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 recent years, substantial progress has been achieved in learning-based reconstruction of 3D objects. At the same time, generative models were proposed that can generate highly realistic images. However, despite this success in these closely related tasks, texture reconstruction of 3D objects has received little attention from the research community and state-of-the-art methods are either limited to comparably low resolution or constrained experimental setups. A major reason for these limitations is that common representations of texture are inefficient or hard to interface for modern deep learning techniques. In this paper, we propose Texture Fields, a novel texture representation which is based on regressing a continuous 3D function parameterized with a neural network. Our approach circumvents limiting factors like shape discretization and parameterization, as the proposed texture representation is independent of the shape representation of the 3D object. We show that Texture Fields are able to represent high frequency texture and naturally blend with modern deep learning techniques. Experimentally, we find that Texture Fields compare favorably to state-of-the-art methods for conditional texture reconstruction of 3D objects and enable learning of probabilistic generative models for texturing unseen 3D models. We believe that Texture Fields will become an important building block for the next generation of generative 3D models.

1. Introduction

3D reconstruction is one of the grand goals of computer vision. Recently, the vision community has witnessed impressive progress in reconstruction tasks like single image 3D reconstruction [6, 7, 11, 21, 38] and generating 3D objects [4, 28, 40] using learning-based techniques which resolve ambiguities by incorporating prior knowledge. However, previous work on learning-based 3D reconstruction has mainly focused on the problem of reconstructing geometry. In contrast, texture reconstruction of 3D objects has received less attention from the research community.

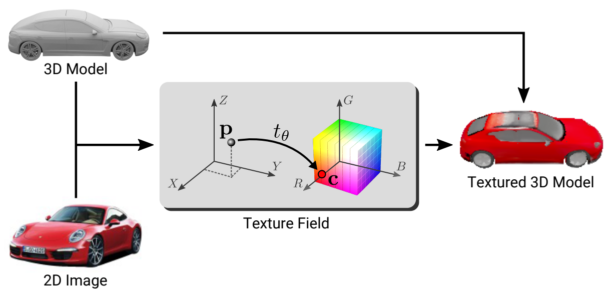

Figure 1: **Texture Fields** take a 3D shape and (optionally) a single 2D image of an object as input and learn a continuous function t_θ which maps any 3D point $\mathbf{p}$ to a color value $\mathbf{c}$, enabling the prediction of accurately textured 3D models.

Previous approaches for texture reconstruction from single images differ in how texture information is represented. Several recent works [34, 37] have proposed volumetric voxel representations for colored 3D reconstruction. Unfortunately, however, due to the computational cost of voxel representations, which grows cubically with the resolution, the voxelizations are limited to rather low resolutions (usually 32^3 or 64^3) and hence cannot represent high frequency details. An alternative representation of texture consists of a 2D texture atlas and a parameterized mesh using a UV-mapping that maps a point on the shape manifold to a pixel in the texture atlas. However, current approaches based on mesh representations usually assume known topology and a predefined template mesh which limits these approaches to specific object categories such as birds [15] or faces [31]. Predicting texture in the general case for arbitrary shapes without template mesh remains an unsolved problem.

Contributions: The main limiting factor of the aforementioned methods is their texture representation which either does not allow for high resolution reconstruction or strongly relies on task-specific shape parameterizations, limiting generality of these methods. An ideal texture representation, in contrast, should be independent of the shape representations (voxels, point cloud, mesh, etc.) and able to represent high frequency detail. Towards this goal, we propose Texture Fields as a novel representation of texture. Our key idea is to learn a continuous function for representing texture information in 3D space, see Fig. 1. By parameterizing this function through a deep neural network we are able to integrate this representation into a deep learning pipeline for 3D texture reconstruction that can be trained end-to-end.

Our experiments on various 3D object categories demonstrate that Texture Fields are able to predict high frequency texture information from a single image. We combine our approach with a state-of-the-art 3D reconstruction method [21], yielding a method which jointly reconstructs both the 3D shape and the texture of an object. Besides conditional experiments, we also extend our novel texture representation to the generative setting and show preliminary results for texture synthesis given a 3D shape model and a latent texture code. We conduct experiments on texture transfer between different objects as well as interpolations in the latent texture space which demonstrate that our generative model learns a useful representation of texture.

2. Related Work

We now briefly discuss the most related works on single image 3D reconstruction, generative image modeling using 3D knowledge and continuous representation learning.

2.1. Single Image Reconstruction

In recent years, 3D reconstruction of shapes from single images has made great progress [6, 7, 11, 21, 38].

Voxels: Tulsiani et al. [37] proposed a voxel-based texture representations for learning colored 3D reconstruction based on ray consistency and multi-view supervision. More recently, Sun et al. [34] combined 2D-to-3D appearance flow estimation and voxel color regression for learning to reconstruct colored voxelizations in a supervised fashion. Unfortunately, however, voxel-based representations are severely limited in resolution due to computational and memory constraints. In contrast, our continuous approach does not require discretization and thus results in more details compared to voxel-based representations as we show in our experimental evaluation in Section 4.

Point Clouds: In recent years, novel view synthesis [27, 41] has been used for reconstructing colored point clouds from single images [23, 35]. By combining novel view synthesis and depth estimation, the method proposed in [35] reprojects the predicted image into a colored point cloud. In [23], point clouds are reconstructed from a set of novel views using multi-view stereo algorithms. Unfortunately, point-cloud based representations are sparse. While dense mesh representation can be extracted from point clouds, the reconstructed shapes typically do not match the quality of state-of-the-art 3D reconstruction approaches and inferring the missing texture information requires additional post-processing steps. In contrary, our approach allows for inferring appearance for any location in 3D space and can be used in combination with arbitrary shape representations.

Meshes: Mesh-based approaches rely on category-specific template models and rigid texture parameterizations [15, 31]. In contrast, our reconstruction approach can represent texture for arbitrary meshes without requiring a UV texture map of a category-specific template model. Note that determining a proper UV-mapping for arbitrary mesh models is a non-trivial problem which is typically solved with various heuristics for atlas generation. The advantage of our approach is that we circumvent mesh parameterization by disentangling the texture from the shape representation.

2.2. Generative Models

Recent work has shown that image generation methods [3, 9, 16, 17, 20, 26] can be improved by exploiting 3D knowledge about the generated shapes [1, 2, 42]. Alhaija et al. [2] propose a model which learns to translate intrinsic properties (depth, normals, material) into RGB images. Zhu et al. [42] learn to predict 3D geometry as well as texture information in 2D images by disentangling shape, texture and pose in an unsupervised setting. In contrast to those image-based approaches, we directly predict the texture for the entire object in 3D.

2.3. Continuous Representations

Recently, parameterized continuous functions gained popularity for 3D shape reconstruction. Several works [6, 21, 24] proposed to formulate 3D reconstruction as learning a continuous occupancy function or a signed-distance field in 3D space, parameterized via a neural network. Furthermore, in [8], a continuous function for color values was used for generating 2D images. In this work, we extend the concept of learning a representation in function space to reconstruct and generate texture information of 3D objects.

3. Method

This section describes the proposed Texture Field representation and demonstrates how it can be applied to conditional and unconditional texture synthesis tasks.

3.1. Texture Fields

Recent approaches to 3D reconstruction [6, 21, 24] represent 3D shapes as continuous functions of occupancy probability or surface distance. In contrast to point-, voxel- or mesh-based representations, these approaches do not rely on a fixed discretization and thus form an ideal basis for representing appearance information. We explore this idea by embedding surface texture as a continuous function in 3D space. In combination with state-of-the-art 3D recon-

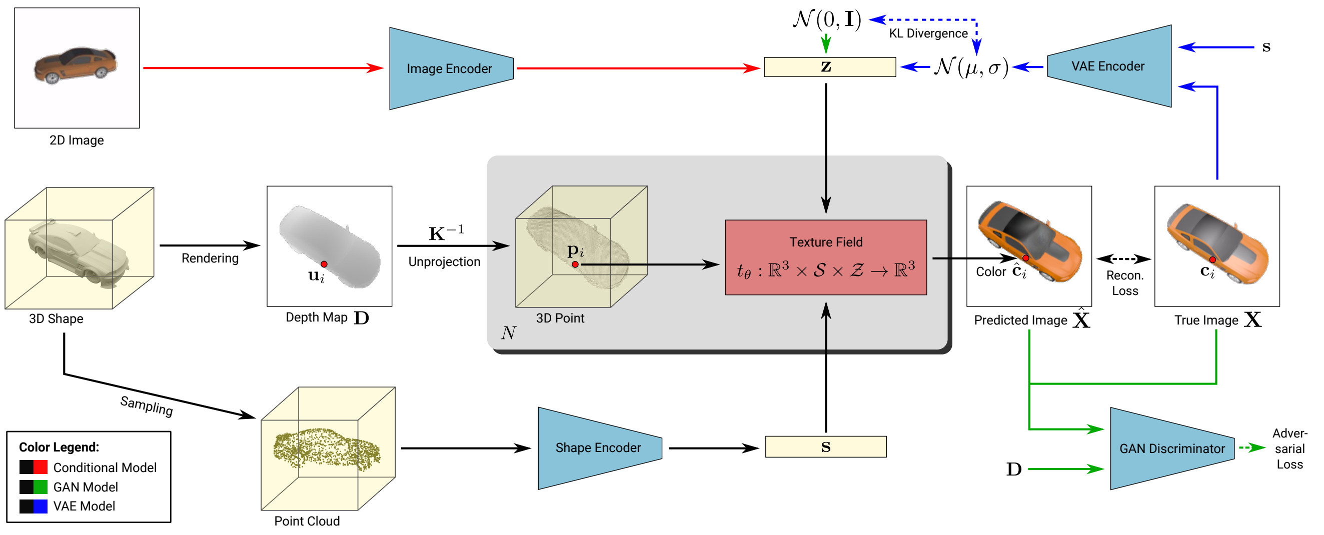

Figure 2: **Model Overview.** Colored arrows show *alternative* pathways representing the Conditional, GAN and VAE model. The blue and red boxes denote trainable components of our model which are parameterized through neural networks, respectively. The 3D shape of the object is encoded into a fixed-length vector representation $\mathbf{s}$. Additionally, we render a depth map $\mathbf{D}$ from a randomly chosen viewpoint and obtain the corresponding 3D points $\mathbf{p}_i$ by unprojecting all N image pixels $\mathbf{u}_i$ into 3D. The reconstruction loss minimizes the difference between the pixel colors predicted by the Texture Field $\hat{\mathbf{c}}_i = t_\theta(\mathbf{p}_i, \mathbf{s}, \mathbf{z})$ and the ground truth pixel colors $\mathbf{c}_i$. For training a Conditional model, the latent variable $\mathbf{z}$ encodes information about the input image. In the unconditional case (i.e., for the GAN and VAE), $\mathbf{z}$ is sampled from a Gaussian distribution.

struction techniques this allows us to reconstruct a textured 3D model from a single 2D image.

Let $t(\cdot)$ denote a function which maps a 3D point $\mathbf{p} \in \mathbb{R}^3$ to a color value $\mathbf{c} \in \mathbb{R}^3$ hence representing texture information by means of a 3D vector field:

$$t : \mathbb{R}^3 \to \mathbb{R}^3 \tag{1}$$

By parameterizing the function t as a neural network t_θ with learnable parameters θ, we reduce the problem to a simple regression task. However, the problem of texture generation remains ill-posed without any further constraints. Thus, in order to inform t_θ about the input shape of the object, we condition t_θ on a shape embedding $\mathbf{s} \in \mathcal{S}$. This helps the Texture Field to guide its predictions towards the object surface and allows for exploiting contextual geometric information such as surface discontinuities which are often aligned with image edges. As input shape, we explore 3D CAD models as well as image-based 3D reconstructions using neural networks [21] in this paper.

Unfortunately, even given the input 3D shape, there still exists a variety of plausible texture explanations. Consider cars, for instance, where the 3D geometry alone does neither determine the color, nor the exact shape of the windows or headlights. However, we may further constrain the task by providing information about the object appearance.

More specifically, we condition t_θ on an additional 2D image taken from an arbitrary viewpoint. Note that an image provides only partial appearance information as it only depicts the object from a single perspective. Furthermore, we encode the image into a viewpoint-invariant global feature representation $\mathbf{z} \in \mathcal{Z}$. Thus, our approach does not assume the camera extrinsics wrt. the object to be known and therefore is able to texture untextured 3D shapes using images "in the wild". Importantly, the input image need not depict an object of the exact same shape as the 3D model. This would be a strong restriction in practice as often only an approximate shape can be retrieved from a single image.

In summary, we define a *Texture Field* as a mapping from 3D point $\mathbf{p}$, shape embedding $\mathbf{s}$ and condition $\mathbf{z}$ to color $\mathbf{c}$:

$$t_\theta : \mathbb{R}^3 \times \mathcal{S} \times \mathcal{Z} \to \mathbb{R}^3 \tag{2}$$

In the following, we will consider both the conditional case as well as the unconditional case. For the unconditional case, we exploit probabilistic generative networks [9, 19], capturing ambiguity in a *random* latent code $\mathbf{z}$.

Model: An overview over our *Texture Field* model is shown in Fig. 2. Colored arrows indicate alternative pathways representing the Conditional, GAN and VAE model. Blue and red boxes denote the trainable components of our model which are parameterized by neural networks whose parameters are trained jointly. We now provide details on each of the components of our model.

Shape Encoder: To inform the Texture Field t_θ about the 3D shape of the object, we uniformly sample 3D points from the input shape (typically provided in form of a traingular mesh) and pass the resulting point cloud to a PointNet encoder [25], adopting the network architecture of [21]. This results in a fixed-dimensional shape embedding $\mathbf{s}$.

Image Encoder: For the conditional case (red arrows in Fig. 2), we additionally extract appearance information from an input image. More specifically, we encode the input image into a fixed-dimensional latent code $\mathbf{z}$ using a standard pre-trained residual network [12] with 18 layers.

Texture Field: Given the input shape $\mathbf{s}$ and the condition $\mathbf{z}$, our proposed Texture Field model is able to predict a color value $\mathbf{c}_i$ for any 3D point $\mathbf{p}_i$. Thus, it would be possible to directly color every point on the 3D mesh. Unfortunately, mesh-based appearance representations require additional UV-mappings for representing texture. We therefore train our texture model in 2D image space which provides a regular and hence efficient representation[1].

Towards this goal, we render depth maps $\mathbf{D}$ and corresponding color images $\mathbf{X}$ from arbitrary viewpoints using OpenGL. The color at pixel $\mathbf{u}_i$ and depth d_i is predicted as

$$\hat{\mathbf{c}}_i = t_\theta\left(d_i\, \mathbf{R}\, \mathbf{K}^{-1}\mathbf{u}_i + \mathbf{t}, \mathbf{s}, \mathbf{z}\right) \tag{3}$$

where i denotes the index for pixels with finite depth values $i \in \{1, \dots, N\}$ and N refers to the number of foreground pixels in the image (i.e., pixels where the object is visible). Here, the camera intrinsics and extrinsics are denoted by $\mathbf{K} \in \mathbb{R}^{3\times3}$ and $(\mathbf{R} \in \mathbb{R}^{3\times3}, \mathbf{t} \in \mathbb{R}^3)$, respectively, and pixel $\mathbf{u}_i$ is represented in homogeneous coordinates. For training our model, we compare the predicted color $\hat{\mathbf{c}}_i$ to the color of the corresponding pixel $\mathbf{c}_i$ in the rendered image $\mathbf{X}$.

3.2. Training

This sections describes how we train our conditional and unconditional models. See Fig. 2 for a visual illustration.

Conditional Setting: In the conditional case, we input an embedding $\mathbf{z}$ of the input image to our network[2]. We train $t_\theta(\mathbf{p}, \mathbf{s}, \mathbf{z})$ in a supervised setting by minimizing the ℓ_1-loss between the predicted image $\hat{\mathbf{X}}$ and the rendered image $\mathbf{X}$:

$$\mathcal{L}_{cond} = \frac{1}{B}\sum_{b=1}^{B}\sum_{i=1}^{N_b} \|t_\theta(\mathbf{p}_{bi}, \mathbf{s}_b, \mathbf{z}_b) - \mathbf{c}_{bi}\|_1 \tag{4}$$

Here, B represents the mini-batch size. Each element of the mini batch represents an image with N_b foreground pixels (i.e., pixels where the object is visible). Note that the shape encoding $\mathbf{s}_b$ and the input encoding $\mathbf{z}_b$ implicitly depend on the parameters of the shape and image encoder networks, respectively. We train the parameters of all three networks (shape encoder, image encoder, Texture Field) jointly.

Unconditional Generative Model: In the unconditional setting we are only given the 3D shape as input but no additional information about the appearance of the object. As mentioned above, this is a highly ill-posed task with many *valid* explanations. We therefore utilize probabilistic generative models which capture the ambiguity in the output using a latent code $\mathbf{z}$ sampled from a Gaussian distribution (see green and blue models in Fig. 2). In particular, we adapt two recent deep latent variable models to our setting: a generative adversarial networks (GAN) [9] and a variational auto-encoder (VAE) [19]. Both models have been applied to a variety of image-based tasks in the past [9, 10, 16, 19, 26, 29] and are thus ideally suited in the context of our image-based loss formulation.

Let us first consider adversarial training. The problem of learning a generative model for texture information given a 3D shape can be tackled using a *conditional GAN* [22] where the generator is conditioned on the 3D shape. The generator is represented as a Texture Field $t_\theta : \mathbb{R}^3 \times \mathcal{S} \times \mathcal{Z} \rightarrow \mathbb{R}^3$ which maps the latent code $\mathbf{z}$ for every given 3D location $\mathbf{p}_i$ conditioned on the shape embedding $\mathbf{s}$ to an RGB image:

$$\hat{\mathbf{X}} = G_\theta(\mathbf{z}_b|\mathbf{D}_b, \mathbf{s}_b) = \{t_\theta(\mathbf{p}_{bi}, \mathbf{s}_b, \mathbf{z}_b) | i \in \{1, \dots, N_b\}\} \tag{5}$$

As above, b denotes one element of the mini-batch and $\mathbf{s}_b$ depends on the parameters of the shape encoder. We use a standard image-based discriminator $D_\phi(\mathbf{X}_b|\mathbf{D}_b)$ conditioned on the input depth image $\mathbf{D}$ by concatenating it with the input image. For training the model, we use a non-saturating GAN loss with R_1-regularization [20].

An alternative method for learning a latent variable model is given by a *conditional VAE* (cVAE) [32]. Our cVAE model comprises an encoder network that maps color image $\mathbf{X}$ to mean $\boldsymbol{\mu}$ and variance $\boldsymbol{\sigma}$ of an isotropic normal distributed random variable $\mathbf{z}$ which follows the distribution $q_\phi(\mathbf{z}|\mathbf{X}, \mathbf{s})$. Our Texture Field model is used as decoder by predicting the color value $\hat{\mathbf{c}}_i$ of each pixel $\mathbf{u}_i$ for the corresponding 3D locations $\mathbf{p}_i$ conditioned on the shape embedding $\mathbf{s}$. Following [19, 29], we minimize the variational lower bound

$$\mathcal{L}_{\text{VAE}} = \frac{1}{B}\sum_{b=1}^{B}\Bigg[\beta\ \text{KL}(q_\phi(\mathbf{z}|\mathbf{X}_b, \mathbf{s}_b)\ \|\ p_0(\mathbf{z}_b)) + \sum_{i=1}^{N_b} \|t_\theta(\mathbf{p}_{bi}, \mathbf{s}_b, \mathbf{z}_b) - \mathbf{c}_{bi}\|_1\Bigg] \tag{6}$$

where KL refers to the Kullback-Leibler divergence, $p_0(\mathbf{z}) = \mathcal{N}(\mathbf{z}|0, \mathbf{I})$ denotes the standard normal distribution, $\mathbf{z}_b$ is a sample from the posterior distribution

[1]Note that this does not restrict our model in any way. At test time, our model can be evaluated for arbitrary 3D points.

[2]It is important to note that the input image and the images $\mathbf{X}$ used for supervising the model during training need *not* be the same as we learn a viewpoint-invariant representation $\mathbf{z}$.

$q_\phi(\mathbf{z}|\mathbf{X}_b, \mathbf{s}_b)$ and β is a trade-off parameter between the KL-divergence and the reconstruction loss [14]. Again, b denotes one element of the mini-batch and $\mathbf{s}_b$ depends on the parameters of the shape distribution. In practice, we set $\beta = 1$. During training, we optimize $\mathcal{L}_{\text{VAE}}$ using the reparameterization trick [19, 29]. At test time, we obtain new texture samples for the 3D object under consideration by sampling $\mathbf{z}$ from a standard normal distribution.

3.3. Implementation details

In all of our experiments, we implement the texture field $t_\theta(\cdot, \mathbf{s}, \mathbf{z})$ using the fully connected ResNet [12] architecture from [21], see supplementary for details. For the image encoder, we use a ResNet-18 architecture [12], pretrained on ImageNet. For the shape encoder, we adopt the PointNet [25] architecture from [21]. The GAN discriminator and the VAE encoder are based on the discriminator from [20]. We train both our supervised model and our VAE model end-to-end using Adam [18] with learning rate 10^{-4}. Our GAN is trained with alternating gradient descent using the RMSProp Optimizer [36] with the same learning rate.

4. Experimental Evaluation

We evaluate our approach in three different experiments. In the first part, we investigate the **representation power** of Texture Fields by analyzing how well the method can represent high frequency textures when trained on a single 3D object. In the second part, we apply our method to the challenging task of **single view texture reconstruction** in which we predict the full texture of 3D objects given only the 3D shape and a single view of this object. Moreover, we combine our method with a state-of-the-art shape reconstruction method [21] for full textured 3D reconstruction which also includes reconstructing the shape of the 3D object. In the last experiment, we explore if our representation can also be used in a **generative setting** where the goal is to produce a varied distribution of possible textures given only the shape of the 3D object without further input.

Baselines: Only few prior works have considered texture reconstruction in the general case without predefined template mesh or information about the camera view. Therefore, we construct a first simple baseline by mapping the input image onto the mesh by projecting all vertices into the input view to determine their color. While this simple baseline receives additional information about the camera of the input view, we believe it still serves as a good sanity check to see if our method is actually learning something useful and if it is able to correctly fill-in occluded regions. As a second baseline we consider a novel-view-synthesis (NVS) approach which uses the same image encoder as our approach, but applies a UNet architecture[3] [30] to transform a depth rendering of the object into an RGB image. While this approach can also generate novel views of the object under consideration, it requires additional (lossy) post-processing to generate a complete texture map of the object. In particular, there is no guarantee for this baseline that the newly generated views are consistent under viewpoint changes. Lastly, we consider Im2Avatar [34] as a baseline for full textured 3D reconstruction from single images. To the best of our knowlege, this method currently achieves the highest resolution (64^3) among all voxel-based 3D reconstruction methods which predict colors and produces state-of-the-art results. We use the official implementation[4] of Im2Avatar.

[3]See supplementary material for details.

Dataset: Unless specified otherwise, we use the categories 'cars', 'chairs', 'airplanes' and 'tables' from the ShapeNet dataset [5] as these categories contain rich texture information while also providing a large variety of shapes. For our conditional experiments, we use the renderings provided by Choy et al. [7] as input. For training our models, we render 10 additional images and depth maps per object from random views in the upper hemisphere.

Metrics: For evaluation, we consider three different metrics in image space for random views of the objects. Firstly, to evaluate how well our method and the baselines capture the appearance distribution for a given object category, we use the Fréchet Inception Distance (FID) [13]. This is a common metric between distributions of images and widely used in the GAN literature [3, 13, 16, 17]. Moreover, to better estimate the distance between a predicted view and the ground truth on a per-instance basis, we measure the structure similarity image metric (SSIM) [39]. Since we found that SSIM mostly captures local properties of images, we additionally introduce a more global perceptual measure, which we call *Feature-ℓ_1-metric*. Similarly to FID, the Feature-ℓ_1-metric is computed by embedding both the image produced by the method under consideration and the ground truth view into feature space using an Inception network. The Feature-ℓ_1-metric then computes the mean absolute distance between the feature activations of the predicted and the ground truth image.

4.1. Representation Power

The goal of this experiment is to explore the representation power of our Texture Field model. This "overfitting" experiment helps to disentangle the quality of the image encoder from the Texture Field representation itself and provides an upper bound on the reconstruction quality that we can expect when applying Texture Fields to more difficult tasks. We train our method separately on 3D meshes of a cat and a human[5], wrt. 512^2 px renderings from 500 views.

Qualitative results are shown in Fig. 3, comparing our

[4]https://github.com/syb7573330/im2avatar

[5]3D models from free3d.com and turbosquid.com.

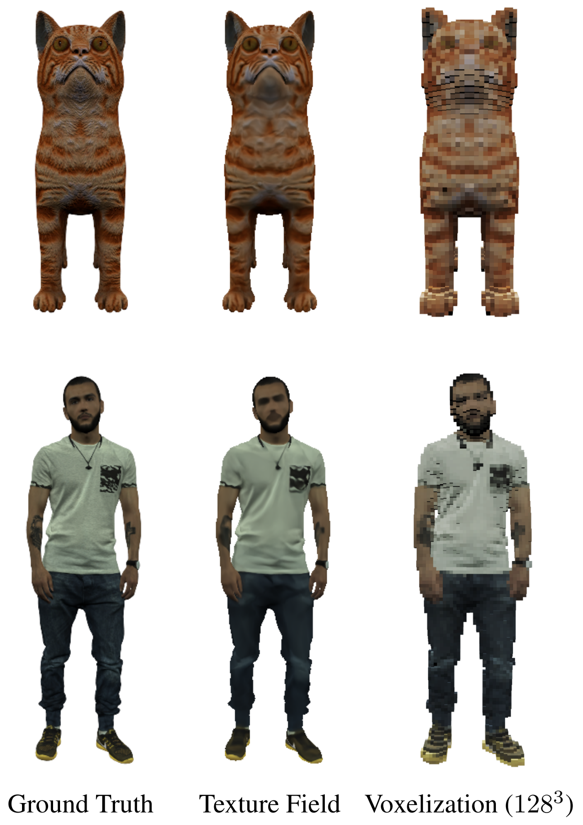

Figure 3: **Representation Power.** Comparison of a Texture Field fitted to a 3D model of a cat/human to a voxel-based representation and the corresponding ground truth model.

model to a voxelization at a fixed resolution of 128^3 voxels. We observe that Texture Fields can represent high-frequency information while the voxel representation is inherently restricted to a Manhattan world of limited granularity. This experiment validates that Texture Fields are a promising texture representation.

4.2. Single Image Texture Reconstruction

We now turn our attention to the challenging task of single-image texture reconstruction. Towards this goal, we conduct experiments for texture reconstruction given a 3D model together with a 2D image of the object from a random camera view. We train our method using the conditional setup described in Section 3.2. At test time we apply the trained model in three different settings. In the first setting, we use ground truth 3D shapes as input and apply our model to synthetic renderings of the object. In the second setting, we combine our method with a state-of-the art shape reconstruction approach [21] to obtain a full 3D reconstruction pipeline of both shapes and textures. Finally, in the third setting, we investigate if our model also transfers

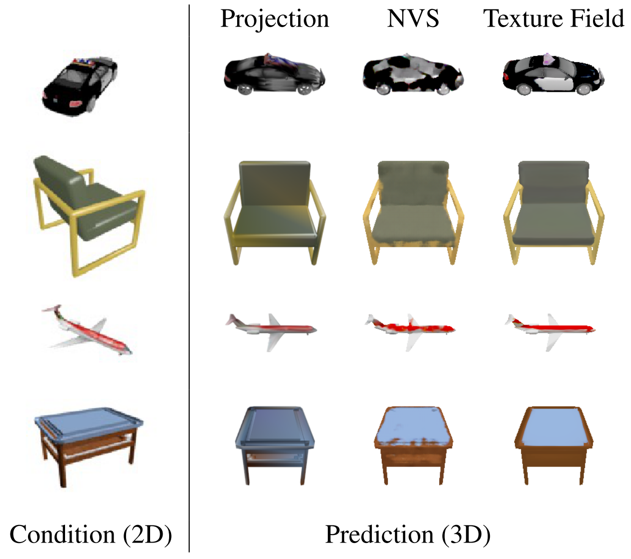

Figure 4: **Texture Reconstruction.** A qualitative comparison between our method (Texture Field) and the baselines is shown. In this experiment, we use GT shapes as input.

to real data. Here, we use images of real cars together with similar 3D shapes from the ShapeNet dataset.

GT Shapes: When using ground truth shapes, our model successfully learns to extract texture from just a single image of the object as illustrated in Fig. 4. In particular, our model can complete parts of the texture that were not visible in the input view. At the same time, our model successfully transfers texture regions visible in the input view to the shape (e.g. windows and tires of the car). In contrast, both the projection baseline and NVS show considerable artifacts. While the projection baseline transfers texture incorrectly into ambiguous regions, NVS often leads to unrealistic image artifacts such as rough edges and bleeding colors. A quantitative comparison is provided in Table 1. While NVS achieves the best SSIM, our method performs best in terms of FID and Features-ℓ_1 distance. This is consistent with the qualitative result in Fig. 4, as SSIM is a local score whereas FID and the Features-ℓ_1 distance are global scores that better capture visual similarity and realism.

Full Pipeline: In order to obtain a full single-view textured 3D reconstruction pipeline, we combine Texture Fields with Occupancy Networks [21]. For a fair comparison, we also combine the projection baseline and the NVS baseline with the output from [21].

Our qualitative results in Fig. 7 and Fig. 8 demonstrate that our approach is able to reconstruct 3D models with texture from just a single view of the model. In contrast to Im2Avatar[6] [34], NVS and the projection baseline, we ob-

[6]Unfortunately, Sun et al. provide training data only for a subset of our training data. We therefore train and evaluate Im2Avatar on the training

	FID			SSIM			Feature-ℓ_1		
	Projection	NVS	Texture Field	Projection	NVS	Texture Field	Projection	NVS	Texture Field
airplanes	15.375	17.816	**9.236**	**0.970**	0.964	0.968	0.143	0.158	**0.136**
cars	70.070	72.209	**24.271**	0.840	**0.887**	0.885	0.236	0.238	**0.192**
chairs	8.045	8.788	**5.791**	0.931	**0.947**	0.941	0.127	0.125	**0.124**
tables	10.254	9.311	**8.846**	0.934	**0.953**	0.943	0.123	**0.117**	0.123
mean	25.936	27.031	**12.036**	0.919	**0.938**	0.934	0.157	0.159	**0.144**

Table 1: **Single Image Texture Reconstruction.** Quantitative Evaluation using the FID, SSIM and Feature-ℓ_1 metrics.

	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**
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**
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**
mean	70.785	141.209	69.306	**51.659**	0.844	0.734	**0.867**	**0.867**	0.229	0.281	0.225	**0.211**

Table 2: **Full Pipeline.** Quantitative Evaluation using the FID, SSIM and Feature-ℓ_1 metrics.

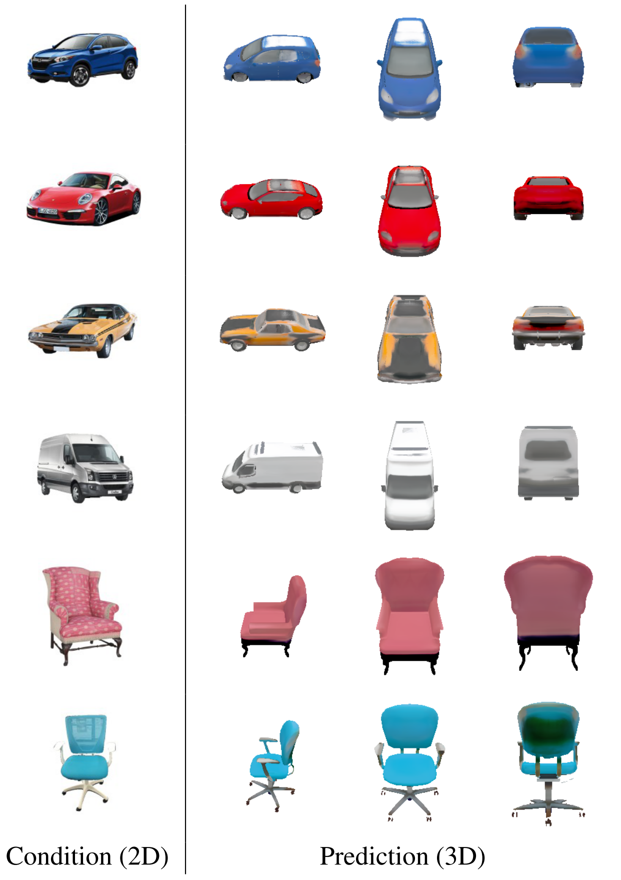

Figure 5: **Texture Reconstruction from Real Images.** In this experiment, our model transfers texture from previously unseen *real* images to unseen 3D CAD models.

and test split provided by Sun et al. As the test split provided by Sun et al. and our test split are disjoint for the "chairs" and "tables" categories, we can therefore also only show a qualitative comparision on "cars" category.

serve that our method achieves more consistent and realistic looking outputs. This is also reflected in the numerical results in Table 2: while NVS and our method both obtain the best SSIM, our method achieves the best FID and Features-ℓ_1 distances.

Real images: In Fig. 5, we finally investigate whether our approach is also able to transfer texture information from real input images to ground truth CAD models. Towards this goal, we apply our approach to the images provided by [33] and [42] and select CAD models that are similar to the object shown in the input image. We observe that our model generalizes reasonably well to real data despite only being trained on synthetic data.

4.3. Unconditional Model

In this section, we conduct unconditional experiments to investigate if Texture Fields can also be applied in a purely generative manner where we only provide the shape of the object to the network, but not the 2D image. Towards this goal, we train both, the VAE and the GAN model, on the "cars" category. During training, we provide both target images and depth maps but no input views to the network. During testing, we sample the latent code z from a standard normal distribution to obtain random texture samples for the given 3D object.

Random samples for the GAN and the VAE model are shown in Fig. 10. While our unconditional models successfully generate realistic textures, both the VAE and GAN samples contain artifacts similar to those present when applying VAEs and GANs to the image domain. For example, the samples of the VAE model are globally consistent but slightly blurry. In contrast, the samples for the GAN model are sharper but contain artifacts. In the future, we would like to explore combinations of VAEs and GANs and more

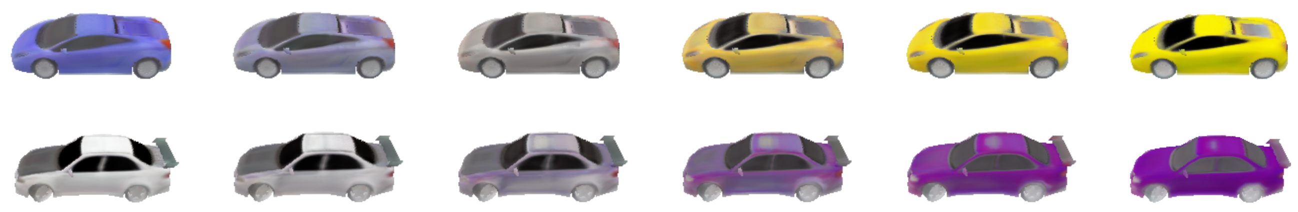

Figure 6: **Latent Space Interpolations (VAE).** Our generative model learns a meaningful latent space embedding.

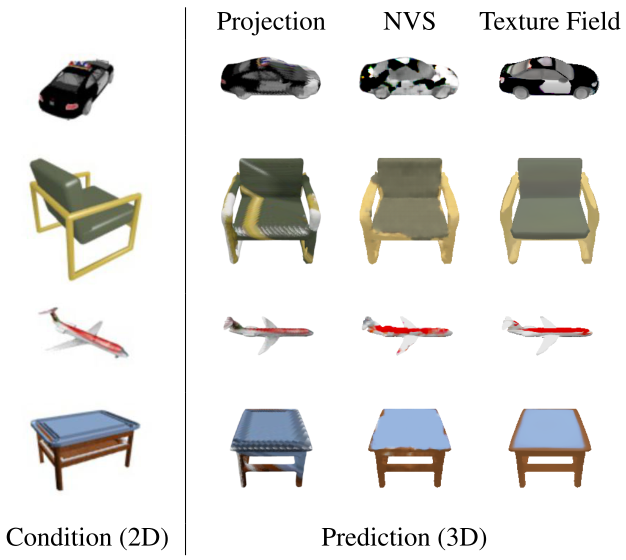

Figure 7: **Full Pipeline.** Results using Occupancy Networks [21] for 3D reconstruction in combination with projection, NVS and Texture Fields for appearance estimation.

advanced models and training methods [10, 17] to improve upon these initial results.

Fig. 6 shows interpolations in the latent space for the VAE model. We see that the VAE has learned a meaningful latent space and can hence interpolate smoothly between different texture samples. Moreover, in Fig. 9, we demonstrate that our VAE model also allows for successfully transferring texture from one model to another one.

5. Conclusion

In this paper we introduced Texture Fields, a novel continuous representation for texture of 3D shapes. Our experiments show that Texture Fields can predict high frequency textures from just a single object view. Moreover, we have demonstrated that Texture Fields can also be used in an unconditional setting where we are only given the shape of the 3D object. We hence believe that Texture Fields are a useful representation for 3D reconstruction and hope that they will become an integral part of the next generation of 3D generative models.

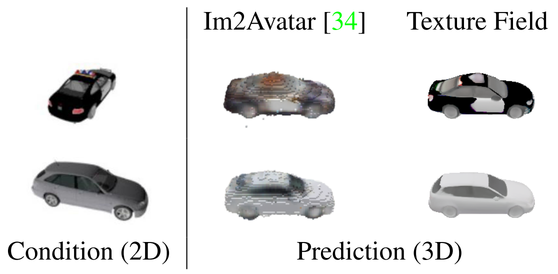

Figure 8: **Voxels vs. Function Space.** We compare our full pipeline against a voxel-based reconstruction method [34].

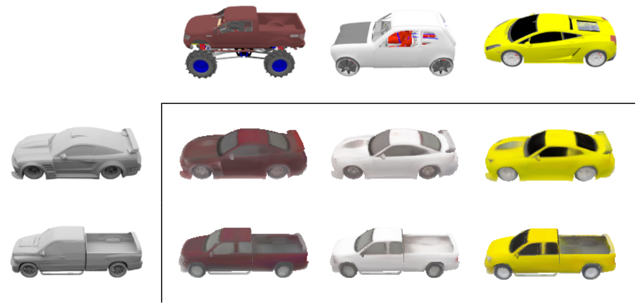

Figure 9: **Texture Transfer (VAE).** Our model transfers appearance information (top) to the target models (left).

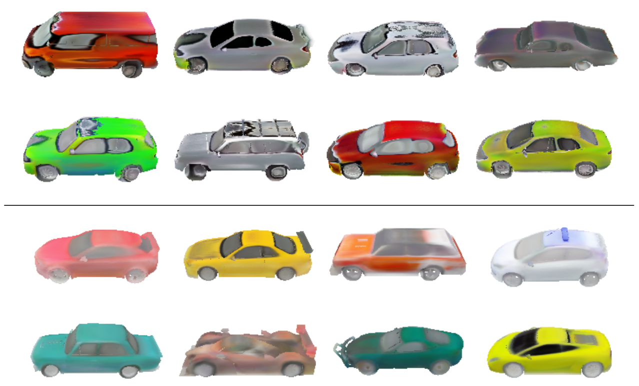

Figure 10: **Generative Model.** Textures generated using the GAN (top 2 rows) and VAE (bottom 2 rows) models. Note that no 2D image is provided as input in this setting.

Acknowledgements

This work was supported by Microsoft Research through its PhD Scholarship Program, by the Intel Network on Intelligent Systems and by an NVIDIA research gift.

References

[1] H. Alhaija, S. Mustikovela, A. Geiger, and C. Rother. Geometric image synthesis. In *Proc. of the Asian Conf. on Computer Vision (ACCV)*, 2018. 2

[2] H. Alhaija, S. Mustikovela, L. Mescheder, A. Geiger, and C. Rother. Augmented reality meets computer vision: Efficient data generation for urban driving scenes. *International Journal of Computer Vision (IJCV)*, 126(9):961–972, 2018. 2

[3] A. Brock, J. Donahue, and K. Simonyan. Large scale GAN training for high fidelity natural image synthesis. In *Proc. of the International Conf. on Learning Representations (ICLR)*, 2019. 2, 5

[4] A. Brock, T. Lim, J. M. Ritchie, and N. Weston. Generative and discriminative voxel modeling with convolutional neural networks. *arXiv.org*, 1608.04236, 2016. 1

[5] A. X. Chang, T. A. Funkhouser, L. J. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su, J. Xiao, L. Yi, and F. Yu. Shapenet: An information-rich 3d model repository. *arXiv.org*, 1512.03012, 2015. 5

[6] Z. Chen and H. Zhang. Learning implicit fields for generative shape modeling. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2018. 1, 2

[7] C. B. Choy, D. Xu, J. Gwak, K. Chen, and S. 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. 1, 2, 5, 12

[8] M. Garnelo, J. Schwarz, D. Rosenbaum, F. Viola, D. J. Rezende, S. M. A. Eslami, and Y. W. Teh. Neural processes. In *Proc. of the International Conf. on Machine learning (ICML) Workshops*, 2018. 2

[9] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. C. Courville, and Y. Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems (NIPS)*, 2014. 2, 3, 4

[10] K. Gregor, I. Danihelka, A. Graves, D. J. Rezende, and D. Wierstra. DRAW: A recurrent neural network for image generation. In *Proc. of the International Conf. on Machine learning (ICML)*, 2015. 4, 8

[11] T. Groueix, M. Fisher, V. G. Kim, B. C. Russell, and M. Aubry. AtlasNet: A papier-mâché approach to learning 3d surface generation. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2018. 1, 2

[12] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2016. 4, 5, 11, 12

[13] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *Advances in Neural Information Processing Systems (NIPS)*, 2017. 5

[14] I. Higgins, L. Matthey, A. Pal, C. Burgess, X. Glorot, M. Botvinick, S. Mohamed, and A. Lerchner. beta-VAE: Learning basic visual concepts with a constrained variational framework. In *Proc. of the International Conf. on Machine learning (ICML)*, 2017. 5

[15] A. Kanazawa, S. Tulsiani, A. A. Efros, and J. Malik. Learning category-specific mesh reconstruction from image collections. In *Proc. of the European Conf. on Computer Vision (ECCV)*, 2018. 1, 2

[16] T. Karras, T. Aila, S. Laine, and J. Lehtinen. Progressive growing of GANs for improved quality, stability, and variation. In *Proc. of the International Conf. on Learning Representations (ICLR)*, 2018. 2, 4, 5

[17] T. Karras, S. Laine, and T. Aila. A style-based generator architecture for generative adversarial networks. *arXiv.org*, 2018. 2, 5, 8

[18] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *Proc. of the International Conf. on Learning Representations (ICLR)*, 2015. 5

[19] D. P. Kingma and M. Welling. Auto-encoding variational bayes. *Proc. of the International Conf. on Learning Representations (ICLR)*, 2014. 3, 4, 5

[20] L. Mescheder, A. Geiger, and S. Nowozin. Which training methods for gans do actually converge? In *Proc. of the International Conf. on Machine learning (ICML)*, 2018. 2, 4, 5, 12

[21] L. Mescheder, M. Oechsle, M. Niemeyer, S. Nowozin, and A. Geiger. Occupancy networks: Learning 3d reconstruction in function space. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019. 1, 2, 3, 4, 5, 6, 8, 11, 16

[22] M. Mirza and S. Osindero. Conditional generative adversarial nets. *arXiv.org*, 1411.1784, 2014. 4

[23] E. Park, J. Yang, E. Yumer, D. Ceylan, and A. C. Berg. Transformation-grounded image generation network for novel 3d view synthesis. *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, July 2017. 2

[24] J. J. Park, P. Florence, J. Straub, R. Newcombe, and S. Lovegrove. DeepSDF: Learning continuous signed distance functions for shape representation. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019. 2

[25] C. R. Qi, H. Su, K. Mo, and L. 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. 4, 5, 11

[26] A. Radford, L. Metz, and S. Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. In *Proc. of the International Conf. on Learning Representations (ICLR)*, 2016. 2, 4

[27] K. Rematas, C. H. Nguyen, T. Ritschel, M. Fritz, and T. Tuytelaars. Novel views of objects from a single image. *IEEE Trans. Pattern Anal. Mach. Intell.*, 39(8):1576–1590, 2017. 2

[28] D. J. Rezende, S. M. A. Eslami, S. Mohamed, P. Battaglia, M. Jaderberg, and N. Heess. Unsupervised learning of 3d structure from images. In *Advances in Neural Information Processing Systems (NIPS)*, 2016. 1

[29] D. J. Rezende, S. Mohamed, and D. Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In *Proc. of the International Conf. on Machine learning (ICML)*, 2014. 4, 5

[30] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, 2015. 5, 12

[31] S. Saito, L. Wei, L. Hu, K. Nagano, and H. Li. Photorealistic facial texture inference using deep neural networks. *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 2326–2335, 07 2017. 1, 2

[32] K. Sohn, H. Lee, and X. Yan. Learning structured output representation using deep conditional generative models. *Advances in Neural Information Processing Systems (NIPS)*, 2015. 4

[33] X. Sun, J. Wu, X. Zhang, Z. Zhang, C. Zhang, T. Xue, J. B. Tenenbaum, and W. T. Freeman. Pix3d: Dataset and methods for single-image 3d shape modeling. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2018. 7

[34] Y. Sun, Z. Liu, Y. Wang, and S. E. Sarma. Im2avatar: Colorful 3d reconstruction from a single image. *arXiv.org*, abs/1804.06375, 2018. 1, 2, 5, 6, 8

[35] M. Tatarchenko, A. Dosovitskiy, and T. Brox. Multi-view 3d models from single images with a convolutional network. In *Proc. of the European Conf. on Computer Vision (ECCV)*, 2016. 2

[36] T. Tieleman and G. Hinton. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*, 2012. 5

[37] S. Tulsiani, T. Zhou, A. A. Efros, and J. Malik. Multi-view supervision for single-view reconstruction via differentiable ray consistency. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2017. 1, 2

[38] N. Wang, Y. Zhang, Z. Li, Y. Fu, W. Liu, and Y.-G. Jiang. Pixel2Mesh: Generating 3D mesh models from single RGB images. In *Proc. of the European Conf. on Computer Vision (ECCV)*, 2018. 1, 2

[39] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Trans. on Image Processing (TIP)*, 13(4):600–612, 2004. 5

[40] 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 (NIPS)*, 2016. 1

[41] T. Zhou, S. Tulsiani, W. Sun, J. Malik, and A. A. Efros. View synthesis by appearance flow. In *Proc. of the European Conf. on Computer Vision (ECCV)*, 2016. 2

[42] J.-Y. Zhu, Z. Zhang, C. Zhang, J. Wu, A. Torralba, J. B. Tenenbaum, and W. T. Freeman. Visual object networks: Image generation with disentangled 3D representations. *Advances in Neural Information Processing Systems (NIPS)*, 2018. 2, 7

Supplementary Material

A. 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.

A.1. Architectures

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

Texture Field: In Fig. 11, the network architecture of the Texture Field is shown. We adapt the architecture proposed in [21] 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. 11 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 [12]. 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.

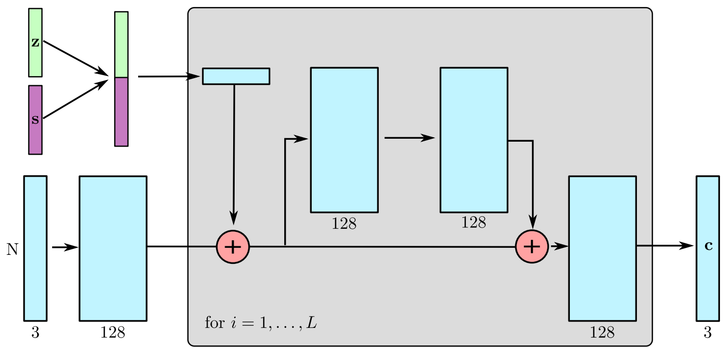

Figure 11: **Texture Field.** This figure illustrates the architecture of Texture Fields. Shape Embedding $\mathbf{s}$ and condition/latent texture code $\mathbf{z}$ are injected to each ResNet block. For each of the N 3D points, the network outputs a 3-dimensional color value $\mathbf{c}$.

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 [21]. The architecture is depicted in Fig. 12. Based on PointNet [25], the network consists of 5 Resnet blocks with max pooling layers.

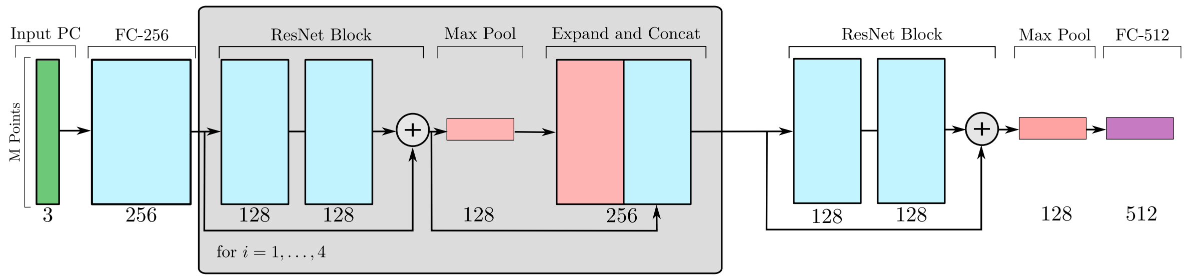

Figure 12: **Shape Encoder.** Similar to a PointNet encoder [25] , the network shown here determines a feature vector from a set of points. We use the ResNet-based version, proposed in [21].

Image Encoder: As encoder for the input image, we use a pretrained ResNet-18 architecture [12], visualized in Fig. 13. After the last ResNet-Block we apply a average pooling layer and a fully-connected layer for deriving the image embedding **z**.

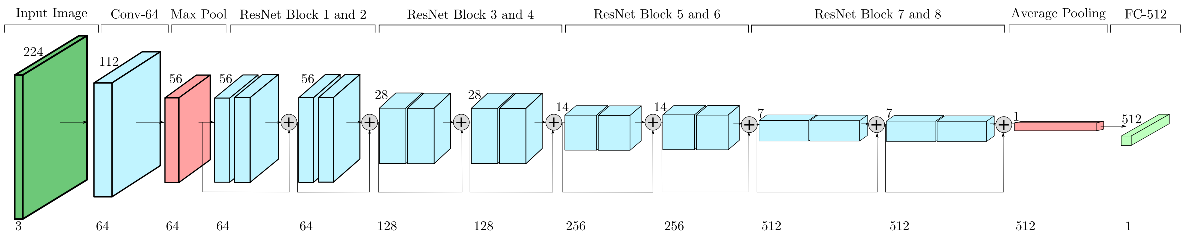

Figure 13: **Image Encoder.** As encoder of the input images for the conditional task, we apply the ResNet-18 network pretrained on ImageNet.

VAE Encoder: For training the VAE, we encode the ground truth views using the ResNet-based network in Fig. 14. The encoder receives an image as well as the shape embedding as input and predicts mean μ and log-standard deviation $\log \sigma$ of normal- distributed random variables in the latent space. For injecting the shape embedding, we pass **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 **z**.

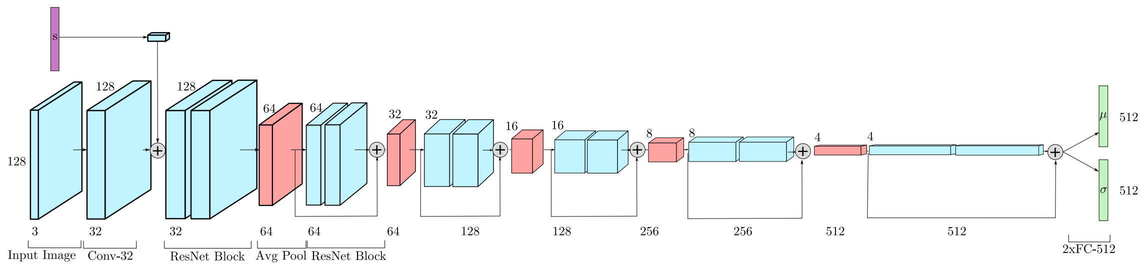

Figure 14: **VAE Encoder.** Here, we illustrate the network of the encoding part of the VAE. The encoder maps an image and the shape embedding **s** to mean μ and log-standard deviation $\log \sigma$ of the latent variable **z**.

GAN Discriminator: We apply the network shown in Fig. 15 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 [20], the input is mapped to a single scalar.

A.2. NVS Baseline

In Fig. 16, 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 [30] and inject the image encoding into each layer.

A.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 ShapeNet categories 'cars', 'chairs', 'airplanes' and 'tables'. 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 and 8,512 table 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. [7].

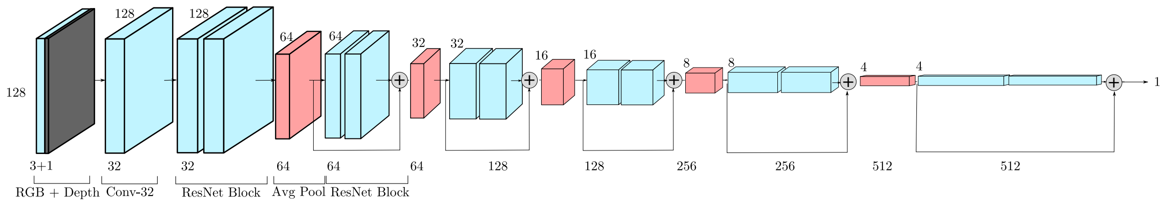

Figure 15: **GAN Discriminator.** As input for the discriminator we use a RGB image and a corresponding depth image. By using average pooling and ResNet blocks, the input is mapped to a single scalar value.

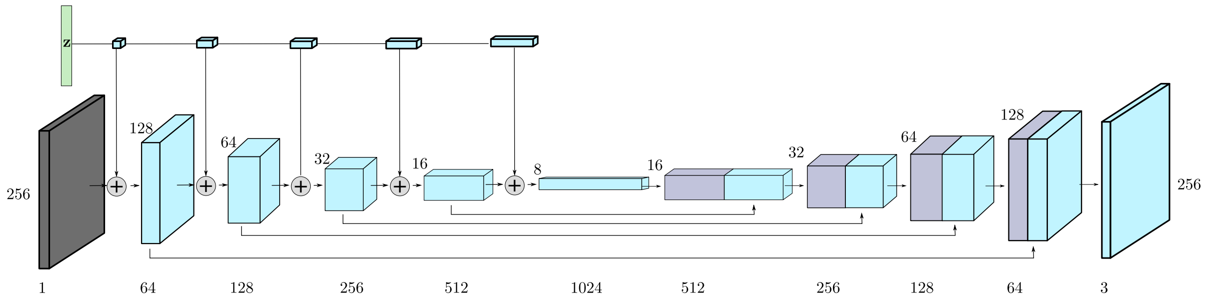

Figure 16: **NVS.** In this figure, we show the U-Net-based architecture of the NVS baseline. We inject the image embedding $\mathbf{z}$ into each layer in the encoding part of the network.

B. Further Results

We present more results for each of the experiments in the following Figures 17, 18, 19, 20, 21, 22 and 23.

C. 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. 24, 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.

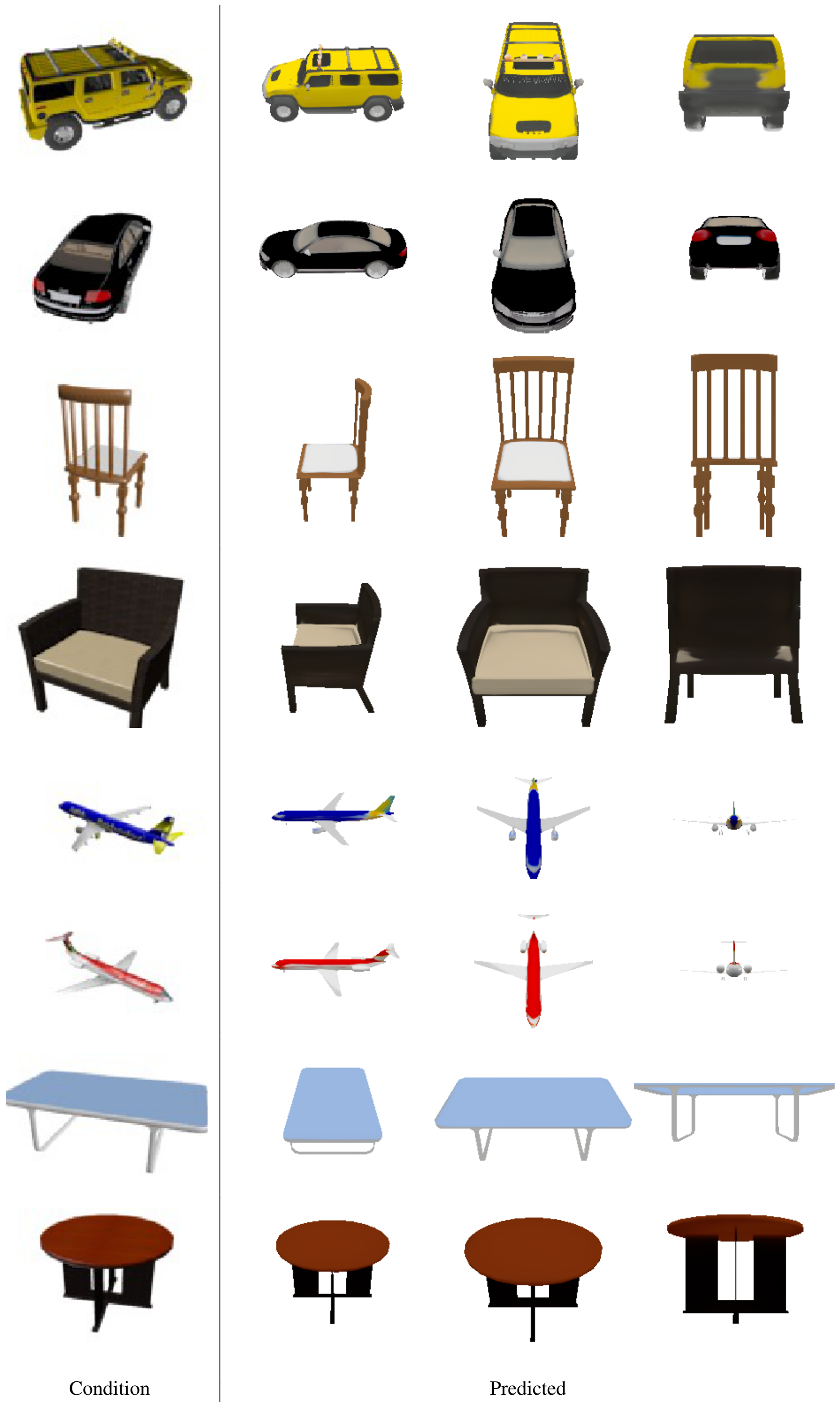

Figure 17: **Texture Reconstruction with Texture Field.** In this Figure, we use our model to predict texture for untextured 3D CAD models based on a single view of the same objects. The texture is properly predicted for all categories and contains details as lights and number plates. Very high-frequency details are sometimes leading to some blurriness.

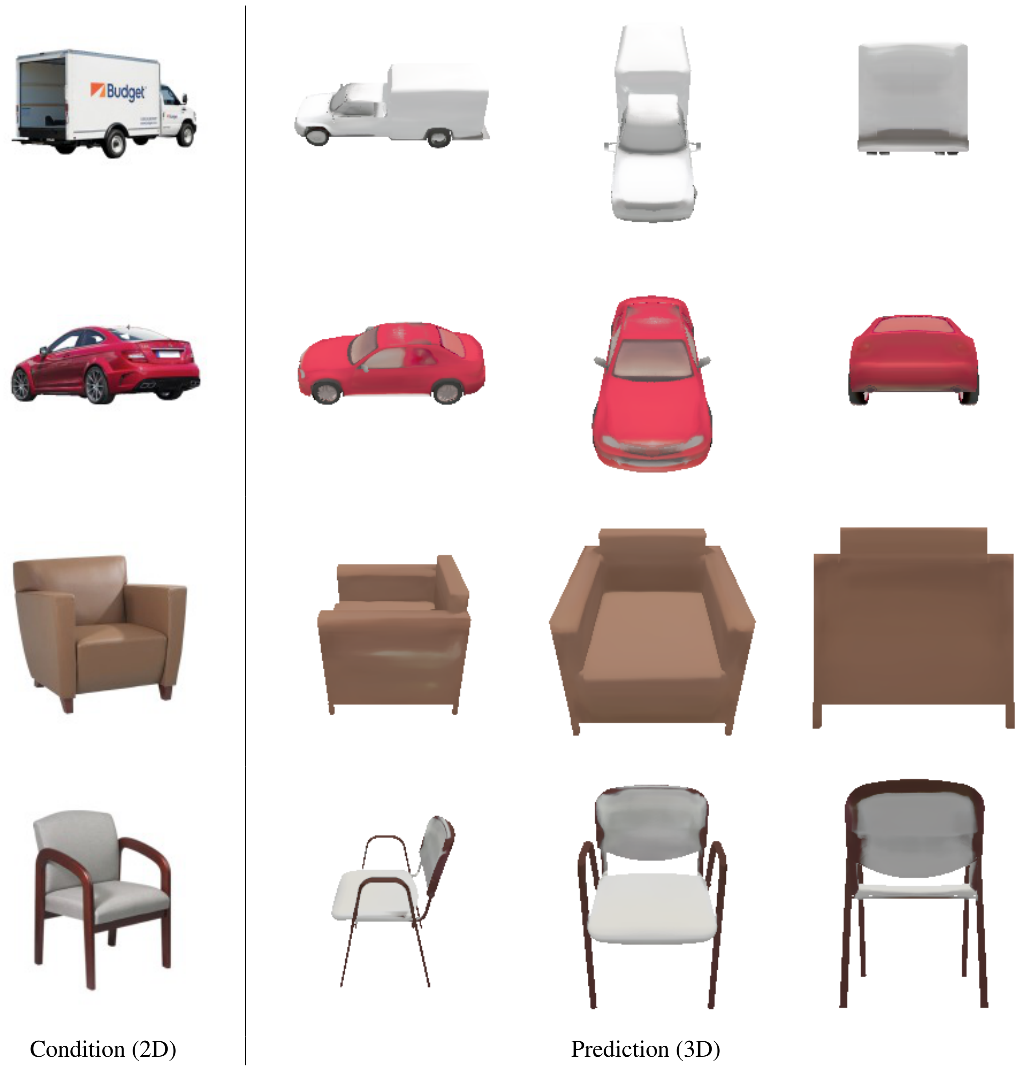

Figure 18: **Texture Reconstruction with Real Images.** In this figure, we show results for texturing untextured synthetic CAD models from a single real input image using our approach.

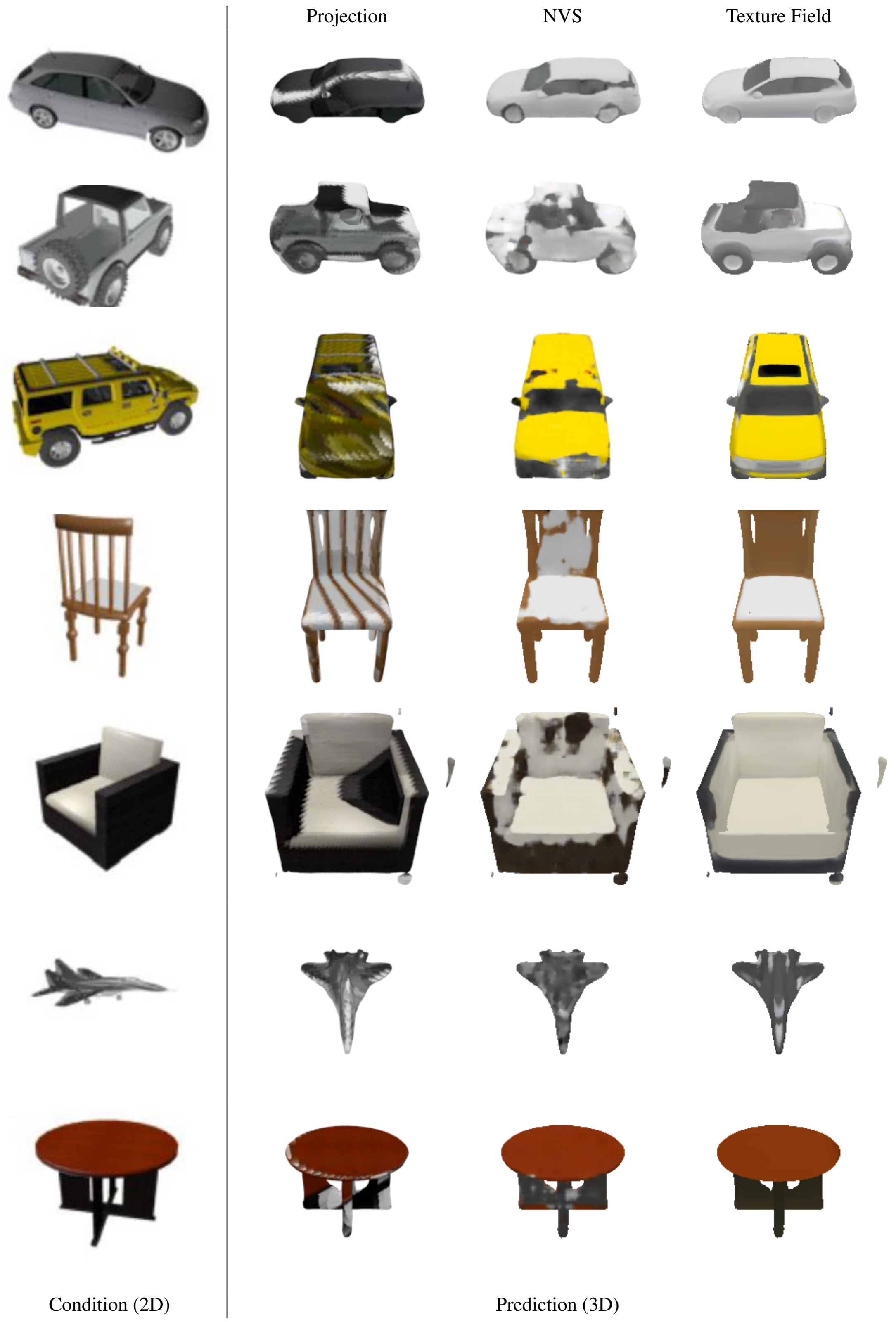

Figure 19: **Full Pipeline.** Our full pipeline for texture and shape reconstruction leads to plausible textured 3D objects, as shown in this figure. We use the same shape reconstruction model (ONet) [21] for all approaches.

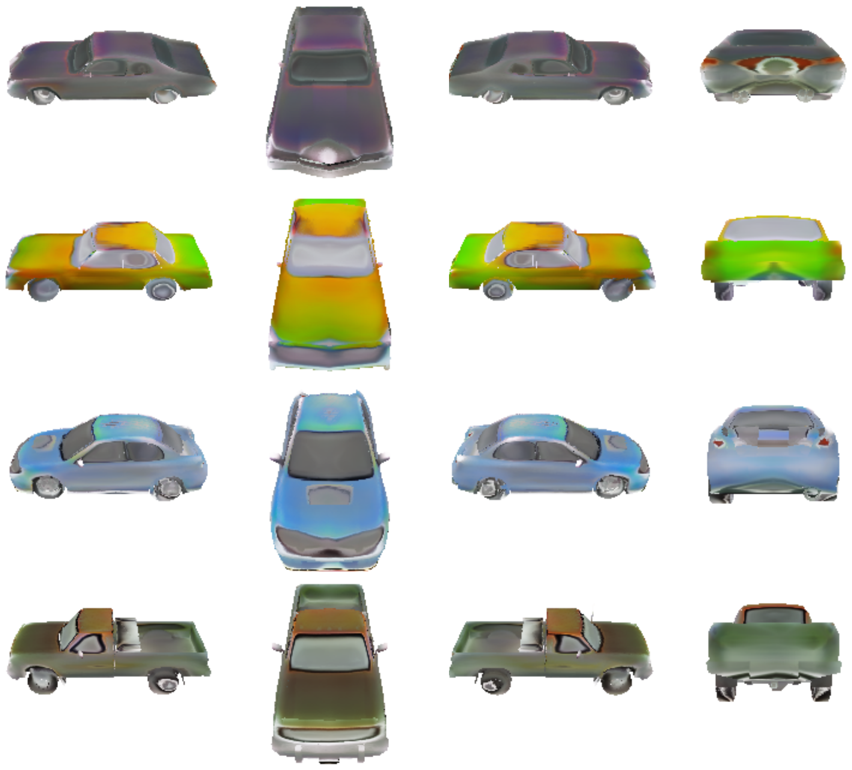

Figure 20: **GAN.** In this figure, we show 3D CAD objects with generated texture using our GAN-based model. Our GAN predictions exhibits GAN typical artifacts.

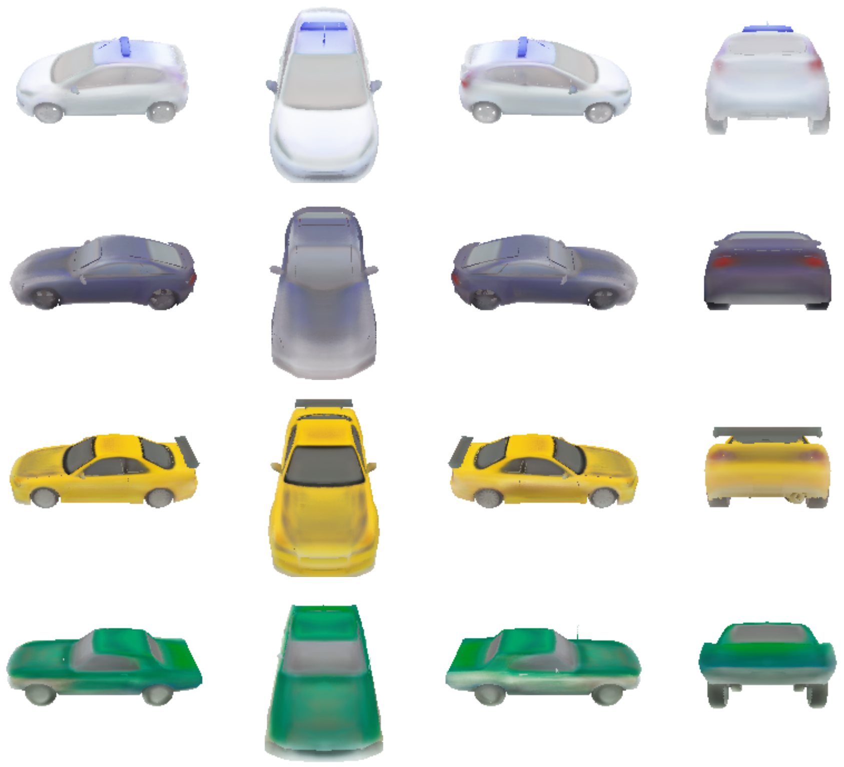

Figure 21: **VAE.** This figure illustrates predicted textures using the VAE model. The results are globally consistent, but exhibits blur in some cases.

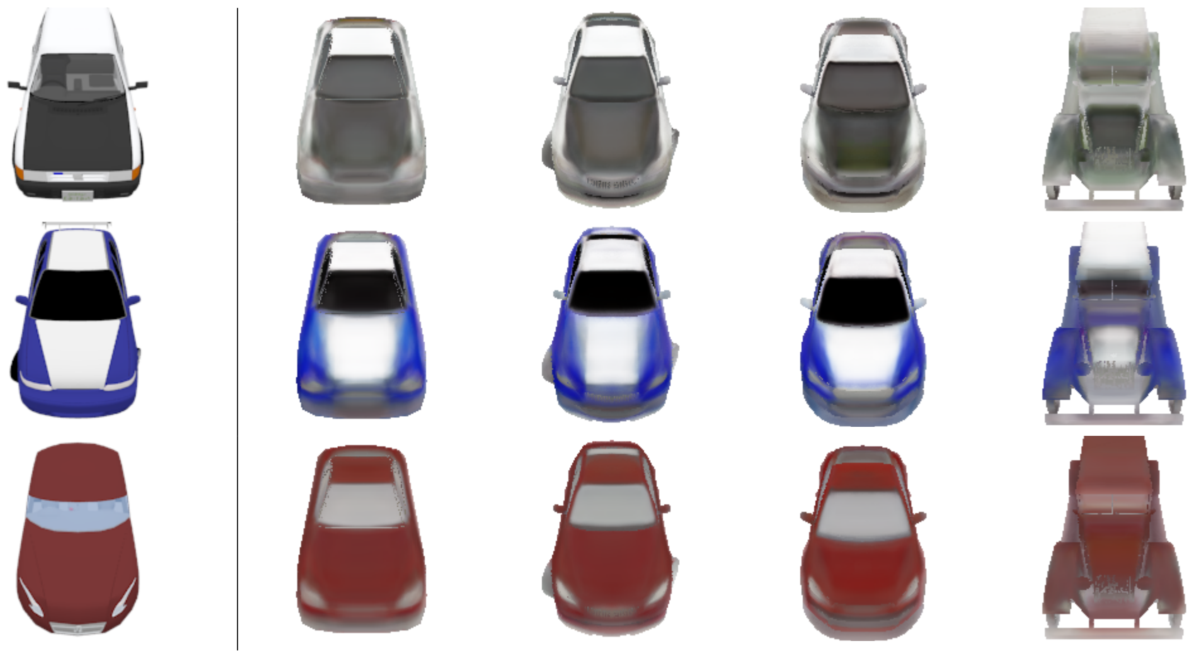

Figure 22: **Texture Transfer.** We utilize the VAE model for texture transfer from one car to another. We encode the image on the left and use the latent code for synthesizing texture for different shapes.

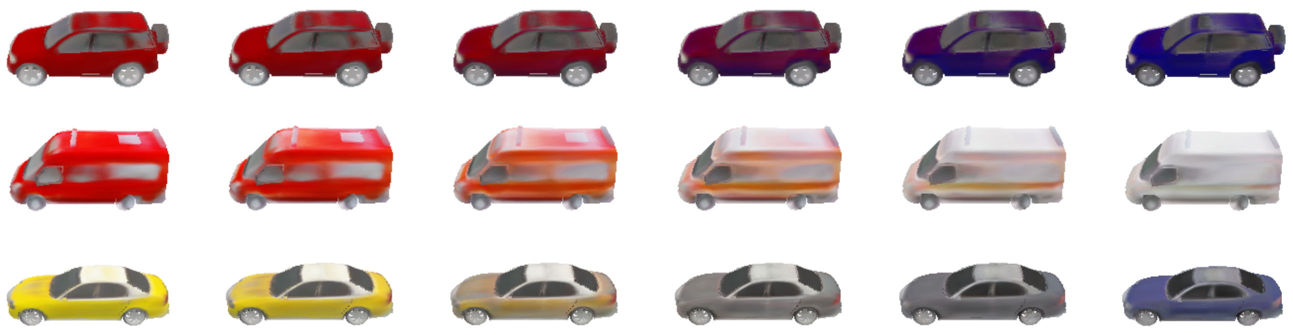

Figure 23: **Latent Space Interpolations.** Here, we illustrate latent space interpolations. The results show that our model learns a continuous and meaningful latent space.

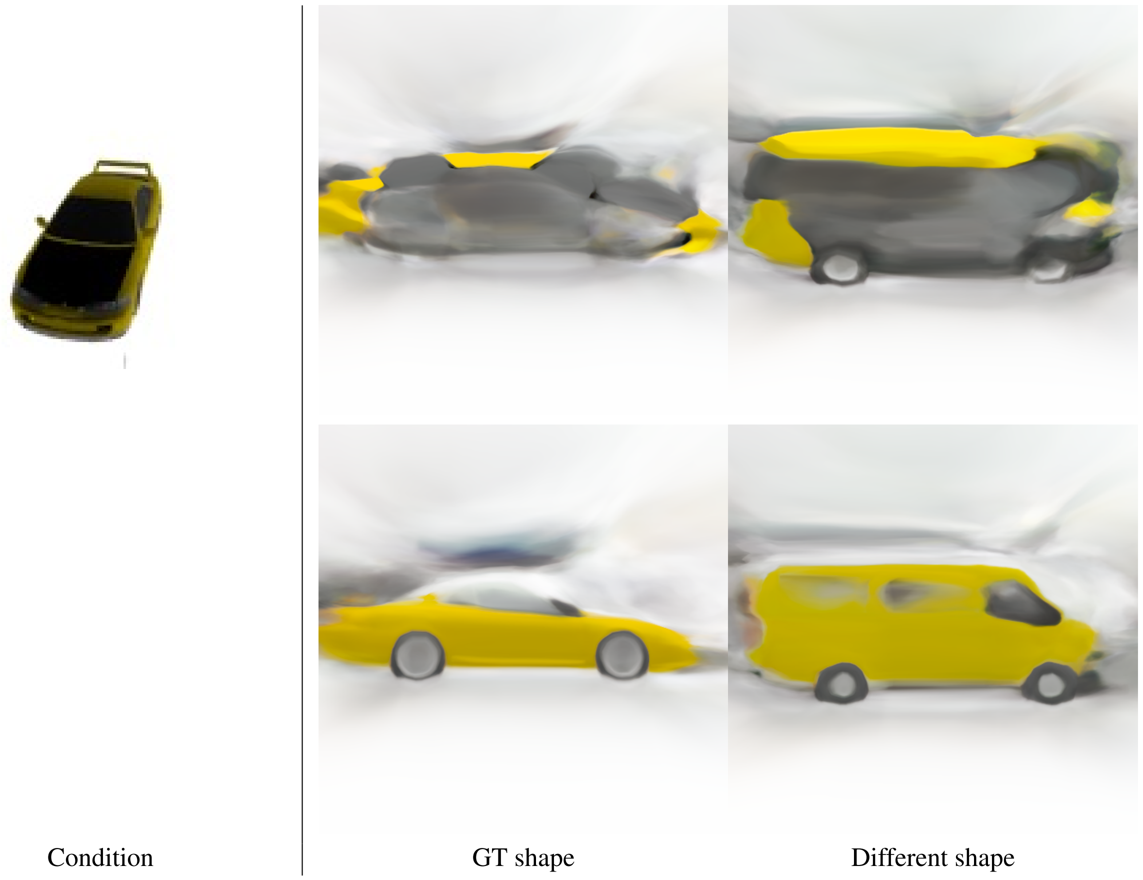

Figure 24: **Texture Field Illustrations.** In this figure, we show predicted color values along cuts through the car models. The image condition is shown on the left and on the right two different results for cuts are depicted. In the top row a cut through the middle of the cars is shown, whereas in 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.