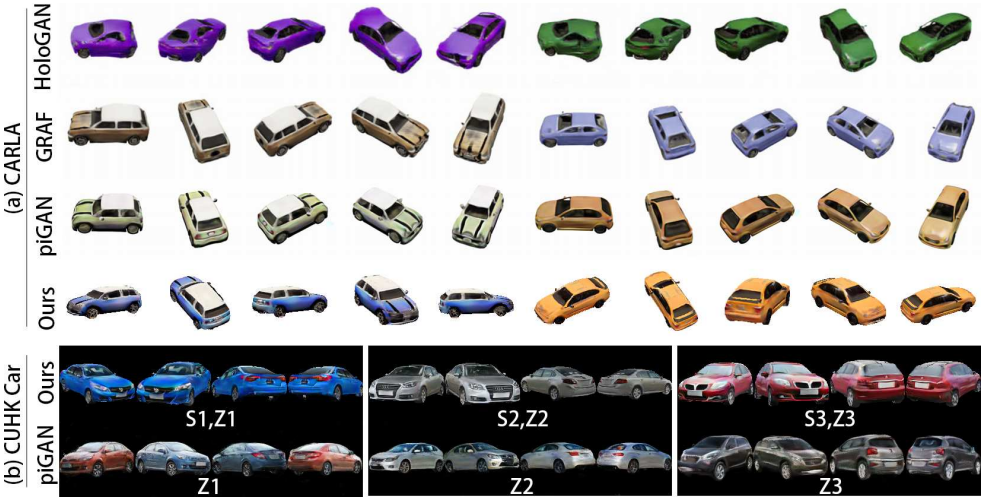

Figure 4: **Comparison** (a) Our texture generator trained on the CAR-LA dataset produces reasonable textures with guaranteed view consistency comparable to results from HoloGAN[19], GRAF [25] and piGAN[2] (the results from the first three rows are from [2]). (b) Compared to our method, piGAN trained on the CUHK-Car dataset fails to maintain view consistency and produces lower quality results.

Table 2: Quantitative comparison against HoloGAN and piGAN. (GIQA is scaled by 10^2). Furthermore, we ran a user study on the CUHK-Car dataset to validate image quality and view consistency.

	CAR-LA			CUHK-Car			User study (CUHK-Car)			
	Holo GAN	pi GAN	Ours	Holo GAN	pi GAN	Ours		Holo GAN	pi GAN	Ours
GIQA↑	5.128	**6.417**	6.289	6.335	9.870	**9.910**	Qual.	1.67	39.80	**58.53**
FID↓	85.80	30.49	**28.32**	87.95	44.78	**32.59**	Cons.	40.00	29.20	**82.50**

HoloGAN, our method maintains view consistency, while producing comparable quality to piGAN, albeit on a different shape than the CAR-LA shape. When training piGAN on the real-world *CUHK-Car* dataset, we see (Figure 4(b)) that it struggles with the uneven view distribution and the increased level of detail, and comparably it does not reach the same quality level as our method.

A quantitative comparison using FID [13] and GIQA [9] between the different methods on both the *CAR-LA* and *CUHK-Car* datasets can be found in Table 2. We use the pre-trained piGAN model for the *CAR-LA* dataset provided by the authors [2], and train both HoloGAN and piGAN with the respective authors' released code for the other cases. All error numbers are calculated using the same validation set. Our method produces comparable image quality (*i.e.*, GIQA) than piGAN and it outperforms HoloGAN. However, when considering the distribution quality (*i.e.*, FID), our method significantly outperforms both prior methods.

Finally, we conducted a user study on 12 subjects to ascertain the perceptual quality of our method compared to prior methods trained on the *CUHK-Car* dataset. To assess the image quality, the participant is shown three images generated by each of the methods, and the user is asked to identify the most realistic one. We perform 40 trials per user, and for each

method we report the percentage that the generated image was preferred (Table 2 'Qual.'). To assess view consistency, we show the participants four views of the same instance, and asked the user whether all images represent the same object (*i.e.* car). This experiment was repeated for 30 trials per user, and we report the percentage that the views are considered consistent (Table 2, 'Cons.'). The user study indicates that our method is perceived to produce more realistic images as well as more consistent visualizations.

5.3 Limitations

Our method is not without limitations. First, a good texture generator requires good training data; the shape database needs to contain sufficiently varied shapes and the objects in the photographs need to be similar in shape. We introduced a number of strategies to reduce the impact of the training gap (*i.e.*, proxy image generation, shifting/scaling, and multi-discriminators), however, there is a limit to how much they can bridge. Second, our texture parameterization cannot well handle shapes with complex geometrical features (*e.g.*, with self-occlusions), and it relies on semantic localization in the view-based charts (*e.g.*, deformable objects would be difficult to handle). Third, our method has difficulty handling thin geometrical features because the resulting narrow silhouette cues will dominate the texture cues in the discriminator. Finally, our method mainly focuses on promoting low level semantic consistency, and it cannot learn high level semantics (*e.g.*, it might produce a shoe with different logos in different charts).

6 Conclusion

We present a method for learning texture generators for 3D shape collections from internet photo sets. Our method builds on the StyleGAN architecture. We employ a texture generator per chart in a view-based texture parameterization. In addition, we augment the StyleGAN architecture with multiple discriminators for different views. Each discriminator operates on visualizations of a shape with generated texture rendered from a random viewpoint within a predefined view cone. To encourage inter-chart semantic consistency, we use overlapping view cones for the discriminators. To condition the texture generator, while retaining control on the texture 'style', we augment the latent vector z by a pixel-wise multiplication of the silhouette. To bridge the inherent distribution gap between the shapes' silhouettes and the silhouettes in the photo collection, we introduce a proxy conditional image generation problem on the silhouettes of the 3D shapes. We alternate between updating the discriminators by training the texture generator conditioned by the silhouettes from the shape data set and the proxy image generator conditioned by the silhouettes from the photo collection. We demonstrated our method on three different shape collections, and performed an extensive ablation study to justify our design decisions.

For future work we would like to combine learning a shape and texture generator from internet photo collections. We believe the presented method offers a first step in this direction.

Acknowledgments Pieter Peers was supported in part by NSF grant IIS-1909028.

References

[1] Cgtrader. https://www.cgtrader.com/.

[2] Eric R. Chan, Marco Monteiro, Petr Kellnhofer, Jiajun Wu, and Gordon Wetzstein. Pi-gan: Periodic implicit generative adversarial networks for 3d-aware image synthesis. In *CVPR*, pages 5799–5809, June 2021.

[3] Angel X. Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, Jianxiong Xiao, Li Yi, and Fisher Yu. ShapeNet: An Information-Rich 3D Model Repository. Technical Report arXiv:1512.03012 [cs.GR], Stanford University — Princeton University — Toyota Technological Institute at Chicago, 2015.

[4] Kang Chen, Kun Xu, Yizhou Yu, Tian-Yi Wang, and Shi-Min Hu. Magic decorator: Automatic material suggestion for indoor digital scenes. *ACM Trans. Graph.*, 34(6), October 2015.

[5] Yu Deng, Jiaolong Yang, Sicheng Xu, Dong Chen, Yunde Jia, and Xin Tong. Accurate 3d face reconstruction with weakly-supervised learning: From single image to image set. In *CVPR Workshop on Analysis and Modeling of Faces and Gestures*, 2019.

[6] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. Carla: An open urban driving simulator. In *Conference on robot learning*, pages 1–16. PMLR, 2017.

[7] Matheus Gadelha, Subhransu Maji, and Rui Wang. 3d shape induction from 2d views of multiple objects. In *2017 International Conference on 3D Vision (3DV)*, pages 402–411. IEEE, 2017.

[8] Leon Gatys, Alexander S Ecker, and Matthias Bethge. Texture synthesis using convolutional neural networks. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, editors, *NIPS*, volume 28, 2015.

[9] Shuyang Gu, Jianmin Bao, Dong Chen, and Fang Wen. Giqa: Generated image quality assessment. In *ECCV*, pages 369–385. Springer, 2020.

[10] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *ICCV*, pages 2961–2969, 2017.

[11] Eric Heitz, Kenneth Vanhoey, Thomas Chambon, and Laurent Belcour. A sliced wasserstein loss for neural texture synthesis. In *CVPR*, pages 9412–9420, June 2021.

[12] Philipp Henzler, Niloy J Mitra, and Tobias Ritschel. Escaping plato's cave: 3d shape from adversarial rendering. In *ICCV*, pages 9984–9993, 2019.

[13] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, Günter Klambauer, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a nash equilibrium. In *NIPS*, 2017.

[14] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. *CVPR*, 2017.

[15] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *CVPR*, pages 4401–4410, 2019.

[16] Xiao Li, Yue Dong, Pieter Peers, and Xin Tong. Synthesizing 3d shapes from silhouette image collections using multi-projection generative adversarial networks. In *CVPR*, pages 5535–5544, 2019.

[17] Yiyi Liao, Katja Schwarz, Lars Mescheder, and Andreas Geiger. Towards unsupervised learning of generative models for 3d controllable image synthesis. In *CVPR*, June 2020.

[18] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. In *ECCV*, pages 405–421. Springer, 2020.

[19] Thu Nguyen-Phuoc, Chuan Li, Lucas Theis, Christian Richardt, and Yong-Liang Yang. Hologan: Unsupervised learning of 3d representations from natural images. In *ICCV*, pages 7588–7597, 2019.

[20] Thu Nguyen-Phuoc, Christian Richardt, Long Mai, Yong-Liang Yang, and Niloy Mitra. Blockgan: Learning 3d object-aware scene representations from unlabelled images. In *NIPS*, Nov 2020.

[21] Michael Oechsle, Lars Mescheder, Michael Niemeyer, Thilo Strauss, and Andreas Geiger. Texture fields: Learning texture representations in function space. In *ICCV*, 2019.

[22] Keunhong Park, Konstantinos Rematas, Ali Farhadi, and Steven M. Seitz. Photoshape: Photorealistic materials for large-scale shape collections. *ACM Trans. Graph.*, 37(6), November 2018.

[23] Taesung Park, Ming-Yu Liu, Ting-Chun Wang, and Jun-Yan Zhu. Semantic image synthesis with spatially-adaptive normalization. In *CVPR*, pages 2337–2346, 2019.

[24] Dario Pavllo, Graham Spinks, Thomas Hofmann, Marie-Francine Moens, and Aurelien Lucchi. Convolutional generation of textured 3d meshes. In *NIPS*, 2020.

[25] Katja Schwarz, Yiyi Liao, Michael Niemeyer, and Andreas Geiger. Graf: Generative radiance fields for 3d-aware image synthesis. In *NIPS*, 2020.

[26] Vincent Sitzmann, Julien Martel, Alexander Bergman, David Lindell, and Gordon Wetzstein. Implicit neural representations with periodic activation functions. *NIPS*, 33, 2020.

[27] Dmitry Ulyanov, Vadim Lebedev, Andrea Vedaldi, and Victor S Lempitsky. Texture networks: Feed-forward synthesis of textures and stylized images. In *ICML*, volume 1, page 4, 2016.

[28] Ting-Chun Wang, Ming-Yu Liu, Jun-Yan Zhu, Andrew Tao, Jan Kautz, and Bryan Catanzaro. High-resolution image synthesis and semantic manipulation with conditional gans. In *CVPR*, 2018.

[29] Li-Yi Wei, Sylvain Lefebvre, Vivek Kwatra, and Greg Turk. State of the art in example-based texture synthesis. In *Eurographics 2009, State of the Art Report, EG-STAR*, pages 93–117. Eurographics Association, 2009.

[30] Wenqi Xian, Patsorn Sangkloy, Varun Agrawal, Amit Raj, Jingwan Lu, Chen Fang, Fisher Yu, and James Hays. TextureGAN: Controlling deep image synthesis with texture patches. *CVPR*, 2018.

[31] Linjie Yang, Ping Luo, Chen Change Loy, and Xiaoou Tang. A large-scale car dataset for fine-grained categorization and verification. In *CVPR*, pages 3973–3981, 2015.

[32] Jun-Yan Zhu, Zhoutong Zhang, Chengkai Zhang, Jiajun Wu, Antonio Torralba, Joshua B. Tenenbaum, and William T. Freeman. Visual object networks: Image generation with disentangled 3D representations. In *NIPS*, 2018.