Learning to Generate Textures on 3D Meshes

Amit Raj
Georgia Institute of Technology

Cusuh Ham
Georgia Institute of Technology

Connelly Barnes
Adobe Research

Vladimir Kim
Adobe Research

Jingwan Lu
Adobe Research

James Hays
Georgia Institute of Technology

Abstract

Recent years have seen a great deal of work in photorealistic neural image synthesis from 2D image datasets. However, there are only a few works that exploit 3D shape information to aid in image synthesis. To this end, we leverage data from 2D image datasets as well as 3D model corpora to generate textured 3D models. In particular, we propose a framework for texturing meshes from multiview images. Our framework first uses 2.5D information rendered using the 3D models along with user inputs as an intermediate view dependent representation. These intermediate representations are then used to generate realistic textures for particular views in an unpaired manner. Finally, we use a differentiable renderer to combine the generated multiview texture into a single textured mesh. We demonstrate results of multiview texture synthesis for example meshes.

1. Introduction

There has been significant work in deep generative modeling in recent years [8, 23]. GANs and VAEs have been employed in a number of applications such as inpainting, conditional generation, and high resolution image synthesis. Most works, however, focus on unconstrained or constrained generation of *2D images*.

In a similar stride there has been some work in generation of 3D geometry from latent codes or from image collections [24]. However, there have been few works that leverage 3D information to guide image synthesis.

Furthermore there have been approaches for novel view synthesis, which has been explored primarily in the context of 2D conditional generative models [12] . Such a representation is inherently limited in that there is no explicit knowledge of the 3D geometry of the scene or object for which a novel view needs to be synthesized.

On the other hand, there have been approaches for 3D understanding from images in terms of meshes, point clouds, or voxels [7, 10, 19]. There are a number of representations that can be used for incorporating 3d information, each having its own benefits.

In this work, we aim to leverage the rich geometry information available from 3D corpora and texture information available from 2D datasets in order to generate textured 3D meshes. 3D model datasets usually have rich geometric information for a model class but generally lack or often have very simplistic textures.

On the other hand, image collections provide rich photorealistic details about the texture of the objects but do not contain explicit 3D information. By leveraging information available in both these datasets, we can generate textured models of objects, which we can then visualize from arbitrary view points.

An application of particular interest is synthesis of user-guided colors or textures as demonstrated by Scribbler and TextureGAN [14, 20]. Generating images that incorporate user guidance is useful in content authoring for amateur artists. Allowing a user to be able to guide the synthesis of textures while borrowing the statistics from real images would speed up pipelines involving painting texture and advertising. In addition, image synthesis from sparse guidance is beneficial in use cases where the user needs to explore different styles quickly, for instance cars of assorted colors or handbags of different materials and textures.

We propose a two-stage approach to generate textured 3D models. First, we render different 2.5D information (normalized object coordinates, masks, albedo) from a set of viewpoints for the mesh. These representations allow us to use existing deep architectures for handling 2D images and inject some degree of 3D information. We then use these synthetic renderings to generate textured images for a set of views using a network trained with unpaired 2D/ 3D data.

The generated realistic textures match the statistics of real image collection. These generated textures can then be reprojected to consistent world coordinates to obtain a textured 3D point-cloud which can then be visualized from novel view points. A naive aggregation of generated textures form each viewpoint in the form of a point cloud exhibits inconsistencies in terms of color and lighting. To address this, we additionally generate textured meshes by using a differentiable renderer to reconcile the textured images from multiple views.

Further, we show that our choice of representation allows for either control of global color and fine grained color control.

To summarize our contributions are threefold:

- We propose a two-stage architecture to texture 3D models by first generating textured multiview images from rendered 2.5D information and then reconciling the multiview textures using a differentiable renderer.
- Our choice of representation and network allows for direct or indirect color control of generated multivew images
- The differentiable renderer allows for incorporation of local texture properties in a geometry aware manner.

2. Related Works

There are several works which deal with conditional image synthesis and synthesis of geometry.

2.1. Generative models

GANs and VAEs have been used in a variety of generative tasks. Primarily, GANs have been demonstrated to perform well on generation of high-resolution images [8, 9], image inpainting [6], and image-to image-translation. CycleGAN [22] and UNIT [11] demonstrated image translation between disparate domains with unpaired data. BiCycleGAN[23] and MUNIT[5] extended the above to diverse translation from a single image. The control on the diversity in both these cases are either not interpretable or are limited to selection from a set of modes as in BiCycleGAN. We intend to address this issue by allowing fine grained user guidance for texture synthesis.

2.2. 3D understanding

Works such as MarrNet [7, 24] have demonstrated inferring geometry from single or multiview images. Kanazawa et al [7] learn to infer geometry from image collections and subsequently obtain a textured mesh from the inferred geometry and input image. Delanoy et al. [4] have demonstrated synthesis of geometry from sketches presented in canonical views. There have also been a number of works in the graphics community that learn to align and deform models to input images to infer geometry of model. Our proposed approach works in conjunction with such models as we assume that we have access to 3D models, either ground truth or inferred, which we can use for guiding the texture synthesis.
Another line of work relies on using existing 3D models and 2D image collections along with a materials database to learn to associate materials to regions of a 3D model, allowing for photorealistic re-lightable renderings of textured 3D models [13] . Our work is similar in spirit to this idea, however we eschew the need for a material dataset by training an end-to-end generative model that in addition to learning to associate texture to parts, also learns to generate view-dependent textures.

2.3. Synthetic to real

A number of works generate synthetic data to aid the learning process. The idea is that we can use renderings of 3D models with precise knowledge regarding viewpoint, lighting or texture to provide supervisory signals to train generative networks, which are otherwise unavailable or prohibitively expensive to obtain from natural images. This helps in combining synthetic renderings and real 2D images to learn realistic textures. SFMNet [17] uses synthetic renderings of faces to learn an intrinsic image decomposition, allowing for relighting of faces. Shrivastava et al.[15] synthetic renderings of eyes along with a small subset of real eye images to train a model that can generate richer set of realistic looking eye images. GIS [1] use a generative model to insert vehicles in outdoor driving scenes to augment driving datasets. We borrow ideas from literature in this area to generate a set of intermediate images that aids in the learning process.

2.4. User-guided generation

A lot of image-to-image translation are unguided, or have sparse guidance regarding generated textures. BiCycleGan provides some degree of control and diversity of generation at the cost of interpretability. The approach presented by Scribbler and TextureGan[14, 20] allow for generation of realistic interpretable texture and color information. StarGAN [3] provide coarser control in that they can generate images in a particular set of domains. However, these methods are unable to reason about the geometry of the object or the scene, and hence the color or texture incorporation might be inadequate in some cases. Our method aims to address this issue by explicitly reasoning about the geometry of the image.

3. Approach

Our approach consists of two stages–first we use 2.5D information rendered from a mesh to generate textures for multiple views for a given model. These textures are learned using unpaired 2.5D and image data similar to CycleGAN. The generated textures from multiple views are reconciled using a differentiable renderer. We show that we have color control over the generated textures.

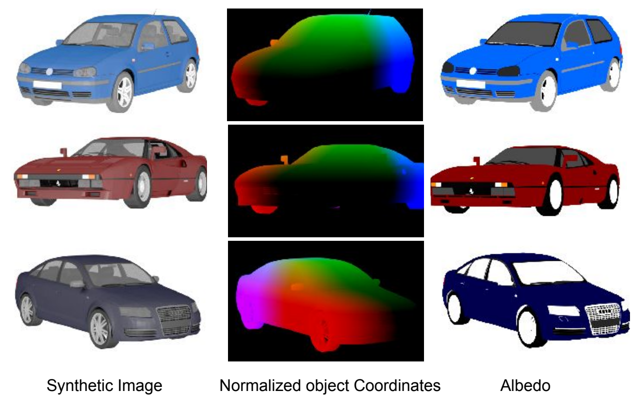

Figure 1. We render 2.5D information from different viewpoints for each mesh, which are used as inputs to our multiview texture synthesis stage.

3.1. Training Data:

We use models from the ShapeNet database for 3D data and render 2.5D information. Our training data consists of the following: For a given mesh (V, F) with $V \in \mathcal{R}^{N\times3}$ and $F \in \mathcal{R}^{M\times3}$ in each of the model classes we generate the following 2.5D channels : the Normalized object coordinate representation (S_{noc}), albedo (S_{albedo}), shaded synthetic rendering (S_{syn}) and object mask (S_{mask}) for several different viewpoints v_i. We represent this set of inputs as S^{v_i} for each view. The NOC representation as presented in [18] is the space contained in a unit cube around the object, each pixel of a NOC image represents the (x, y, z) coordinate of the point in some canonical orientation.

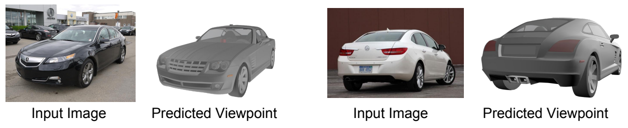

Figure 2. Viewpoints predicted for input images using the RenderForCNN framework.

The viewpoints are sampled from a distribution that matches the statistics of the viewpoint distribution of real images. To determine this distribution, we use the pre-trained models from RenderForCNN [16]. This gives us an estimate of the approximate distribution of azimuth and elevation values for the objects in our 2D training images. To generate the synthetic data, we normalize the the given 3D model, and sample azimuth and elevation angles from the estimated distribution on a sphere of fixed radius.

3.2. Framework:

We propose a two stage approach to generate textures for meshes by using only unpaired supervision available from image collections. Our framework consists of the Image-texturing stage T_1 which takes as input the geometry information S^{v_i}(stacked tuple of NOC, albedo,mask) to generate images I^{v_i} corresponding to a viewpoint and a mesh-texturing stage T_2 that combines textures from multiple views $\{I_{v_i}\}_{i=1}^{n}$ into a single consistent textured mesh.

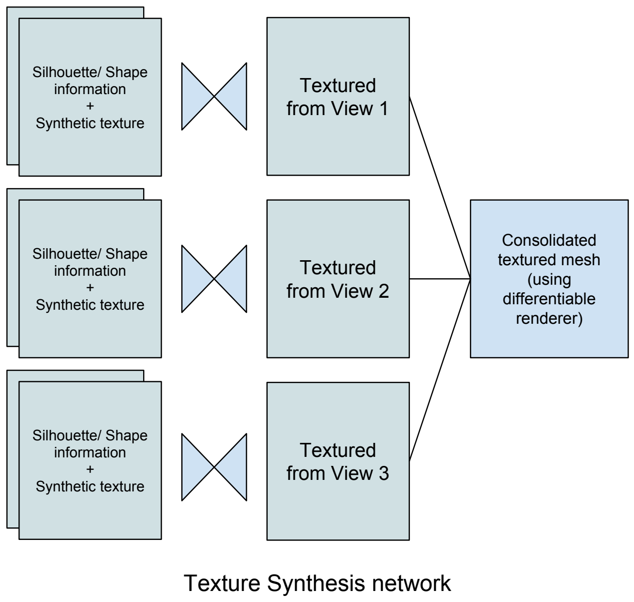
Silhouette/ Shape information + Synthetic texture
Textured from View 1
Silhouette/ Shape information + Synthetic texture
Textured from View 2
Silhouette/ Shape information + Synthetic texture
Textured from View 3
Consolidated textured mesh (using differentiable renderer)
Texture Synthesis network

Figure 3. Our framework consists of two stages. In the first stage, we render 2.5D information(Normalized Object coordinates, albedo and masks) in multiple views and generate realistic textures for these views using CycleGAN [22]. In the second stage, we use a differentiable renderer to consolidate textures from multiple views into a single textured mesh.

3.2.1 Multiview texture-Synthesis stage:

We train the multiview texture-synthesis stage in an unpaired manner using an architecture similar to CycleGan[22] and VON[24]. Since there is no direct supervision between the The 2.5D guiding channels and the available image collection, we rely on unpaired image to image translation models to learn an association between 2.5D geometry information and natural images. The network generates outputs whose statistics match real images for each view.

We inject global and fine grained color control by using albedo channel stacked with NOC and mask as input. In this setting, the network is encouraged to generate textures that match the real image collection but are guided by simple flat shaded colors from the synthetic renderings. We use the synthetic textures provided with Shapenet.

Stage 1 consists of the following inputs: Stacked tuple of NOCS maps, albedo and mask of the object $S_i^{v_i}$ for a particular viewpoint v_i. The framework consists of two networks : (F) that converts stacked 2.5D information $(S_i^{v_i})$ into real image domain$(\mathcal{R})$ and G that converts a real image into stacked 2.5D information. It is trained with the following 3 components:

$$L_{cycle_r} = ||G(F(x)) - x||_2^2 \tag{1}$$

$$L_{cycle_s} = ||G(F(y)) - y||_2^2 \tag{2}$$

Where $x \sim S_i^{v_i}$ and $y \sim \mathcal{R}$

Additionally, we have an adversarial objective where the discriminators D_f and D_g that learns to classify real and generated images in each domain,

$$L_{adv_f} = E_{s_{v_i} \sim S_i^{v_i}}[\log(D_f(F(x)))] + E_{y \sim \mathcal{R}}[\log(1 - D_f(y))]$$

$$L_{adv_g} = E_{y \sim \mathcal{R}}[\log(D_g(G(x)))] + E_{s_{v_i} \sim S_i^{v_i}}[\log(1 - D_g(y))]$$

We generate image textures for multiple viewpoints from this stage. However, since the only loss guiding it are the cycle loss and adversarial loss, the generated images have inconsistencies between viewpoints, in terms of specular lighting and luminance of the texture across the surface. That is we have a textures $\{I_i^v\}$ for each view point v_i. To account for this, we aggregate the generated image textures in the next stage.

3.2.2 Mesh-Texturing Network

This stage uses a neural mesh renderer to combine all the generated textures into a single consistent textured mesh. We assume that we have access to a UV parameterization of the desired mesh *a priori*. We then start with a candidate texture that needs to be optimized and render the mesh with the candidate texture from multiple viewpoints.

For the given views $\{v_i\}_{i=1}^n$ and a given texture atlas to be optimized $I_{uv} \in \mathcal{R}^{H \times W \times C}$, we use a differentiable renderer $R(I_{uv}, v_i) : \mathcal{R}^{H \times W \times C} \times \mathcal{R}^3 \rightarrow \mathcal{R}^{H \times W \times C}$, to optimize the following:

$$L_{recon} = \|R(m, v) - I^v\|^2$$

$$L_{feat} = \sum_i w_i \, \|\phi_i((R(m, v)) - \phi_i(I^v)\|$$

where L_{recon} is a pixel-wise reconstruction loss, L_{feat} is the feature loss between the rendered image and the generated image; $\phi_i(I)$ are feature maps obtained from the i-th layer of a pretrained feature extractor (VGG-19 here)

$$L_{tv_{uv}} = \sum_{i,j} |(I_{uv}(i+1, j) - I_{uv}(i, j))| + |(I_{uv}(i, j+1) - I_{uv}(i, j))|$$

$$L_{tv_{proj}} = \sum_{i,j} |(I^v(i+1, j) - I^v(i, j))| + |(I^v(i, j+1) - I^v(i, j))|$$

Additionally, we impose a total-variation loss both in UV-space and in the projected space to promote consistency and smoothness of texture. For known views, the final loss is given by:

$$L_{total} = \lambda_{recon} * L_{recon} + \lambda_{feat} * L_{feat} + \lambda_{tv_{uv}} * L_{tv_{uv}} + \lambda_{tv_{proj}} * L_{tv_{proj}} \tag{3}$$

Imposing only supervised losses, causes inconsistent textures, hence we also render the model in viewpoints v_j for which we don't have strict supervision and minimize the following additional loss term

$$L_{global} = \|\phi_k((R(m, v)) - \phi_k(I^v)\| \tag{4}$$

for a random $I^v \sim I_i^v$. That is, for views with no supervision, we randomly sample a generated image I_v, and match the feature of the rendered image at a very deep layer of a pretrained feature extrator. This serves the dual purpose of maintaining global consistency in the optimized texture as well as filling certain regions of the UV map, that was not visible in any view. Hence, during the unsupervised training phase we have the following:

$$L_{total} = \lambda_{global} * L_{global} + \lambda_{tv_{proj}} * L_{tv_{proj}} \tag{5}$$

4. Experiments

4.1. Dataset and Architectures

For the 3D models we used the Shapenet[2] cars classes and rendered a number of 2.5D channels used during training. All the meshes are processed to generate a UV atlas over a 1024*1024 image using the auto-uv feature of blender. The UV maps can be obtained from any reasonable auto UV tool. For real image collections, we used the Comprehensive cars dataset from CUHK[21]. This dataset contains over 1000 models of cars in various viewpoints. We clustered the images based on viewpoint and only used a filtered subset as mentioned. The dataset is divided into front, back, side, and 3/4th views. We determine the distribution of azimuth and elevation angles of the cars in the dataset using RenderForCNN. We observe a clustering of views and only sample cars from a few of the clusters with the largest number of samples.

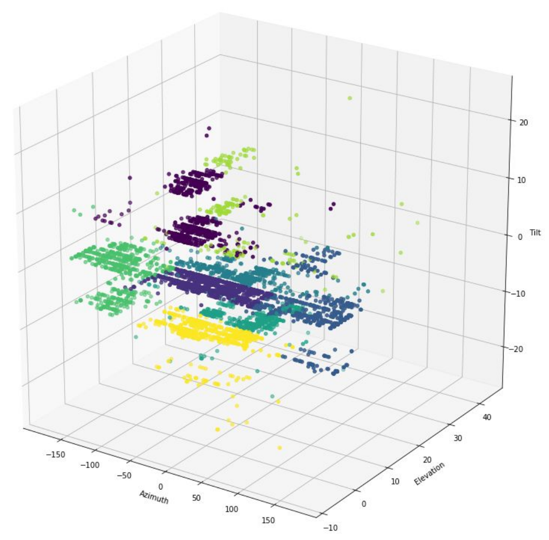

Figure 4. The viewpoints in the cars dataset are clustered into a set of views. We filter the dataset to consider views only from the largest central clusters.

4.2. Qualitative Evaluations

We use the cars class from ShapeNet to demonstrate color guided texture synthesis. We want to demonstrate explicit global and fine grained control on the colors. For this purpose we use a CycleGAN model that takes as input the NOCS representation and the albedo or flat shaded channel rendered image and generates a textured image in the proposed view. Additionally, we jitter the images in LAB space to increase the variance of the color captured. We

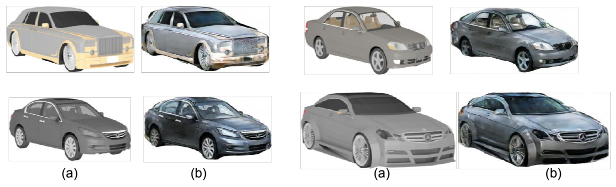

Figure 5. Results of texture synthesis network for cars dataset. (a) Synthetic textured images from a single viewpoint. (b) Realistic textured images generated from same viewpoint.

notice that to generate realistic textures, the network bakes the lighting into generated texture from every view. However this generated lighting is not consistent across multiple views. The flat shaded image then defines the fine grained color information (for instance, patterns and logos) that can otherwise not be captured by unguided generation. Additionally, global color control is afforded by simply changing the 'ab' channels of the flat shaded image as in 6. We notice that this is not same as changing the 'ab' channel of the

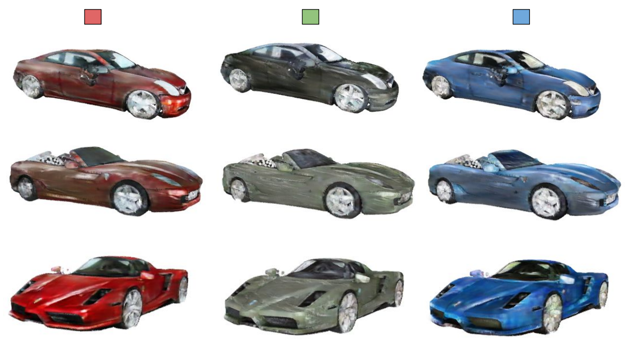

Figure 6. Our framework is able to generate cars with realistic textures compliant with the input color. Each row represents a different 3D model, And each column contains images generated by altering the 'ab' channel of the flat shaded image in the input. We see that, this has a different effect than just changing the 'ab' channel of the generated texture

generated image, as some colors do not look realistic w.r.t to the data distribution.

4.3. Quantitative Evaluation

It is not particularly straightforward to measure the quality of the final textured mesh, as we do not have reference textures for the mesh. We propose a proxy evaluation by re-projecting the final textured mesh in a number of views and calculating the inception score of the projected images and the FID of the projected images compared to the real data distribution. We look at both these scores together to get a complete idea of the performance of our framework. We notice that the images from the final textured mesh have a lower FID score than images rendered with synthetic textures, since our final textured mesh better respects the statistics of real images. Additionally, we notice that the rendered synthetic images have the highest inception score, here IS measures diversity more than quality of the texture. Since the synthetic images have a wider range of colors and textures the IS score indicates that the synthetic distribution is more diverse than the real data distribution. Images generated by projecting our textured mesh has an IS in between the real and synthetic images since it is more diverse than the real image distribution because of the color control whilst still restricted to the statistics of real image distributions, hence not exhibiting as much color variation as the synthetic images.

	IS	FID
Synthetic	**2.06 ± 0.32**	23.25 ± 0.42
Real	1.43 ± 0.14	-
Ours	1.79 ± 0.28	**21.21 ± 0.49**

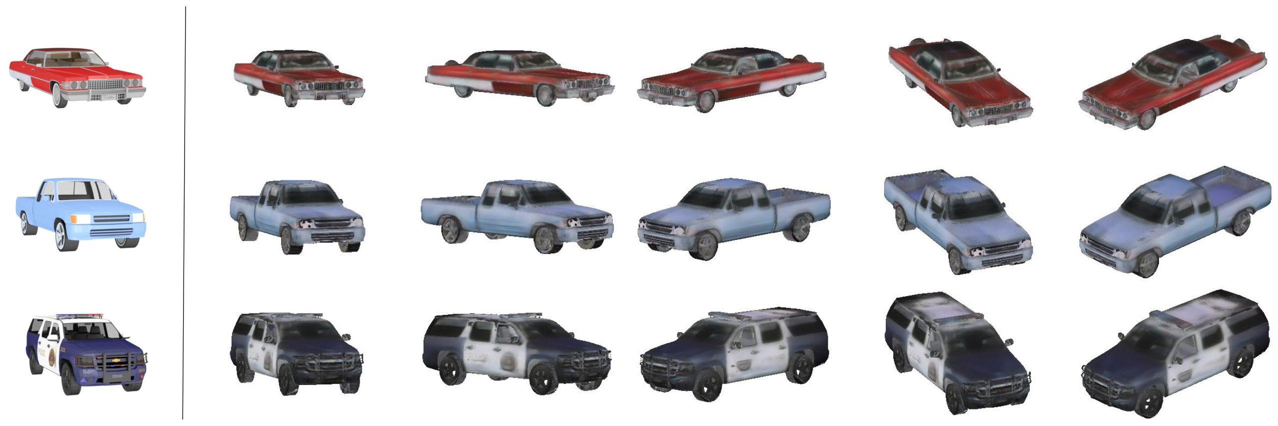
Figure 7. Qualitative results for aggregated textures. Given rendered synthetic images from multiple viewpoints we generate realistic textures for each viewpoint and aggregate them to a single textured mesh. The first column represents the synthetic cars in some viewpoint. Columns 2-6 represent realistic textures aggregated using 3 views (left, right and front) projected onto different views

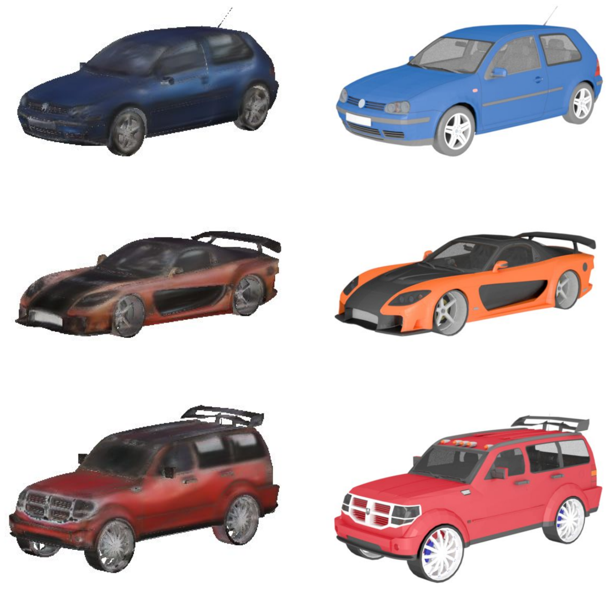
Figure 8. The textured mesh obtained by multiview aggregation projected to a new viewpoint (left). Synthetic texture from the same viewpoint (right). We see that the generated texture capture more lighting and shading information than the synthetic textures.

5. Discussion and Conclusion

We have demonstrated preliminary results towards learning to generate textures for mesh from image collections. We present a two stage approach to generate multiview textures and aggregate them into a single textured mesh for the cars class. We intend to explore extensions to more general class of models and geometries. Additionally, we notice that the framework requires an estimate of the distribution of viewpoints of the objects for better performance. Making the model rely less on this requirement would help the model reason better about the textures of the model. We believe that our framework would be beneficial to tasks in novel-view synthesis, geometry aware texture generation and user guided mesh texturing.

References

[1] H. A. Alhaija, S. K. Mustikovela, A. Geiger, and C. Rother. Geometric image synthesis. *arXiv preprint arXiv:1809.04696*, 2018. 2

[2] A. X. Chang, T. Funkhouser, L. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su, et al. Shapenet: An information-rich 3d model repository. *arXiv preprint arXiv:1512.03012*, 2015. 4

[3] Y. Choi, M. Choi, M. Kim, J.-W. Ha, S. Kim, and J. Choo. Stargan: Unified generative adversarial networks for multi-domain image-to-image translation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8789–8797, 2018. 2

[4] J. Delanoy, M. Aubry, P. Isola, A. A. Efros, and A. Bousseau. 3d sketching using multi-view deep volumetric prediction. *Proceedings of the ACM on Computer Graphics and Interactive Techniques*, 1(1):21, 2018. 2

[5] X. Huang, M.-Y. Liu, S. Belongie, and J. Kautz. Multimodal unsupervised image-to-image translation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 172–189, 2018. 2

[6] S. Iizuka, E. Simo-Serra, and H. Ishikawa. Globally and locally consistent image completion. *ACM Transactions on Graphics (ToG)*, 36(4):107, 2017. 2

[7] A. Kanazawa, S. Tulsiani, A. A. Efros, and J. Malik. Learning category-specific mesh reconstruction from image collections. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 371–386, 2018. 1, 2

[8] T. Karras, T. Aila, S. Laine, and J. Lehtinen. Progressive growing of gans for improved quality, stability, and variation. *arXiv preprint arXiv:1710.10196*, 2017. 1, 2

[9] T. Karras, S. Laine, and T. Aila. A style-based generator architecture for generative adversarial networks. *arXiv preprint arXiv:1812.04948*, 2018. 2

[10] C.-H. Lin, C. Kong, and S. Lucey. Learning efficient point cloud generation for dense 3d object reconstruction. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018. 1

[11] M.-Y. Liu, T. Breuel, and J. Kautz. Unsupervised image-to-image translation networks. In *Advances in Neural Information Processing Systems*, pages 700–708, 2017. 2

[12] E. Park, J. Yang, E. Yumer, D. Ceylan, and A. C. Berg. Transformation-grounded image generation network for novel 3d view synthesis. In *Proceedings of the ieee conference on computer vision and pattern recognition*, pages 3500–3509, 2017. 1

[13] K. Park, K. Rematas, A. Farhadi, and S. M. Seitz. Photoshape: photorealistic materials for large-scale shape collections. In *SIGGRAPH Asia 2018 Technical Papers*, page 192. ACM, 2018. 2

[14] P. Sangkloy, J. Lu, C. Fang, F. Yu, and J. Hays. Scribbler: Controlling deep image synthesis with sketch and color. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5400–5409, 2017. 1, 2

[15] A. Shrivastava, T. Pfister, O. Tuzel, J. Susskind, W. Wang, and R. Webb. Learning from simulated and unsupervised images through adversarial training. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2107–2116, 2017. 2

[16] H. Su, C. R. Qi, Y. Li, and L. J. Guibas. Render for cnn: Viewpoint estimation in images using cnns trained with rendered 3d model views. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2686–2694, 2015. 3

[17] S. Vijayanarasimhan, S. Ricco, C. Schmid, R. Sukthankar, and K. Fragkiadaki. Sfm-net: Learning of structure and motion from video. *arXiv preprint arXiv:1704.07804*, 2017. 2

[18] H. Wang, S. Sridhar, J. Huang, J. Valentin, S. Song, and L. J. Guibas. Normalized object coordinate space for category-level 6d object pose and size estimation. *arXiv preprint arXiv:1901.02970*, 2019. 3

[19] J. Wu, C. Zhang, T. Xue, B. Freeman, and J. Tenenbaum. Learning a probabilistic latent space of object shapes via 3d generative-adversarial modeling. In *Advances in neural information processing systems*, pages 82–90, 2016. 1

[20] W. Xian, P. Sangkloy, V. Agrawal, A. Raj, J. Lu, C. Fang, F. Yu, and J. Hays. Texturegan: Controlling deep image synthesis with texture patches. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8456–8465, 2018. 1, 2

[21] L. Yang, P. Luo, C. Change Loy, and X. Tang. A large-scale car dataset for fine-grained categorization and verification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3973–3981, 2015. 4

[22] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE international conference on computer vision*, pages 2223–2232, 2017. 2, 3

[23] J.-Y. Zhu, R. Zhang, D. Pathak, T. Darrell, A. A. Efros, O. Wang, and E. Shechtman. Toward multimodal image-to-image translation. In *Advances in Neural Information Processing Systems*, pages 465–476, 2017. 1, 2

[24] J.-Y. Zhu, Z. Zhang, C. Zhang, J. Wu, A. Torralba, J. Tenenbaum, and B. Freeman. Visual object networks: Image generation with disentangled 3d representations. In *Advances in Neural Information Processing Systems*, pages 118–129, 2018. 1, 2, 3