Supplementary Material for
Texture Fields: Learning Texture Representations in Function Space

Michael Oechsle[1,2] Lars Mescheder[1] Michael Niemeyer[1] Thilo Strauss[2†] Andreas Geiger[1]
[1]Autonomous Vision Group, MPI for Intelligent Systems and University of Tübingen
[2]ETAS GmbH, Bosch Group, Stuttgart
[1]{firstname.lastname}@tue.mpg.de [†]{firstname.lastname}@etas.com

Abstract

In the following, we present the network architectures for our model and provide more information on training and data pre-processing. Moreover, we provide additional results for each experiment in the main paper. In the third part, we visualize Texture Fields in order to obtain more insights into what they are actually learning.

Supplementary Material

1. Implementation Details

In this section, we provide more information about the network architectures used in the experiments. Furthermore, we explain the architecture of the novel view synthesis baseline and provide more information about the pipeline for image based training.

1.1. Architectures

Here, we describe the architectures of each part of our model, shown in Figure 2 of the main paper.

Texture Field: In Fig. 1, the network architecture of the Texture Field is shown. We adapt the architecture proposed in [4] for the task of texture prediction. This architecture is used for all of our experiments. The inputs to a Texture Field are a 3D position $\mathbf{p}$, shape embedding $\mathbf{s}$ and a condition or latent code $\mathbf{z}$. The shape embedding provides information about the global shape to the network, whereas $\mathbf{z}$ is used for the image condition. Fig. 1 shows the architecture applied for a set of N 3D locations of a single 3D model. All points are passed through a fully-connected neural network that outputs a feature vector for each point. The next parts of our architecture consists of ResNet blocks with fully-connected layers [2]. In each block we first inject features of $\mathbf{s}$ and $\mathbf{z}$ by concatenating them, passing them through a fully-connected network and adding the output to the features of each point. We apply $L = 6$ ResNet blocks for the single image texture reconstruction experiments and $L = 4$ ResNet blocks for the generative models. Finally, a fully-connected layer maps the 128-dimensional feature vector to the image space.

Shape Encoder: For the pointcloud, we sample 2048 points uniformly on the surface of the 3D models. In order to derive an embedding from the pointcloud, we utilize the pointcloud encoder, proposed in [4]. The architecture is depicted in Fig. 2. Based on PointNet [5], the network consists of 5 Resnet blocks with max pooling layers.

Image Encoder: As encoder for the input image, we use a pretrained ResNet-18 architecture [2], visualized in Fig. 3. After the last ResNet-Block we apply a average pooling layer and a fully-connected layer for deriving the image embedding $\mathbf{z}$.

VAE Encoder: For training the VAE, we encode the ground truth views using the ResNet-based network in Fig. 4. The encoder receives an image as well as the shape embedding as input and predicts mean μ and log-standard deviation $\log \sigma$ of normally distributed random variables in the latent space. For injecting the shape embedding, we pass $\mathbf{s}$ through a fully connected network with 32 as output dimension and add the output to each feature pixel. Then, we iteratively apply average

pooling and a ResNet block for 5 times. Finally, we use two separate fully-connected networks to map the features to mean and log-standard deviation of the latent code $\mathbf{z}$.

GAN Discriminator: We apply the network shown in Fig. 5 as discriminator in the conditional GAN set up. We condition the discriminator on the depth image by concatenating the depth and RGB image. Using a similar architecture as in [3], the input is mapped to a single scalar.

1.2. NVS Baseline

In Fig. 6, we show the architecture of the novel view synthesis baseline (NVS). The networks predict a RGB image given a depth image as input. We apply a U-Net-based architecture [6] and inject the image encoding into each layer.

1.3. Data preparation

We train Texture Fields from a dataset consisting of rendered images and corresponding depth images as well as intrinsic and extrinsic camera information. To this end, we render images from 10 random views in the upper hemisphere for the 3D objects of the used ShapeNet categories. For lighting we use a hemispheric light source. Additionally, we render depth images from the same random views and store camera intrinsics and extrinsics, in order to be able to reproject each pixel in the depth image back to its 3D location. In the end, the data consists of 10 views for each of the 7,499 car models, 6,781 chair models, 4,048 airplane models, 8,512 table models, 1,816 bench models, 1,572 cabinet models, 2,318 lamp models, 3,173 sofa models and 1,939 vessel models. For training our method we use a resolution of 128^2. For the NVS baseline, we use 256^2. We evaluate every method at a resolution of 256^2 from 10 random views. As input images for the conditional experiments, we use the renderings from Choy et al. [1].

2. Further Results

We present additional results for each of the experiments in the following Figures 7, 8, 9, 10, 11, 12, 13, 14 and 15. Category specific results for the full pipeline for single image 3D reconstruction is presented in Table 1. Furthermore, in Fig. 16, we show predictions of our model using input images that do not correspond to the input shape.

While our method is properly reconstructing global pattern of appearances, high frequency details are partly missing. The lack of detail can be attributed to the global image encoder which transforms the input image into a compact 512-dimensional latent code. In addition, we believe that the fully connected ResNet architecture employed by our model contributes to this issue. In future work, we plan to investigate how local appearance information can be better propagated to preserve high-frequency details.

3. Field Visualizations

In this section, we investigate what Texture Fields are actually learning. For this purpose, we use the single image texture reconstruction task and we vary the input shape, while we keep the same image condition.

In Fig. 17, we depict color values predicted by a Texture Field along cuts through car models. We see that the Texture Field learns to predict color values at the location of the shape. As expected, color predictions far from the shape are meaningless as non of the observations constrain these areas. In the interior of the car, a gray color usually appears, whereas outside of the car it is white. By varying the input shape, we observe that the network is changing the locations of the color according to the shape. The Texture Field successfully transfers texture information from the image condition onto arbitrary shapes. This leads us to the conclusion that Texture Fields implicitly decode the shape embedding and reconstruct the texture at encoded shape locations following the image condition.

	FID				SSIM				Feature-ℓ_1			
	Projection	Im2Avatar	NVS	Texture Field	Projection	Im2Avatar	NVS	Texture Field	Projection	Im2Avatar	NVS	Texture Field
airplanes	79.146	-	70.592	**61.760**	0.918	-	**0.921**	**0.921**	0.230	-	0.223	**0.216**
benches	**55.828**	-	80.231	72.285	0.822	-	0.827	**0.833**	**0.225**	-	0.236	0.227
cabinets	**41.081**	-	69.953	47.673	0.839	-	**0.885**	0.881	0.190	-	0.193	**0.180**
cars	133.411	149.393	122.622	**77.439**	0.786	0.760	0.836	**0.837**	0.281	0.290	0.269	**0.235**
chairs	37.890	158.243	48.926	**36.812**	0.817	0.695	0.841	**0.842**	0.213	0.289	0.218	**0.207**
lamps	69.548	-	70.884	**66.939**	0.890	-	**0.902**	**0.902**	0.253	-	0.250	**0.247**
sofas	42.508	-	53.645	**42.216**	0.839	-	**0.866**	0.864	0.212	-	0.209	**0.197**
tables	32.693	115.992	35.086	**30.627**	0.855	0.749	**0.871**	0.869	0.193	0.265	0.188	**0.186**
vessels	99.600	-	107.069	**99.063**	0.880	-	0.882	**0.883**	0.262	-	0.265	**0.256**
mean	65.745	141.209	73.223	**59.424**	0.850	0.734	**0.870**	**0.870**	0.229	0.281	0.228	**0.217**

Table 1: **Full pipeline.** Quantitative results of the full pipeline and the corresponding baselines for all categories.

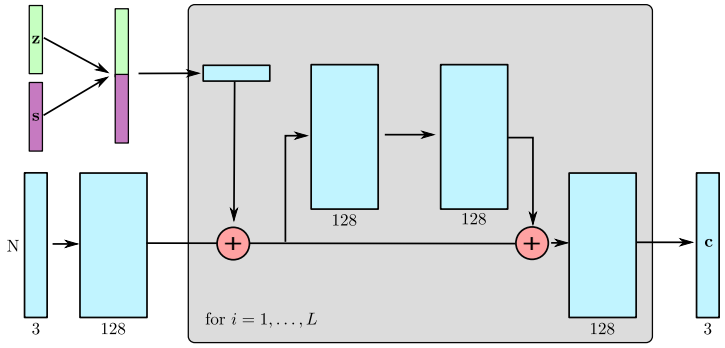


Figure 1: **Texture Field.** This figure illustrates the architecture of Texture Fields. Shape Embedding $\mathbf{s}$ and condition/latent texture code $\mathbf{z}$ are injected to each ResNet block. For each of the N 3D points, the network outputs a 3-dimensional color value $\mathbf{c}$.

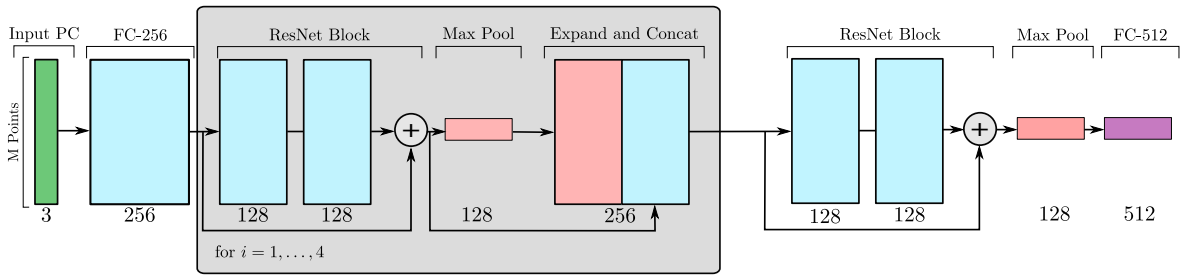


Figure 2: **Shape Encoder.** Similar to a PointNet encoder [5] , the network shown here determines a feature vector from a set of points. We use the ResNet-based version, proposed in [4].

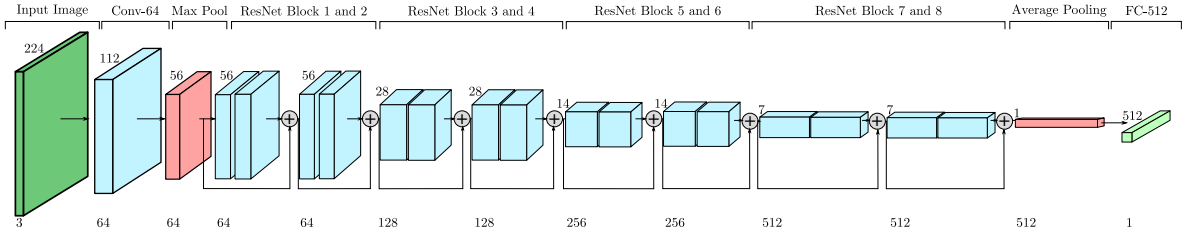


Figure 3: **Image Encoder.** As encoder of the input images for the conditional task, we apply the ResNet-18 network pretrained on ImageNet.

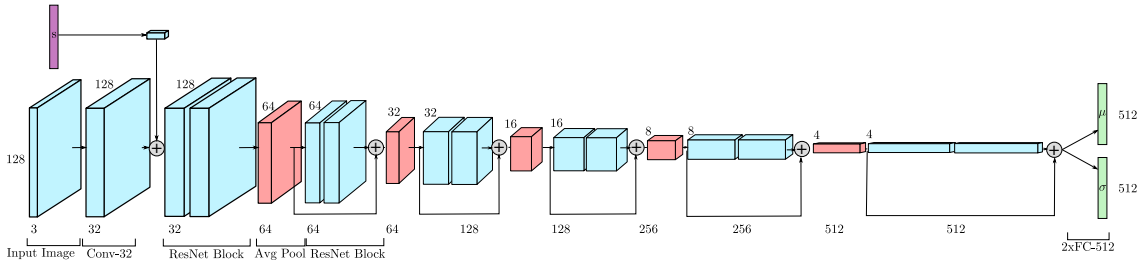


Figure 4: **VAE Encoder.** Here, we illustrate the network of the encoding part of the VAE. The encoder maps an image and the shape embedding $\mathbf{s}$ to mean μ and log-standard deviation $\log \sigma$ of the latent variable $\mathbf{z}$.

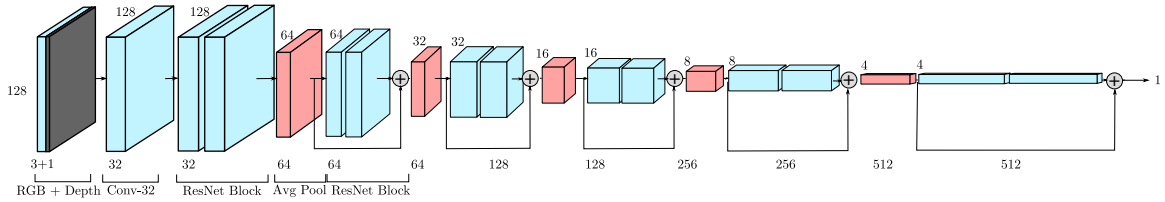


Figure 5: **GAN Discriminator.** As input for the discriminator we use a RGB image and a corresponding depth image. By using average pooling and ResNet blocks, the input is mapped to a single scalar value.

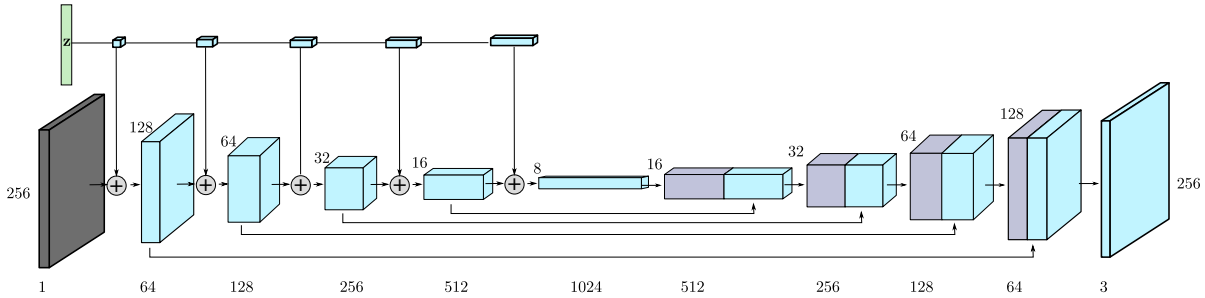


Figure 6: **NVS.** In this figure, we show the U-Net-based architecture of the NVS baseline. We inject the image embedding $\mathbf{z}$ into each layer in the encoding part of the network.

Figure 7: **Overfitting to 10 Human Body Models.** By assigning an individual embedding to each of the model, we train the network to predict texture given the embedding of the models. Qualitatively, we see that Texture Fields can represent high frequency textures also for multiple 3D models. Remark: the last example contains large dark body tattoos.

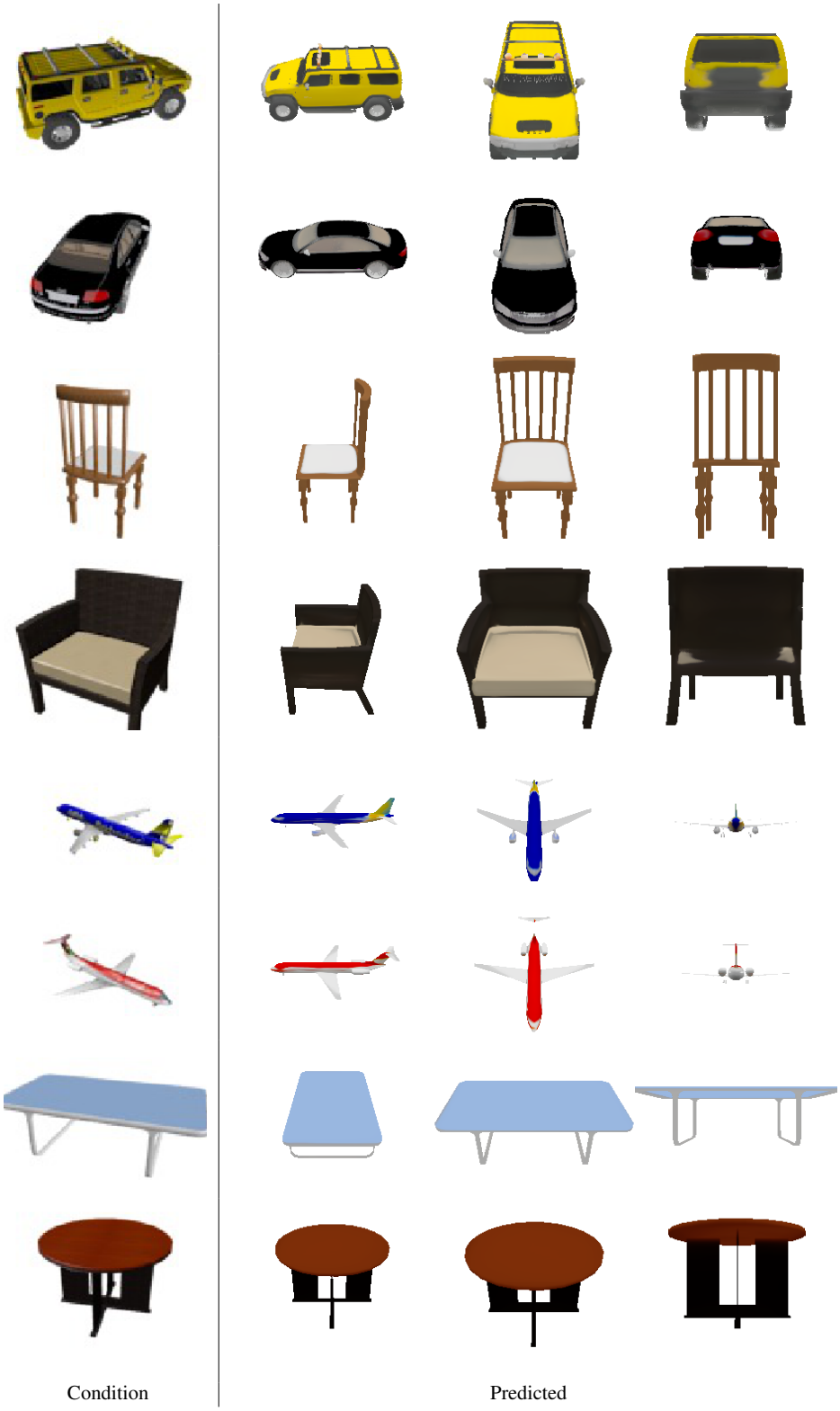


Figure 8: **Texture Reconstruction with Texture Field.** In this Figure, we use our model to predict texture for untextured 3D CAD models based on a single view of the same objects. The texture is properly predicted for all categories and contains details such as lights and number plates. Very high-frequency details are sometimes leading to blurriness.

Figure 9: **Texture Reconstruction with Texture Field.** We observe that learning texture reconstruction of categories with more diverse shapes, appearances and less samples, e.g. cabinets or lamps, is more difficult. Predictions of these categories suffer more often from blurry and inconsistent areas.

Figure 10: **Texture Reconstruction with Real Images.** In this figure, we show results for texturing untextured synthetic CAD models from a single real input image using our approach.

Figure 11: **Full Pipeline.** Our full pipeline for texture and shape reconstruction leads to plausible textured 3D objects, as shown in this figure. We use the same shape reconstruction model (ONet) [4] for all approaches.

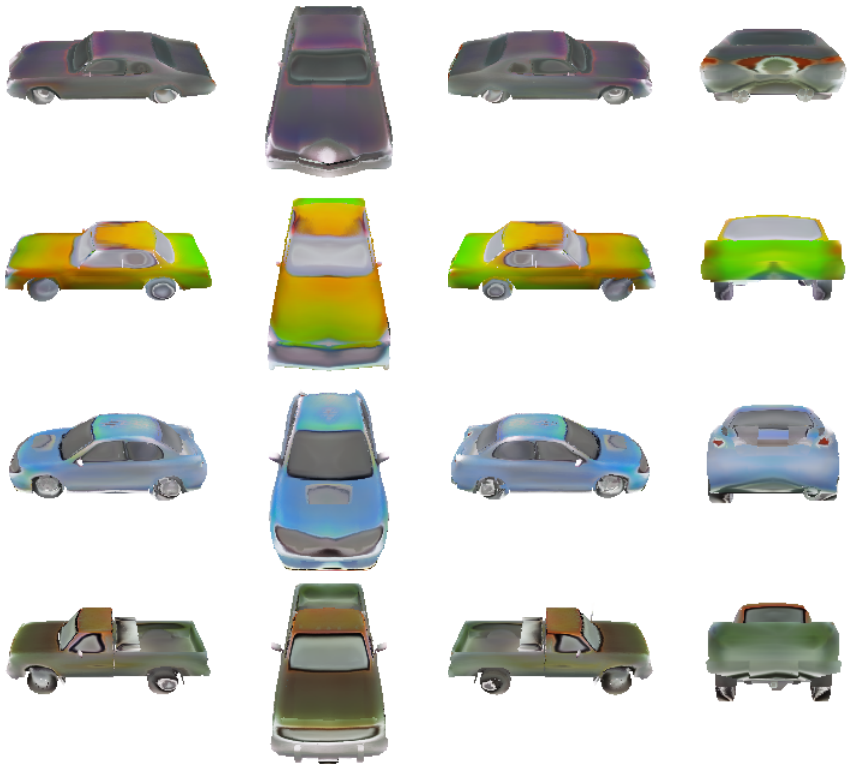


Figure 12: **GAN.** In this figure, we show 3D CAD objects with generated texture using our GAN-based model. Our GAN outputs exhibit GAN typical artifacts.

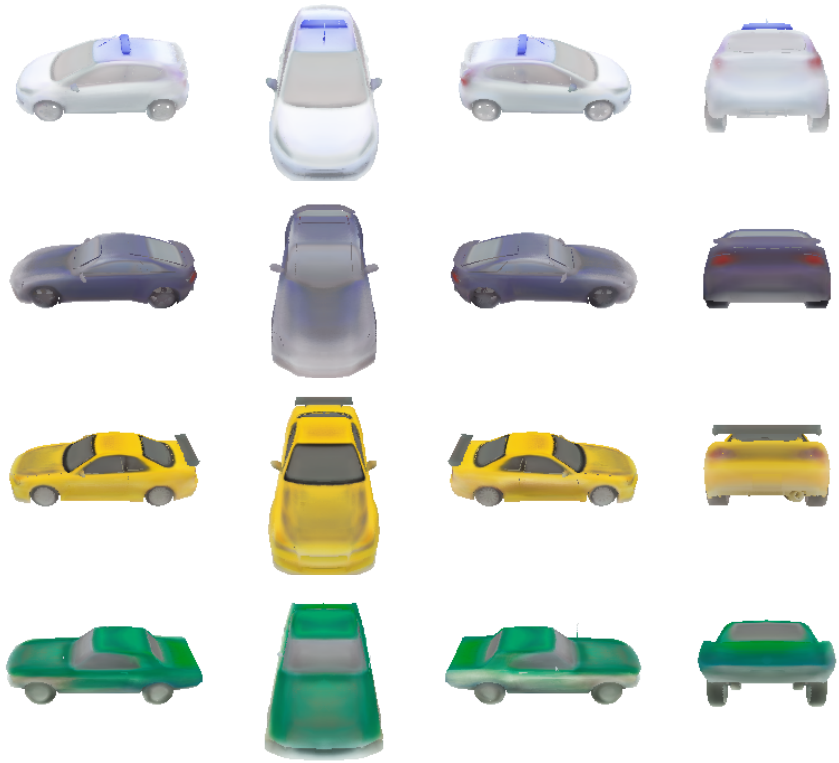


Figure 13: **VAE.** This figure illustrates predicted textures using the VAE model. The results are globally consistent, but exhibits blur in some cases.

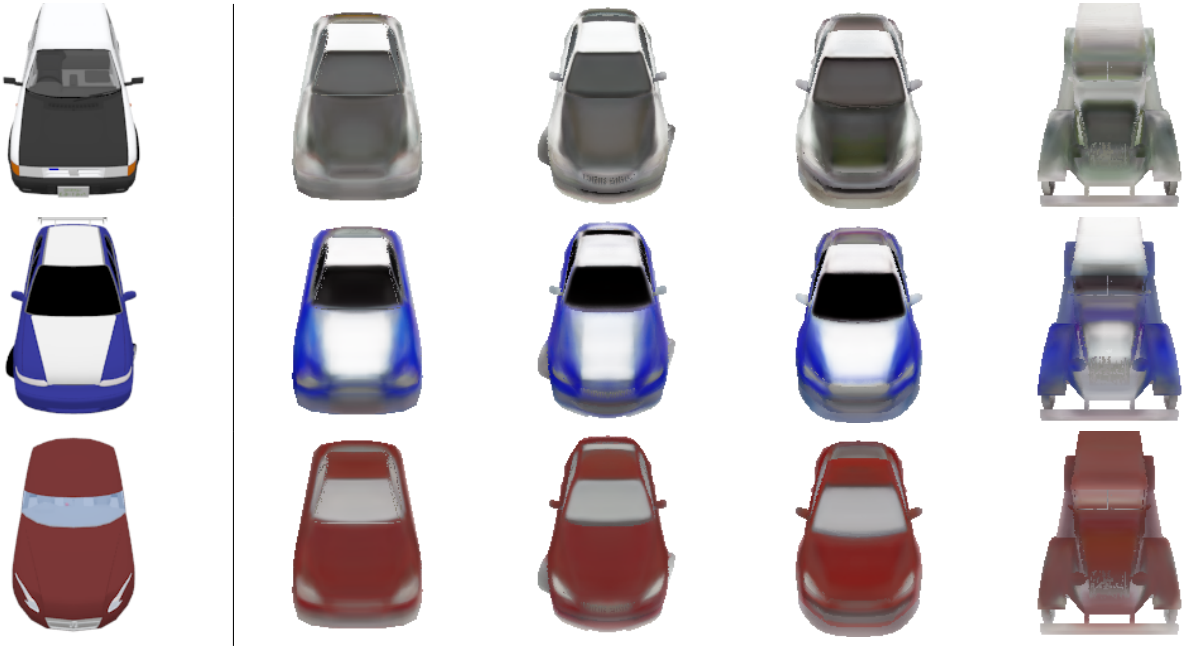


Figure 14: **Texture Transfer.** We utilize the VAE model for texture transfer from one car to another. We encode the image on the left and use the latent code for synthesizing texture for different shapes.

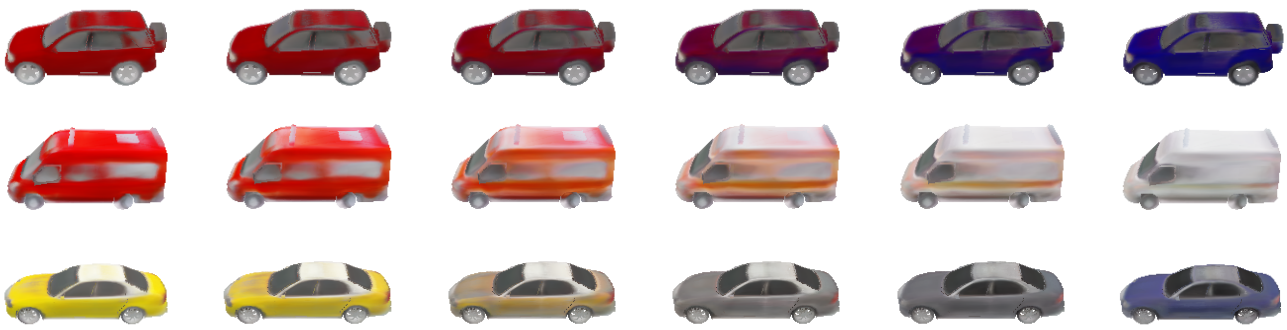
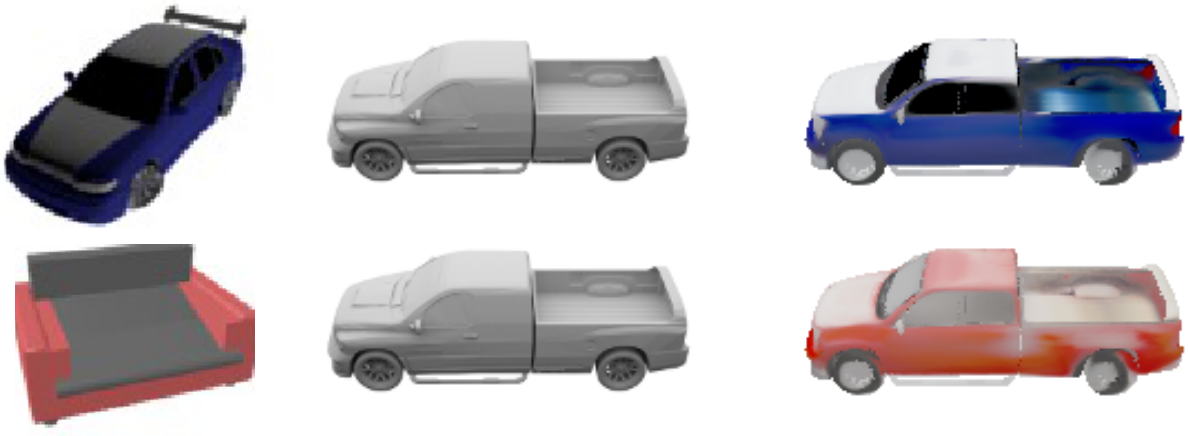


Figure 15: **Latent Space Interpolations.** Here, we illustrate latent space interpolations. The results show that our model learns a continuous and meaningful latent space.



Input image Input shape Prediction

Figure 16: **Single Image Texture Transfer.** In this figure, we show predictions of our model when presenting a novel input image that does not correspond to the input shape (first row) or even to the object category the model has been trained on (second row). We observe that our model is able to plausibly transfer appearance to novel 3D geometries. This indicates that shape and texture are disentangled to some degree by our model.

Figure 17: **Texture Field Illustrations.** In this figure, we show predicted color values along cuts through the car models. The image condition is shown on the left and on the right two different results for cuts are depicted. In the top row a cut through the middle of the cars is shown, whereas in the bottom row a cut on the right side of the car. Furthermore, we show the results for two different input shapes, the corresponding 3D model and car with a completely different shape. We observe that the Texture Field learns to predict color values at locations close to the input 3D model. Far from the shape, the color values are meaningless.

References

[1] Christopher Bongsoo Choy, Danfei Xu, JunYoung Gwak, Kevin Chen, and Silvio Savarese. 3d-r2n2: A unified approach for single and multi-view 3d object reconstruction. In *Proc. of the European Conf. on Computer Vision (ECCV)*, 2016. 2

[2] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2016. 1

[3] Lars Mescheder, Andreas Geiger, and Sebastian Nowozin. Which training methods for gans do actually converge? In *Proc. of the International Conf. on Machine learning (ICML)*, 2018. 2

[4] Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy networks: Learning 3d reconstruction in function space. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019. 1, 3, 9

[5] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2017. 1, 3

[6] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, 2015. 2