Synthesizing 3D Shapes from Silhouette Image Collections using Multi-projection Generative Adversarial Networks –Supplemental Material–

Xiao Li
University of Science and Technology of China

Yue Dong
Microsoft Research Asia

Pieter Peers
College of William & Mary

Xin Tong
Microsoft Research Asia

This supplemental material includes additional results, discussions, and applications to further demonstrate the advantages of our multi-projection GAN framework. We organize this supplemental material in 3 parts: section 1 includes additional results of the 3D shape synthesis pipeline as detailed in the main paper. Section 2 discusses the application of multi-projection GAN to generate spatially-varying surface reflectance from 2D images only. This extension introduces different types of projection operators, further demonstrating the abilities of multi-projections GANs to learn distributions and correlations from multiple different types of sources. Finally, in section 3 we show an example of how multi-projection GAN can learn the 2D image distribution from the MNIST dataset from 1D observations only.

1. Part I: Additional Results for 3D Shape Generation

Network Structure For completeness, we reiterate the network structure of our *VP-MP-GAN* and provide a visual illustration of the structure. We follow the 3D voxel generator network structure of Wu *et al*. [11] which takes an input vector sampled from $N(0,1)$ and outputs a $64 \times 64 \times 64$ voxel grid via a sequence of 3D convolution and upsample layers. We employ batch normalization after each convolution and upsample layer and use the ReLU activation function.

The discriminators take a 64×64 binary silhouette image as input, and output a single scalar probability value. Each discriminator contains 4 convolutional blocks with a stride of 2, followed by a single fully-connected layer. We share the first convolutional layers among the different discriminators, and use spectrum normalization [7] for each layer with the LeakyReLU activation function.

The view prediction network shares the same convolutional structure as the discriminator network, but with using batch normalization instead of spectrum normalization, and outputting the probability for the 16 discrete view-bins. Figure 1 details the network structure of the generator, discriminator and view prediction networks.

Additional results Figures 2, 3, and 4 show additional results for the *bird*, *chair*, and *car* dataset. We visualize the generated shapes from additional viewpoints to better highlight the quality of the 3D shapes. We would like to highlight that all the results are trained from real photographs only. Figure 5 shows a selection of the training data (both the silhouette image as well as the original source photograph not used for view prediction) assigned to the different viewpoints by our view prediction network.

2. Part II: SVBRDF Synthesis using Multi-Projection GANs

To further demonstrate the flexibility and efficiency of our multi-projection GAN, we train it for Spatially-Varying Bidirectional Reflectance Distribution Function (SVBRDF) modeling of *wood*, *plastic*, and *metal* materials. First, we will briefly introduce the definition of the SVBRDF model. Followed by a description of the data and training setup. Finally, we discuss and analyze the results.

2.1. SVBRDF Definition

We make a number of assumptions common in SVBRDF modeling. We assume the material surface is roughly planar, thus each pixel corresponds to a surface point. We characterize the surface reflectance at each surface point with four reflectance property maps: an RGB diffuse albedo map (k_d), a monochrome specular albedo map (k_s), a surface normal map ($\mathbf{n}$), and a monochrome roughness map (m). These property maps in turn drive a Cook-Torrance [3] BRDF model to describe the angular variations of the

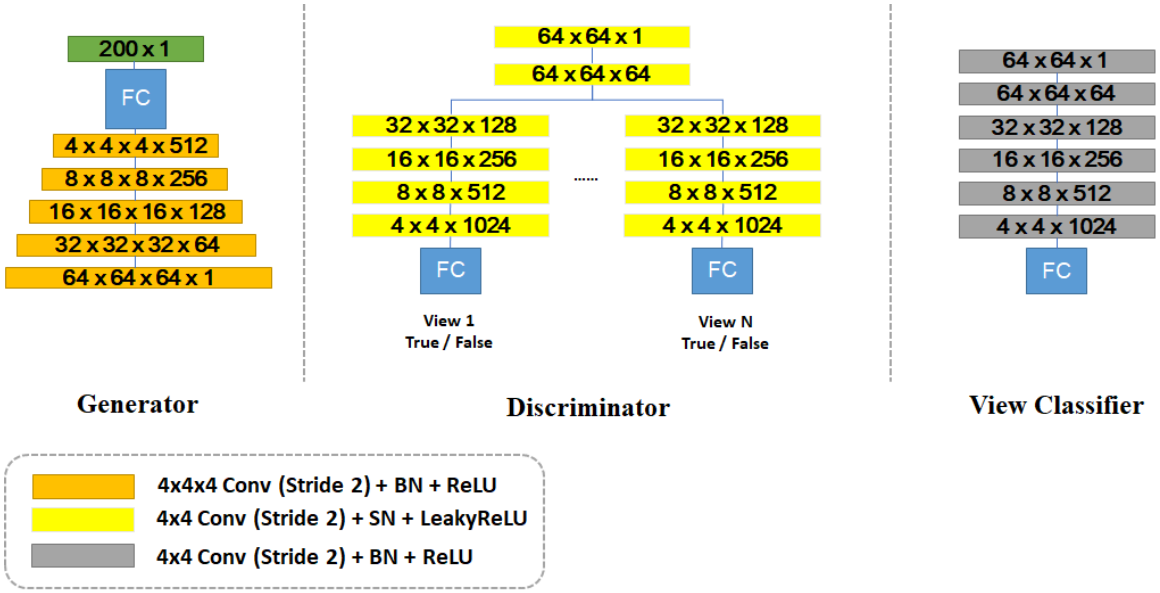

Figure 1. Network structure for *VP-MP-GAN* for synthesizing 3D voxel shapes.

SVBRDF at each surface point:

$$\mathbf{f_r}(\mathbf{x}, \omega_i, \omega_o) = \frac{k_d(\mathbf{x})}{\pi} + \tag{1}$$

$$\frac{k_s(\mathbf{x})}{4} \frac{\mathbf{D}(\mathbf{h}, m(\mathbf{x}))\mathbf{G}(\omega_i)\mathbf{G}(\omega_o)\mathbf{F}(\mathbf{h}, \eta)}{\cos(\theta_i)\cos(\theta_o)}, \tag{2}$$

where $\mathbf{x}$ is the spatial position, and $\omega_i = (\phi_i, \theta_i)$ and $\omega_o = (\phi_o, \theta_o)$ are the incident and outgoing directions, $\mathbf{h}$ is the halfway vector between ω_i and ω_o, $\mathbf{D}(\cdot, \cdot)$ is the spatially-varying Beckmann microfacet distribution parameterized by roughness $m(\mathbf{x})$, $\mathbf{G}(\cdot)$ is the shadowing and masking term, and $\mathbf{F}(\cdot, \cdot)$ is the Fresnel reflectance computed by Schlick's approximation. We store each spatially varying diffuse albedo $k_d(\mathbf{x})$, specular albedo $k_s(\mathbf{x})$, roughness $m(\mathbf{x})$, and surface normals $\mathbf{n}(\mathbf{x})$ in texture maps. Our goal is to learn the distribution of these four property maps for a class of materials, such as wood, plastic, or metal.

2.2. Network Description

We follow the basic architecture of DCGAN [8] for both the generator and discriminator networks. As shown in Figure 6 the generator takes as input $\mathbf{Z} \in \mathbb{R}^{512}$ randomly drawn from a uniform distribution $U(-1, 1)$, and it produces 64×64 output images. We employ the LeakyReLU activation function [6] and use batch normalization [4] for each layer, except for the last layer which employs a sigmoid activation function. We employ four identical networks for each property map (i.e., diffuse albedo, specular albedo, normal map, and roughness map) that share the first two layers; we expect the different maps to share the same high level features. Similarly, the discriminator contains a succession of convolutional layers with the LeakyReLU as an activation function, and each layer is followed by a batch normalization step. To efficiently deal with the large variations between the output maps, we also output a global scale factor for the diffuse albedo, specular albedo, and roughness maps. To generate the final property map, we multiply each of these maps with their respective scale factor. All discriminators use the same architecture, each differing in their data dimensions and projection operators. We also employ mini-batch discrimination [9] for the discriminators.

2.3. Projection Operators

We consider two projection types: a selection and a rendering projection.

Selection Projection: The selection projection directly outputs one of the property maps. Hence, there are four variants of the selection projection that each outputs either the diffuse albedo map, the specular albedo map, the normal map, or the specular roughness map. The selection projection has no additional latent parameters, and it serves to inform what a typical albedo map, normal map, or roughness map "looks like". While these maps are sufficient to model the separate spaces of each property map, they do not provide any cues on the component interrelations, and thus additional projection types are needed.

Rendering Projection: The rendering projection takes as latent parameters viewpoint and the parameters for one of three types of lighting conditions: natural environmental lighting, directional lighting, and constant ambient lighting. The output is an image of the material seen from the viewpoint and under the a randomly selected lighting sampled from the lighting class. The view is always aimed at the material map center. The projection from the four property maps to a rendered image is governed by:

$$I(\mathbf{x}, \omega_o) = \int_\Omega \mathbf{f_r}(\mathbf{x}, \omega_i, \omega_o) \max(\cos\theta_i, 0) \mathbf{L}(\omega_i)\, \mathrm{d}\omega_i, \quad (3)$$

where ω_o is the direction towards the camera sensor and which is defined by the camera's intrinsic and extrinsic parameters, $\mathbf{f_r}$ is defined by Equation 2, and $\mathbf{L}(\omega_i)$ is the incident lighting defined over the sphere of directions Ω. Note that this rendering process assumes that the natural lighting is distant (i.e., only it depends on direction and not position), and that there is no interreflection or shadowing. We evaluate Equation 3 using Monte-Carlo integration for each $\mathbf{x}$ independently. We improve convergence by importance sampling according to the diffuse (first term in Equation 2) and specular part (second term in Equation 2) of the surface reflectance defined at each pixel; we select which BRDF component to sample proportional to their respective albedos. The rendered images will be compared to regular photographs, and thus we need to include the typical camera post-processing steps too. To this end, we simulate auto-exposure by normalizing the lighting based on the total intensity of the upper-hemisphere of the incident lighting, and apply a 2.2 gamma correction and clamp pixel values to 1.0.

2.4. Differential Rendering

For the rendering projection, we employ differential rendering to compute the gradients with respect to the different property maps. The derivatives with respect to the diffuse and specular albedo are relatively straightforward:

$$\frac{\partial I(\mathbf{x})}{\partial k_d(\mathbf{x})} = \frac{1}{\pi} \int_\Omega \max(\cos\theta_i, 0) \mathbf{L}(\omega_i)\, \mathrm{d}\omega_i, \quad (4)$$

$$\frac{\partial I(\mathbf{x})}{\partial k_s(\mathbf{x})} = \frac{1}{4} \int_\Omega \frac{\mathbf{D}(\mathbf{h}, m(\mathbf{x}))\mathbf{G}(\omega_i)\mathbf{G}(\omega_o)\mathbf{F}(\mathbf{h}, \eta)}{\cos(\theta_o)}\, \mathrm{d}\omega_i. \quad (5)$$

Again, both integrals are evaluated using Monte-Carlo integration.

The partial derivative with respect to specular roughness and surface normal are computed with finite differencing. To avoid Monte-Carlo noise aversely impacting the gradients, we reuse the *same* set of samples to compute each term of the finite difference (based on importance sampling of one of the BRDFs). We evaluate the finite difference for roughness for a step of $\Delta m = 0.001$. For the surface normal, we first convert the normal $\mathbf{n} = (x, y, z)$ to spherical coordinates (ϕ_n, θ_n), and compute the 2D finite difference for a step of $\Delta\phi_n = \Delta\theta_n = 0.001$.

2.5. Training Data

We collect two datasets: a large dataset for wood materials intended for validation experiments, and an additional dataset for metals to demonstrate the generality of our method.

In order to compare the quality of our result direct to the results generated by a regular high-dimensional GANs (i.e., training four the four property maps directly), we desire a sufficiently large SVBRDF dataset. Due to lack of such an existing dataset, we rely on artists to help create such a dataset. First, we let artists select $11,000$ image patches of wood materials from the OpenSurfaces dataset [2] which are then fed in to a single image SVBRDF estimation network [5]. This method generates the diffuse albedo map, normal map, and a homogeneous specular roughness and albedo. To obtain a full spatially-varying specular component, we use an artist-defined transfer function that generates the specular component directly from the diffuse albedo map. Finally, the artists manually verify all generated SVBRDFs, and remove implausible SVBRDFs. The resulting final dataset consists of $8,000$ plausible wood SVBRDFs.

For the selection projections, we directly use (a randomly selected subset of) the generated property maps as training data. For the rendering projection, we use the same rendering pipeline to generate training data from (a randomly selected subset of) SVBRDFs. To light the renderings, we collect 98 natural light probes [10] and further augment this dataset by applying random rotations along the up direction. The light and view configurations used for the rendering projections in each experiment are detailed in subsection 2.7.

For the metal and plastic training dataset, we collected $1,000$ artist generated SVBRDFs. However, unlike the wood training dataset, we employ only real images (for the rendering projections) of metal and plastic surfaces from the OpenSurfaces dataset [2]. We classify these selected images in two categories based on their apparent view: top views ($5,000$ images) and grazing views ($5,000$ images). The view ranges for the two image sets are $0\pm5^\circ$ and $80\pm5^\circ$ respectively. All images are taken under unknown natural environmental lighting. No additional processing was preformed to the images as we expect them to be auto-exposed and auto white-balanced.

2.6. Implementation

We implemented our multi-projection SVBRDF GAN (MP-SVBRDF-GAN) in TensorFlow [1], including a differential render layer implemented in CUDA. All the examples in this document are trained using the ADAM opti-

mizer with a 10^{-4} learning rate, $\beta_1 = 0.5$, $\beta_2 = 0.9$. and a $1:1$ training ratio between generator and discriminators.

2.7. Results

In this section, we report our results for SVBRDF modeling of the *wood*, *metal*, and *plastic* datasets, as well as an analysis of our method with respect: to learning the correlations between the different SVBRDF property maps, and to the impact of using multiple types and combinations of projections.

Multi-Projection GAN versus High Dimensional GAN We first compare the efficiency and accuracy on learning the distribution of SVBRDFs for wood using our multi-projection GAN versus a classic high-dimensional GAN trained with a single discriminator and where the data distribution and the target distribution are identical (i.e., we train directly from the corresponded property maps). For the multi-projection GAN we employ six different projections: four selection projections that select the diffuse albedo, specular albedo, specular roughness, and normal maps respectively, and two rendering projections for top view ($0°$ from up direction) and grazing views ($80°$ from the up direction). Both projections are rendered with natural environmental lighting. We use 500 samples per property map; each sample is selected at random and independently for each component, and we use 250 rendered images of randomly selected materials for each rendering projection. We add a $\pm 5°$ random permutation to each view to simulate uncertainty in viewpoint estimation from photographs. We train the classic high dimensional GAN from 650 samples from the high dimensional SVBRDF distribution. Each sample consists of 4 property maps. Hence, the total number of training 'images' is identical ($= 2500$). Figure 7 shows a comparison between a selection of SVBRDFs generated by the two GANs. While our multi-projection GAN can successfully generate wood SVBRDFs, the high dimensional GAN fails to produce reasonable results.

Subspace Training versus Learning Correlations The selection projection is an extreme type of projection that only provides cues to a specific subspace of the output distribution. Hence, it can be interpreted as training a GAN that models a specific component. In general, we expect that a lower dimensional subspace requires less training samples to learn. Correlations between different subspaces are learned from the rendering projections. Figure 8 shows a GAN trained with only the selection projections. As expected, the generated property maps are plausible, however, the correlations between the different property maps are missing.

Our multi-projection GAN can model distributions with correlated as well as uncorrelated subspaces. This is illustrated in Figure 8 where we intentionally correlate the diffuse map and surface normals, and the specular albedo and roughness map, but neither group of property maps correlates to the other. We employ 500 samples to learn the distribution of each component, as well as 250 samples per rendering projection. The view and lighting configuration used for the rendering projections are the same as in the first experiment. As shown in Figure 8, both the correlated as well as uncorrelated relations of the property maps are accurately modeled by our multi-projection GAN.

Comparison between Different Projection Combinations In practice, however, obtaining direct observations of the different components might not be easy. This raises two questions: First, can we compensate a reduction in selection projection training samples with additional rendering projection training samples? In addition, is it possible to completely forgo selection projections by choosing proper rendering projections?

To answer the first question, we further reduce the number of training samples for each selection projection to 250, in which case our multi-projection GAN fails to produce reasonable results (Figure 9 top). Next, we keep the number of samples for the selection projections fixed at 250, and increase the number of samples for each rendering projection to $8,000$. As shown in Figure 9 (middle), while the 250 samples per selection projection are insufficient to even infer a valid distribution for each component, the additional $8,000$ samples per rendering projection allow us to successfully model the correlations, as well as successfully improve the distributions for each component.

To answer the second question, we further remove all the dependence on selection projections, and use four rendering projections that observe the sample frontally and that differ in the types of illuminations: we categorize each training sample based on whether the dominant illumination source is a directional light (from normal incidence), a directional light from a non-normal incidence direction, a natural environmental lighting condition, or a constant ambient lighting condition. We expect the rendering projection from the constant ambient lighting to produce strong cues for the diffuse albedo, the normal incidence directional light and the natural lighting to produce good cues on the specular albedo and roughness, and the random directional lighting to provide cues for the surface normals. With $6,000$ independently selected samples per projection we are able to successfully learn the distribution of wood SVBRDFs (Figure 9 bottom).

Additional Experiments In Figures 10 and 11 we demonstrate that our multi-projection SVBRDF GAN generalizes beyond wood SVBRDFs by learning the distribution of the surface appearance of metals and plastics. The metal and plastic datasets are described in subsection 2.5.

During training, for each rendering projection, we render the generated property maps using the view distribution gathered from the real training images and use a randomly selected natural environmental lighting condition.

3. Part III: Multi-projection GAN for MNIST dataset

To demonstrate that our automatic view-prediction multi-projection GAN is also suited for other types of datasets other than 3D shape modeling, we provide a "synthetic" example where we learn the distribution from the MNIST dataset (Figure 12). For this example, we use 10 projections that given a 2D direction, integrates the space along this 1D direction. We add a latent variable that adds a small (+/- 2.5 degrees) perturbation to the projection direction. We ensure that the projection has the same 'width' as the 2D image space, and further ensure that the center of the projection space is aligned with the center of the 2D image space. The training data was generated using the same projections including the perturbation. We show training results using MP-GAN (with known view distribution), as well as VP-MP-GAN where we also predict the projection directions.

References

[1] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, M. Kudlur, J. Levenberg, R. Monga, S. Moore, D. G. Murray, B. Steiner, P. Tucker, V. Vasudevan, P. Warden, M. Wicke, Y. Yu, and X. Zheng. Tensorflow: A system for large-scale machine learning. In *OSDI*, pages 265–283, 2016. 3

[2] S. Bell, P. Upchurch, N. Snavely, and K. Bala. OpenSurfaces: A richly annotated catalog of surface appearance. *ACM Trans. on Graph.*, 32(4), 2013. 3

[3] R. L. Cook and K. E. Torrance. A reflectance model for computer graphics. *ACM Trans. Graph.*, 1(1):7–24, 1982. 1

[4] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, pages 448–456, 2015. 2

[5] X. Li, Y. Dong, P. Peers, and X. Tong. Modeling surface appearance from a single photograph using self-augmented convolutional neural networks. *ACM Trans. Graph.*, 36(4):45:1–45:11, July 2017. 3

[6] A. L. Maas, A. Y. Hannun, and A. Y. Ng. Rectifier nonlinearities improve neural network acoustic models. In *ICML*, 2013. 2

[7] T. Miyato, T. Kataoka, M. Koyama, and Y. Yoshida. Spectral normalization for generative adversarial networks. In *ICLR*, 2018. 1

[8] A. Radford, L. Metz, and S. Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv*, 2015. 2

[9] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, X. Chen, and X. Chen. Improved techniques for training gans. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *NIPS*, pages 2234–2242. 2016. 2

[10] J. Shi, Y. Dong, H. Su, and S. X. Yu. Learning non-lambertian object intrinsics across shapenet categories. In *CVPR*, pages 5844–5853, 2017. 3

[11] J. Wu, C. Zhang, T. Xue, W. T. Freeman, and J. B. Tenenbaum. Learning a probabilistic latent space of object shapes via 3d generative-adversarial modeling. In *NIPS*, pages 82–90, 2016. 1

Figure 2. Additional results of *VP-MP-GAN* for the *chair* dataset, visualized from different views. All results are synthesized by a generator trained on silhouettes from real-world images.

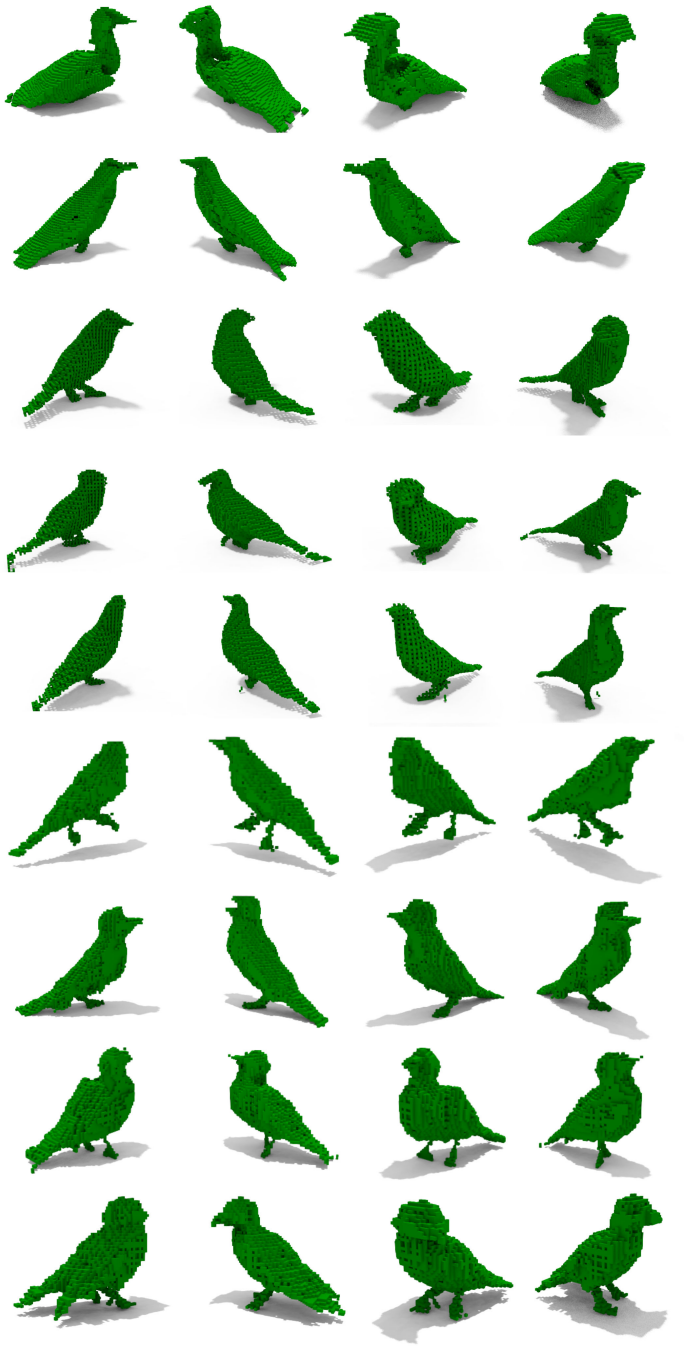

Figure 3. Additional results for *VP-MP-GAN* on the *bird* dataset, visualized from different views. All results are synthesized by a generator trained on silhouettes from real-world images.

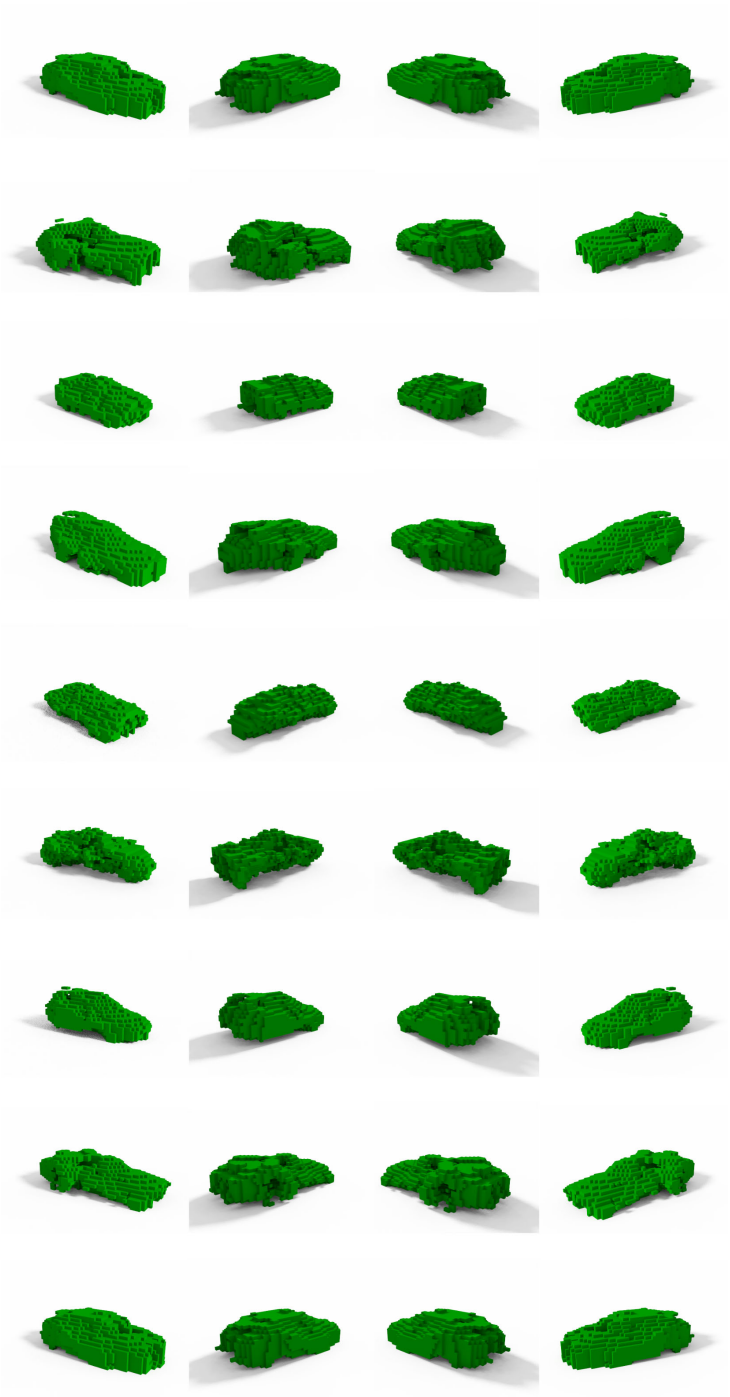

Figure 4. Additional results for *VP-MP-GAN* on the *car* dataset, visualized from different views. All results are synthesized by a generator trained on silhouettes from real-world images.

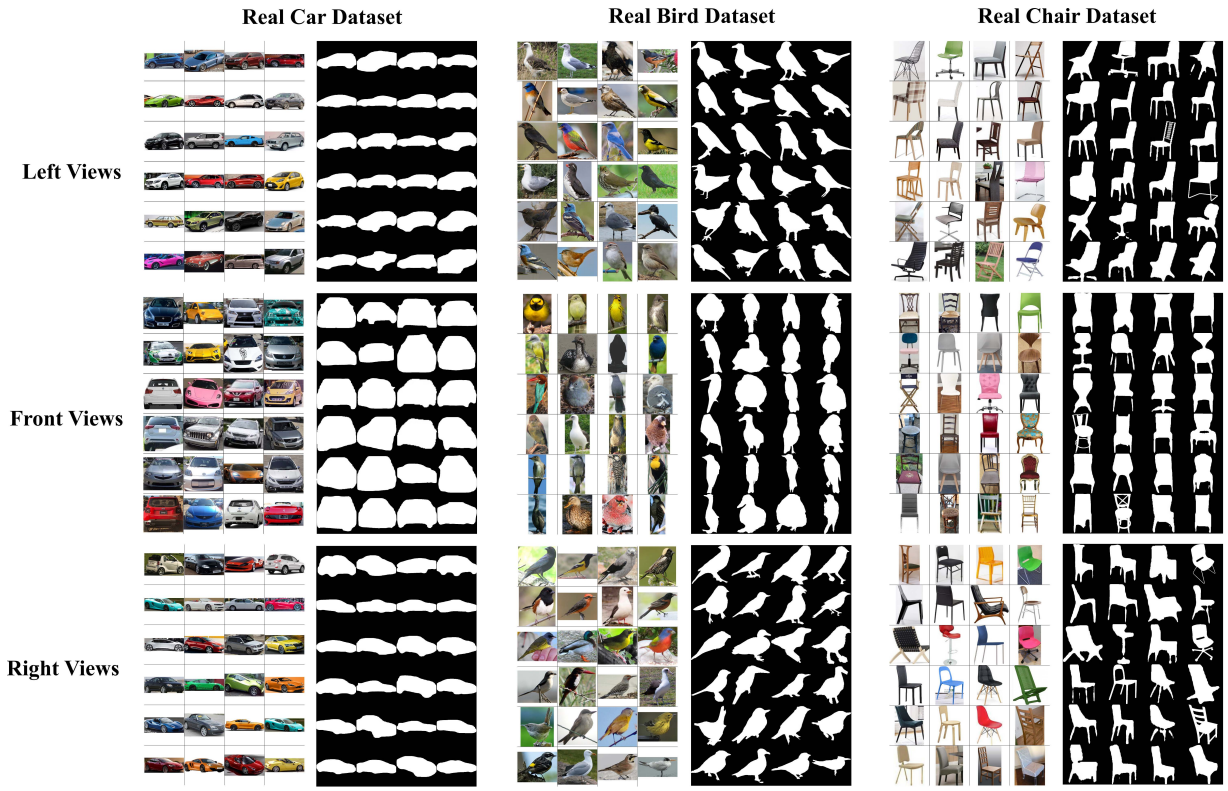

Figure 5. Unannotated training silhouettes and corresponding source photographs for the real-world *chair*, *bird*, and *car* datasets. Note that the corresponding views for each group are not provided as training labels; they are the result of the viewpoint prediction network jointly trained with *VP-MP-GAN*.

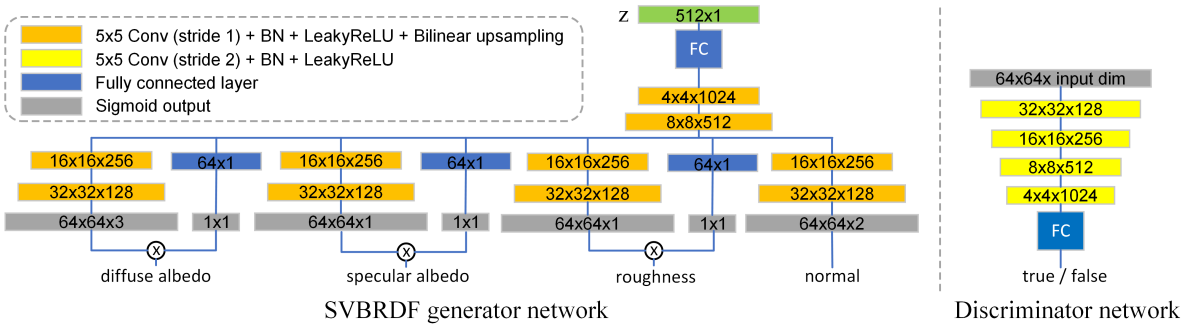

Figure 6. Summary of the multi-projection SVBRDF GAN structure.

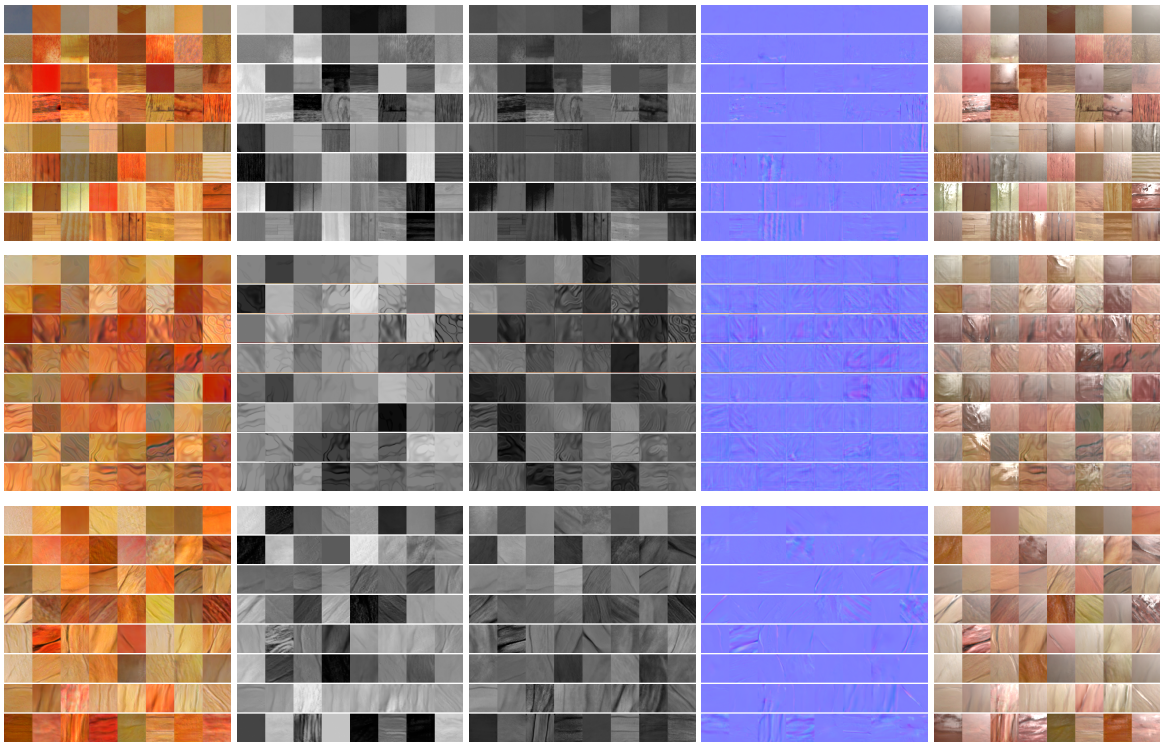

Figure 7. Selected SVBRDFs generated with the high-dimensional GAN (middle) and our multi-projection GAN (bottom) compared to reference training data (top).

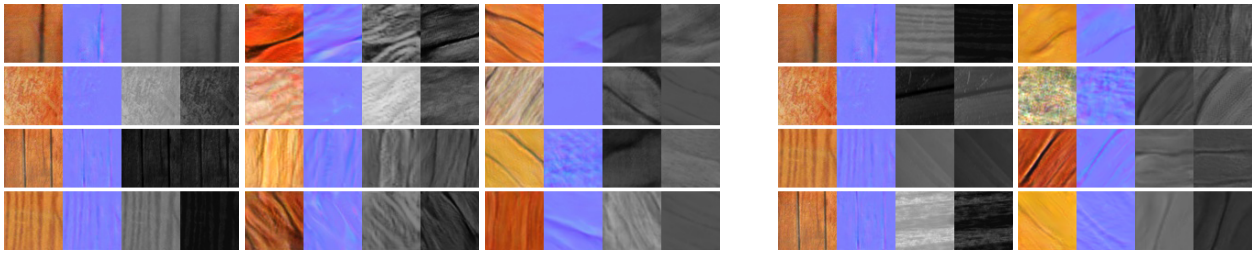

Figure 8. Left: The rendering projections enable our multi-projection GAN to automatically learn the correlations between the property maps. While training without the rendering projections still produces plausible property maps, it misses the correct interrelations. Right: An example of a (by design) partially correlated dataset, our multi-projection GAN is still able to infer the correct embedded (un)correlations from the data. For each example we show, left to right, diffuse albedo, normal map, specular albedo, and roughness.

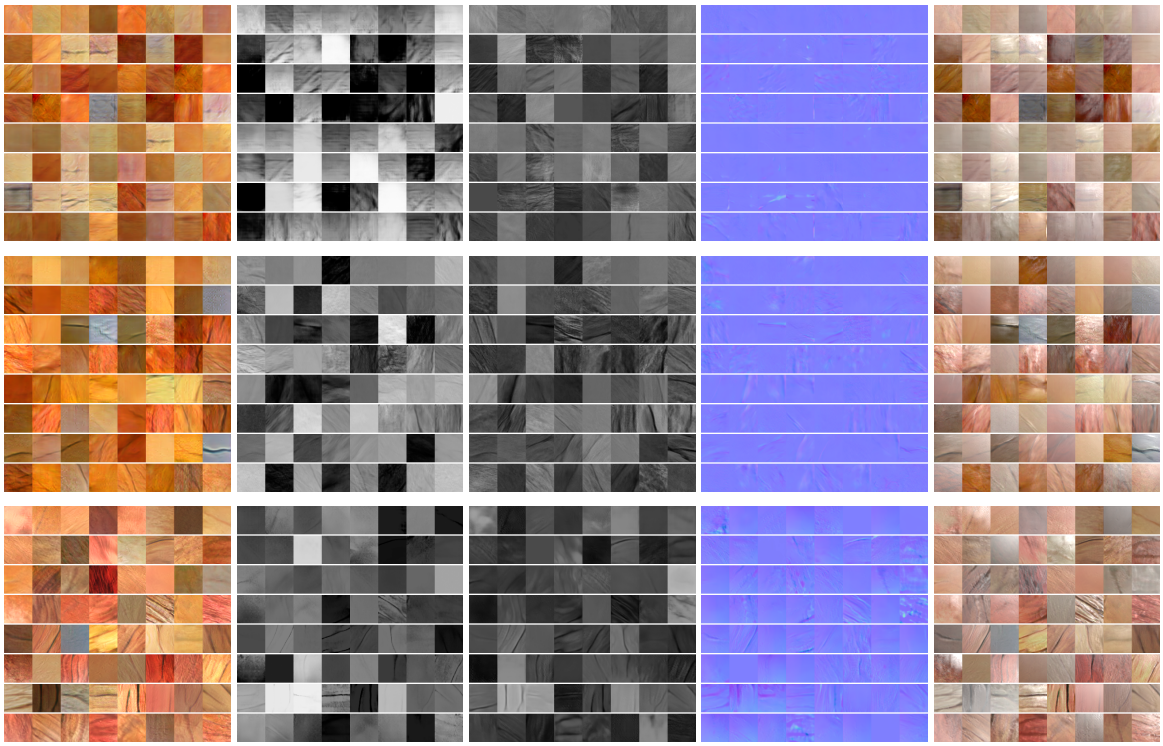

Figure 9. Top: Multi-projection GAN trained with an insufficient number of training samples (250 for each selection and rendering projection). Middle: Increasing the training samples to $8,000$ for the rendering projections only, correctly infers the correlation between the different property maps, as well as improve the fidelity of the distributions of each property map. Bottom: multi-projection GAN trained with only four rendering projections (and no selection projection), each based on a different class of illumination, can successfully learn the distribution of wood SVBRDFs.

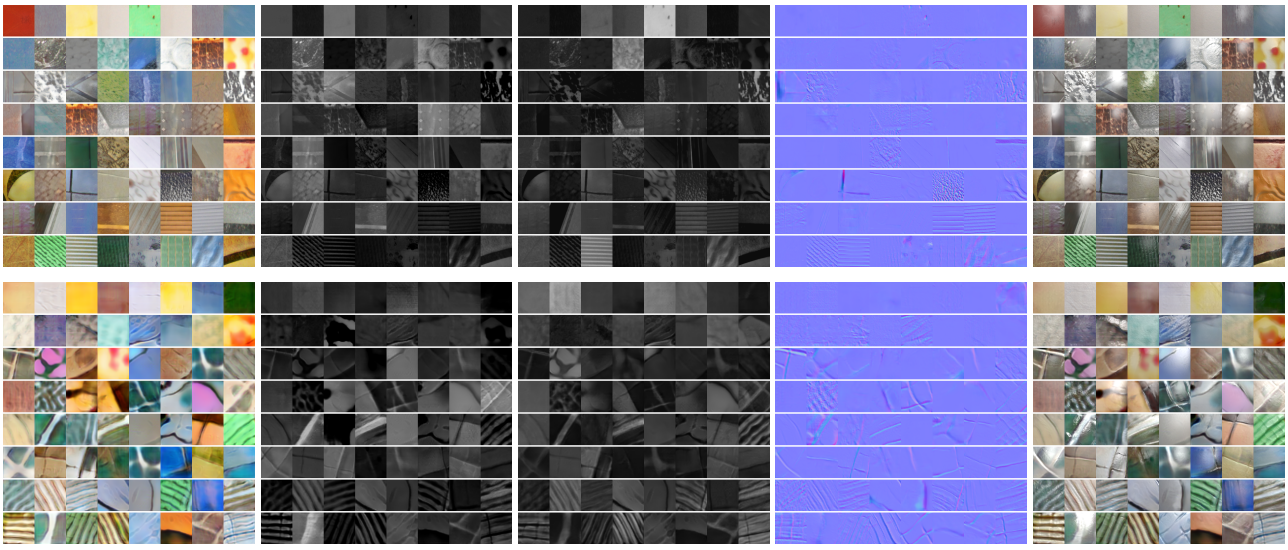

Figure 10. Plastic SVBRDF generator trained with multi-projection GAN (bottom), compared to reference training data (top).

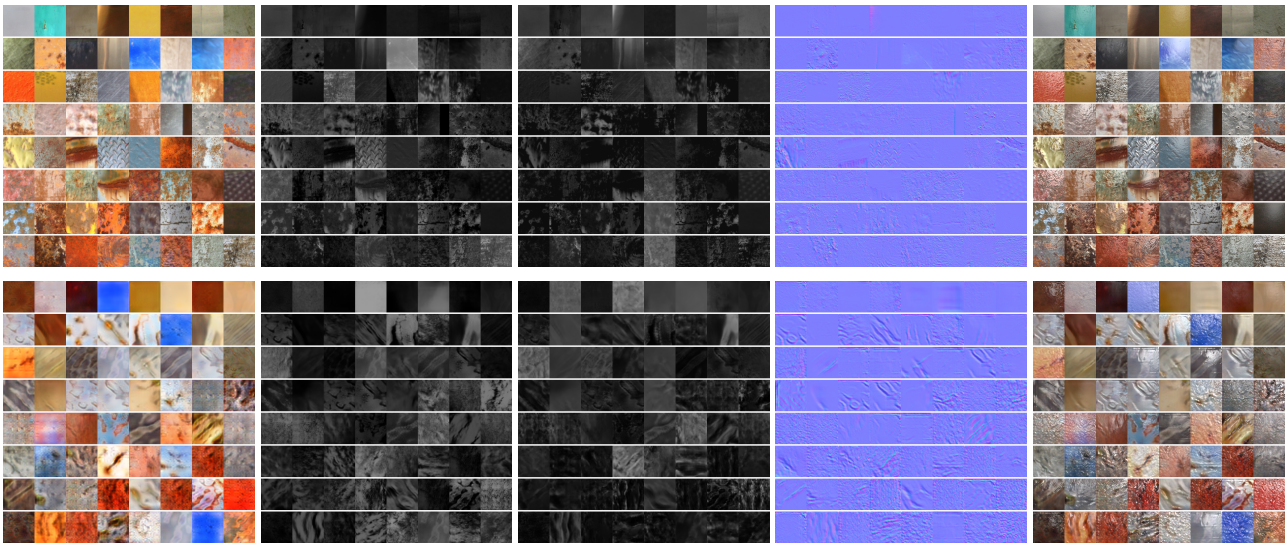

Figure 11. Metal SVBRDF generator trained with multi-projection GAN (bottom), compared to reference training data (top).

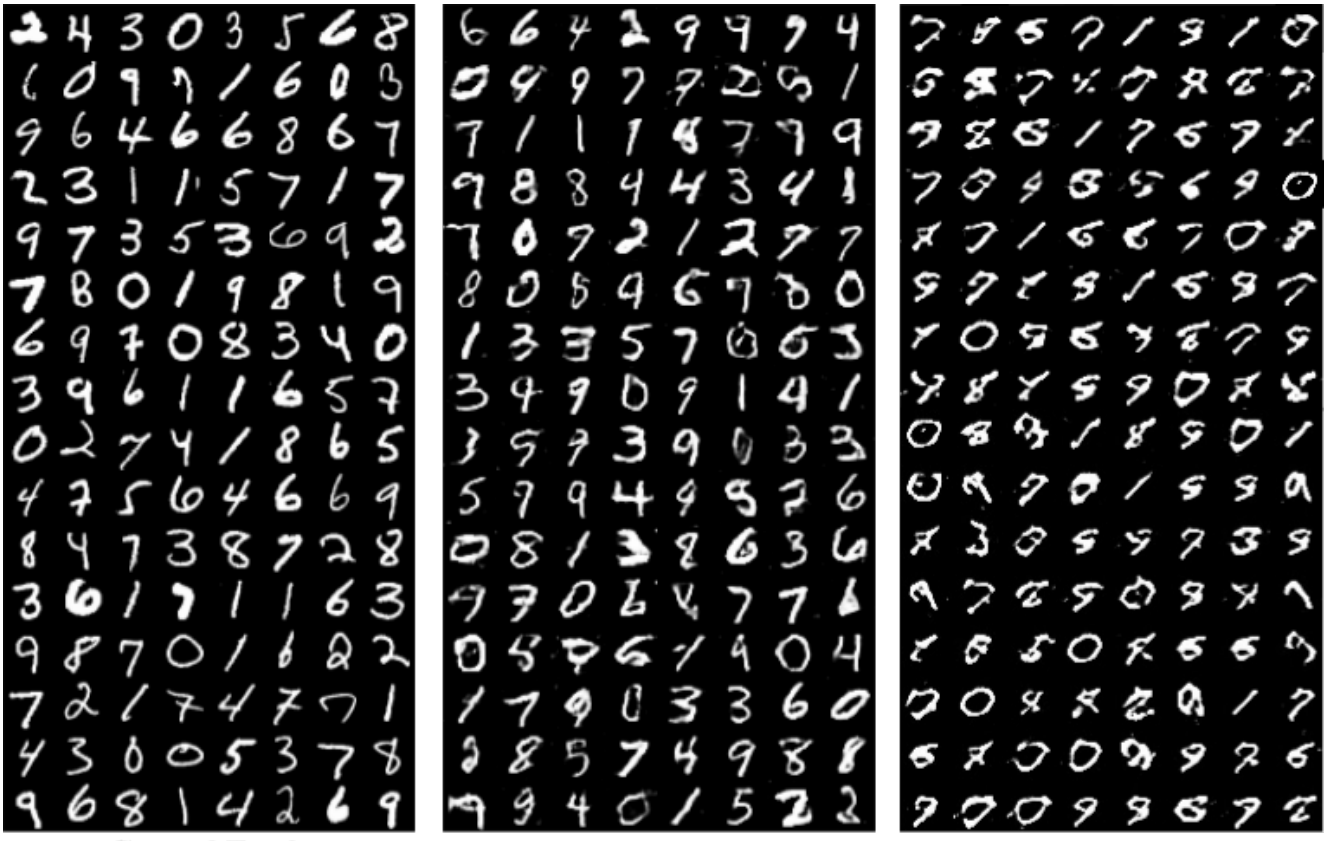

Figure 12. Synthetic experiment on the *MNIST* dataset. Left: reference images. Middle: Results obtained with *MP-GAN* trained from 10 pre-defined projections. Right: Results obtained from the same data with *VP-MP-GAN*. Note due to intrinsic ambiguities between the orientation of the content and the projection direction, *VP-MP-GAN* generates rotated numbers. Nevertheless, both *VP-MP-GAN* and *MP-GAN* are able to generate high quality results from multiple 1D projections.