

TM-NET: Deep Generative Networks for Textured Meshes

LIN GAO, ICT, CAS and UCAS
 TONG WU, ICT, CAS and UCAS
 YU-JIE YUAN, ICT, CAS and UCAS
 MING-XIAN LIN, ICT, CAS and UCAS
 YU-KUN LAI, Cardiff University
 HAO ZHANG, Simon Fraser University

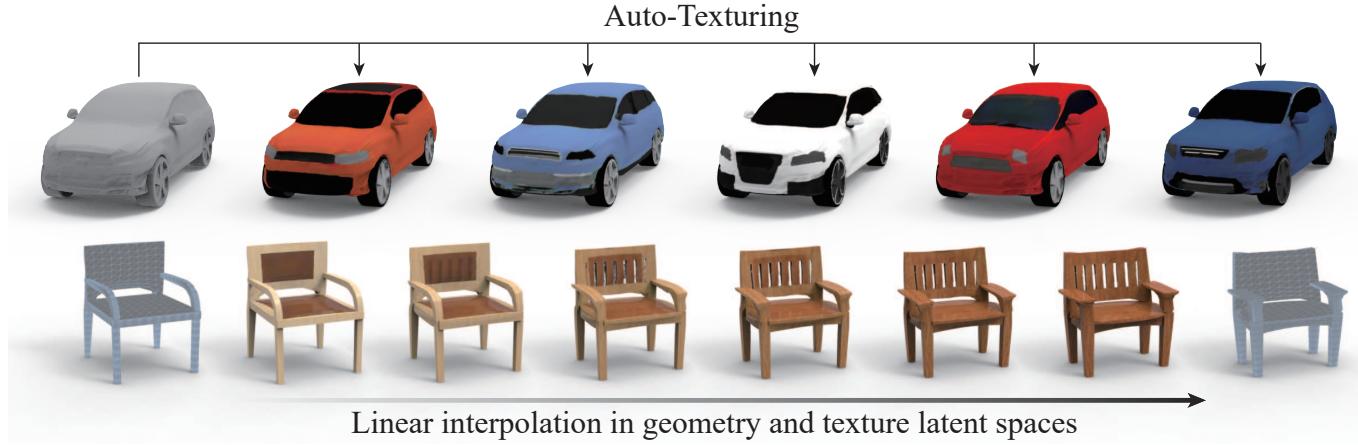


Fig. 1. Our deep generative network for textured meshes, TM-NET, can automatically synthesize *multiple* textures for the same input 3D shape (top) and blend between textured meshes via latent space interpolation (bottom). The chair meshes have a relatively low resolution (less than 4,000 vertices) but the generated textures exhibit the *appearance* of topological details on the chair backs.

We introduce TM-NET, a novel deep generative model capable of generating meshes with detailed textures, as well as synthesizing plausible textures for a given shape. To cope with complex geometry and structure, inspired by the recently proposed SDM-NET, our method produces texture maps for individual parts, each as a deformed box, which further leads to a natural UV map with minimum distortions. To provide a generic framework for different application scenarios, we encode geometry and texture separately and learn the texture probability distribution conditioned on the geometry. We address challenges for textured mesh generation by sampling textures on the conditional probability distribution. Textures also often contain high-frequency details (e.g. wooden texture), and we encode them effectively with a variational autoencoder (VAE) using dictionary-based vector quantization. We also exploit the transparency in the texture as an effective approach to modeling highly complicated topology and geometry.

This work is the first to synthesize high-quality textured meshes for shapes with complex structures. Extensive experiments show that our method produces high-quality textures, and avoids the inconsistency issue common for novel view synthesis methods where textured shapes from different views are generated separately.

CCS Concepts: • Computing methodologies → Shape modeling.

Additional Key Words and Phrases: Shape representation, variational autoencoder, texture, 3D reconstruction, generation

This is the author's version of the work. It is posted here for your personal use. Not for redistribution.

1 INTRODUCTION

With rapid advances in deep learning, many recent works propose deep neural networks for learning 3D shape representations [Chen et al. 2020; Chen and Zhang 2019; Gao et al. 2019; Groueix et al. 2018; Li et al. 2017; Mescheder et al. 2019; Mo et al. 2019a; Park et al. 2019; Wang et al. 2018a; Wu et al. 2019], generations [Chen and Zhang 2019; Gao et al. 2019; Li et al. 2017; Wu et al. 2016, 2019], and reconstructions from different sources such as images [Richter and Roth 2018; Wang et al. 2018b], point clouds [Groueix et al. 2018; Park et al. 2019], and sketches [Han et al. 2017; Lun et al. 2017]. However, current state-of-the-art methods still cannot produce 3D shapes with high visual quality, in terms of surface details and the realism and richness of the shapes' appearance. The generation and reconstruction of geometric and topological details are especially difficult, due to the associated data and computational challenges in training deep models for high-resolution geometries.

In the absence of geometric details, a classical graphics "trick" is to create the *appearance* of surface richness by simulating details in the *image space*, e.g., via texture or bump mapping. As shown in Figure 1, textured meshes can exhibit the kind of realism and visual fidelity that geometry alone cannot convey, even when the underlying mesh resolution is fairly low. However, despite the flourishing of geometric deep learning, there have been few attempts

at developing a deep generative model for textured meshes. To the best of our knowledge, existing works on textured shape generation have predominantly relied on single- or multi-view image guidance [Chen et al. 2019; Liu et al. 2019; Raj et al. 2019].

In this paper, we introduce TM-NET, a deep generative network for *textured meshes*, aiming to fill several current gaps in learning-based realistic 3D shape modeling. First, we seek a *generic* decoder which can generate textured meshes with or without image guidance. Second, the network should be able to generate novel textured meshes and predict textures for a given shape. Last but not least, the generative model should be *part-based* to allow different shape parts to be textured differently, which is naturally expected for real-world models.

With these goals in mind, our design for TM-NET draws inspiration from SDM-NET [Gao et al. 2019], which is a deep generative network for *structured deformable meshes*. Specifically, SDM-NET generates meshes formed by parts, where each part is homeomorphic to a cuboid box and finer-scale geometry of the part can be controlled by deforming the template box. Architecturally, the network consists of a part-level variational autoencoder (VAE) for learning genus-zero mesh parts and a structure-part VAE (SP-VAE), which jointly learns part structures and part geometries from a shape collection.

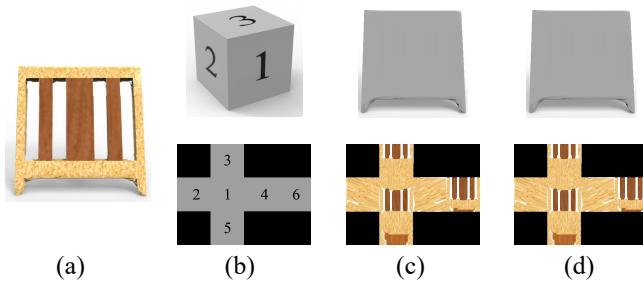


Fig. 2. Texture representation with deformed boxes: (a) input model; (b) template box and unfolded UV map where numbers on the box faces help illustrate the unfolding process; (c) geometry and texture representations with the deformable box; (d) decoded geometry and texture.

SDM-NET provides a fitting entry point for us to develop a textured mesh generator as we desire, since it is part-based via deformable box templates. This allows both geometry and texture to be represented in a well-aligned canonical form: the rich geometry details are approximated and represented by deformations enabled by the template boxes, and these boxes work naturally with the UV-maps for texture mapping. Specifically, we can cut designated edges of a template box and flatten the box by unfolding it onto the UV space that contains texture information; see Fig. 2, for example. The collection of deformable boxes, when mapped to the parameter space, forms a consistent texture atlas even for shapes with substantial structure and/or geometry variations, making effective learning of structured and textured shapes possible. However, to go from SDM-NET to a generic TM-NET capable of generating high-quality textured meshes, we must address several new challenges:

- In general, the mapping between shape parts and textures is far from one-to-one — the same shape part may possess drastically different textures and vice versa. Hence, a joint learning of part geometry and part texture, which resembles the SP-VAE in SDM-NET, would be quite limiting.
- If we were to develop separate part geometry and part texture VAEs, we must ensure global consistency between the part textures within a shape without joint learning. So we specify a central part, and other parts' texture generation is conditional on the central part texture's VGG [Simonyan and Zisserman 2014] feature.
- It is well-known that images generated by VAEs tend to be blurry [Dosovitskiy and Brox 2016], which is counter to our goal of generating textures with high-frequency details.

Figure 3 shows the key components of TM-NET, with the dataset and training processes detailed in Section 4. Like SDM-NET, our model also learns a structured mesh representation, where a textured mesh is represented at both the part level and the overall shape level using VAEs. However, to address the new challenges outlined above, several key changes to the network architectures must be developed. First, at the part level, TM-NET *separately* embeds part geometry (via the PartVAE) and part texture (via the TextureVAE) into their respective latent spaces (see Figure 4), as a means to facilitate learning the texture probability distribution conditional on geometry.

We further introduce conditional autoregressive generative model similar to PixelSNAIL [Chen et al. 2018], which resolve ambiguities of different textures on similar or identical geometry. However, individually generated textures for different shape parts may be inconsistent. Hence, we specify a central part for each category, we train the central part PixelSNAIL conditional on the central part' geometry and for other parts' conditional PixelSNAIL networks, the conditional input is the concatenation of other parts' geometry and the VGG feature of the central part's texture. The additional VGG feature helps to establish the relationship between the central part and other parts.

Furthermore, since part geometry and texture possess different data characteristics, e.g., smooth geometry vs. high-frequency image details, the two part-level VAEs have different architectures. In particular, to address the blurry image issue with conventional VAEs, we adapt VQ-VAE [Razavi et al. 2019a,b] for learning and generating textured mesh parts; see Figure 7. With the aid of vector quantization (VQ), our VQ-VAE allows the learning of a high-dimensional texture latent space to facilitate the generation of quality textures, with high-frequency details and avoiding blurriness.

New textured meshes can be generated by working in the latent spaces of the VAEs, via sampling or linear interpolation. The network can also be trained to synthesize textures for a given shape and perform single-view 3D reconstruction with textures.

Our main contributions can be summarized as follows:

- To the best of our knowledge, TM-NET is the first generative model capable of synthesizing high-quality textured meshes in a part-aware manner, with or without image guidance.
- Our method supports several generative tasks to serve typical applications involving textured meshes.

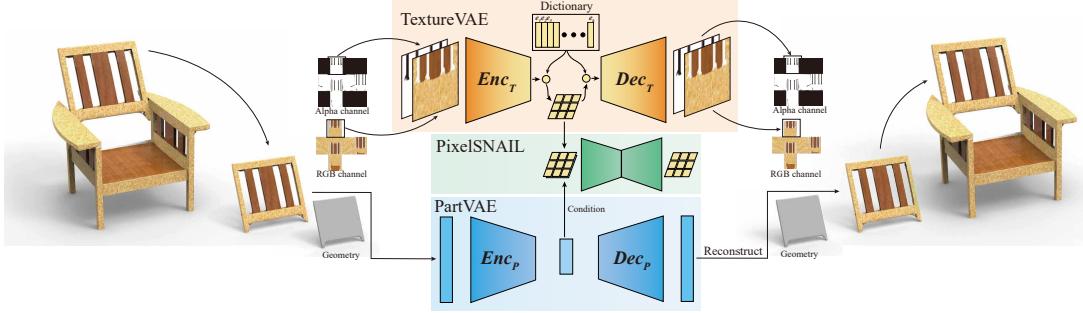


Fig. 3. An overview of the key components of TM-NET. Each part is encoded using two Variational Autoencoders (VAEs): PartVAE for geometry with Enc_p as the encoder and Dec_p as the decoder, and TextureVAE for texture with Enc_T as the encoder and Dec_T as the decoder. To make geometry guided texture generation possible, the conditional autoregressive generative model PixelSNAIL[Chen et al. 2018] is adopted here, which takes the latent vector of PartVAE as condition input and outputs discrete featuremaps that are to be decoded as texture images for the input geometry.

- By utilizing texture *transparency* via the alpha channel, our method is able to generate and reconstruct the *visual appearance* of topological details such as holes in the chair back; see Figure 1. Visually, the obtained results are far superior than those attainable by state-of-the-art decoders and generative networks; they are almost visually indistinguishable from renderings of shapes with the actual holes; see Figure 9

We show the generative capabilities of TM-NET under several application scenarios. Quality of the textured meshes generated by our network is demonstrated both qualitatively and quantitatively. Evaluation is conducted via ablation studies and comparisons to state-of-the-art 3D generative models, including SDM-NET.

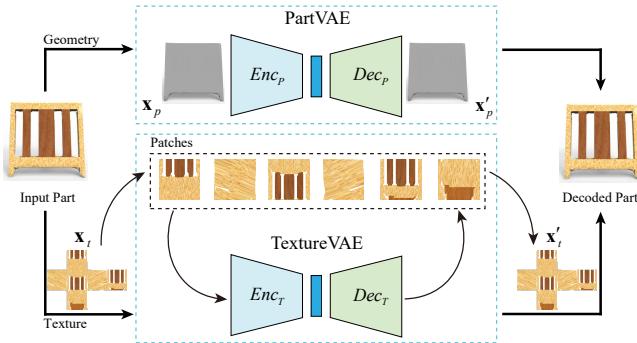


Fig. 4. Our architecture for representing a textured part, which involves a PartVAE for encoding the geometry and a TextureVAE for encoding the texture.

2 RELATED WORK

We review papers most related to our work, namely deep generative models for 3D shapes, neural texture generation, joint 2D-3D encoding, and single-image shape reconstruction.

Deep Generative Models for 3D Shapes. With the development of the deep neural networks such as Variational Auto-encoders (VAEs) [Kingma and Welling 2013] and Generative Adversarial

Networks (GANs) [Goodfellow et al. 2014], novel 3D shapes can be effectively generated by learning the distributions of existing data. For generating 3D shapes, existing deep generative models produce shapes in different representations, including voxels [Wu et al. 2016], point clouds [Achlioptas et al. 2018], implicit functions [Chen and Zhang 2019; Mescheder et al. 2019; Park et al. 2019], deformable meshes [Tan et al. 2018], multi-chart representations e.g. AtlasNet [Groueix et al. 2018] and structured representations, e.g. GRASS [Li et al. 2017], StructureNet [Mo et al. 2019a] and SDM-NET [Gao et al. 2019]. Recent methods with improved shape representations allow 3D shapes with rich geometric details and complex structures to be synthesized. However, none of these methods generate 3D models with textures. Furthermore, textures are crucial for the visual realism and rich appearance of 3D shapes. In this paper, we build our texture generation method on SDM-NET, a recent shape generation method based on structured deformable meshes, where each part is represented by using deformations from a box. The method produces shapes with fine details.

Texture synthesis. Texture synthesis for 2D images [Efros and Leung 1999] and 3D solids [Kopf et al. 2007] have been widely studied in computer graphics. In recent years, neural network based methods have been proposed [Gatys et al. 2015; Snelgrove 2017; Zhou et al. 2018] for texture synthesis, leading to improved results, although these methods are still restricted to image textures, rather than textures over surfaces.

For surface texture generation, the texture representation strongly relies on the underlying geometry representation. Although point clouds are flexible, and suitable for applications such as segmentation and classification [Qi et al. 2017a,b], they are too sparse to represent geometry details [Hu et al. 2019], and also suffer from this major issue when used for representing high-resolution textures. For the voxel-based representation, the texture representation based on voxels [Tulsiani et al. 2017] is also limited by the resolution due to the high memory overhead.

Recently, some works propose to generate textures using deep generative models. According to the statistical properties, textures on 3D shapes can be divided into two categories, namely stochastic

textures such as wood and marble which can often be compactly modeled using a procedural generation process or some samples, and non-stochastic textures which are generic images on surfaces. Henzler et al. [2019] propose a method to encode stochastic textures from 2D exemplars for synthesizing 3D solid textures. While solid textures are useful for certain applications, they are not associated with shape surfaces. The method is also restricted to stochastic textures, which are insufficient for texturing general 3D objects.

Generating textured 3D Shapes. To obtain textured 3D shapes, some works use single-view color images as 2D supervision to recover 3D objects with colors. Hu et al. [2019] use a two-stage method that first infers the object coordinate map and then uses reprojection onto the input image to recover dense point clouds with colors. The method is able to handle input images of different (and even unseen) object categories; however, it only generates a partial point cloud with visible colored points for a given input image, and multi-view fusion is required to obtain complete shapes. Other methods [Chen et al. 2019; Liu et al. 2019; Pavllo et al. 2020] rely on differentiable renderers to train networks to predict the vertex position and associated color of a mesh with fixed topology. These methods largely rely on the input single-view image to provide a strong constraint, whereas our approach aims to generate textures without image guidance. Moreover, the use of a fixed-topology shape such as a sphere as the starting point also restricts their methods to handle shapes with complicated structures and/or rich geometric details.

A fundamental challenge for texturing 3D shapes is an effective texture mapping from the 3D surface to some texture (UV) space. UV mapping is usually difficult to be consistently defined across shape collections, and previous UV-mapping based methods mainly deal with models with fixed topology including planar topology (for faces) [Saito et al. 2017] and spherical topology (for birds) [Kanazawa et al. 2018]. These representations however cannot be generalized to cope with shapes with complex structures. Recent work Texture Fields [Oechsle et al. 2019] avoids this problem by learning textures using mapping from the 3D texture space. However, the use of 3D solid space is not only expensive, but also harder to learn, as textures of interest are usually only shape surfaces. Raj et al. [2019] propose a two-stage approach to generating textured meshes, where in the first stage, a network is trained to map 2.5D depth images to texture images, and in the second stage, multiview texture images are fused. However, the method does not give details how to ensure seamless synthesis of texture atlas, and how to ensure sufficient coverage of texture images, especially for shapes with complex structures where self-occlusion is a problem, and shows examples of a single category (cars) which are of spherical topology. Unlike all these existing methods, our approach is able to generate high-quality textured meshes with complex structures. Our method can also generate geometry and texture simultaneously, which cannot be achieved with existing methods.

Novel view synthesis (NVS). Given a single image, humans can clearly imagine how the object looks like from another viewpoint. Novel view synthesis works focus on generating images from an unknown viewpoint with an input image. [Sun et al. 2018] takes multiple images from different source viewpoints and predicts a flow

field to move the pixels from the source image to the target image. Then it aggregates predictions from different viewpoints to produce the target image. SRN [Sitzmann et al. 2019] proposes a continuous 3D-structure-aware 3D scene representation that maps the world coordinates into a feature representation which helps preserve the geometry surface from novel viewpoints. GeLaTO [Martin-Brualla et al. 2020] moves a step forward to generate the rendered images of transparent objects from a new viewpoint and interpolate rendered images from different viewpoints with U-NET [Ronneberger et al. 2015] and AtlasNet [Groueix et al. 2018]. However, their intermediate texture is an implicit representation. Thus it can not be directly put on 3D shapes for other traditional graphical purposes. Although textured meshes can be rendered from novel viewpoints, our method is fundamentally different from novel view synthesis (NVS) methods, which have been an active research topic in recent years: not only does our method not require a single-view image as input (and can generate textured meshes directly), our method also ensures the rendered images from different views are consistent, which cannot be guaranteed with NVS methods.

3 METHODOLOGY

Given a collection of textured 3D mesh models, our TM-NET encodes the geometry associated with textures for generation and other applications such as interpolation and automatic texturing. As discussed before, we build on recent SDM-NET [Gao et al. 2019] which provides structural deformable boxes to represent 3D objects, each corresponding to a semantic part. Thanks to this representation, it intrinsically provides a way to define consistent texture atlases for objects, even if shapes in the collection may not have identical topology (e.g. a table may have different numbers of legs). Each box can be easily unfolded to the texture domain, with no extra distortion at this stage. In the following subsections, we will describe our texture representation and how to convert textured 3D models to this representation. We then describe the details of our novel network architecture, i.e. how the geometry detail determines its texture, first at the part level and finally the object level.

The geometry and structure are learned jointly in one global VAE to ensure generated textured meshes are consistent in both geometry and structure. A single part can be textured very differently while still being plausible. Instead of generating textures totally dependant on the input geometry, we learn the probability distribution of textures conditional on geometry similar to PixelSNAIL [Chen et al. 2018]. Once the distribution is learned, we can sample several times on the distribution which will lead to multiple reasonable textures for the input shape. However, directly utilizing this at the part level cannot ensure the compatibility of textures between different parts on the same shape. So for each category, we appoint a part as the central part (e.g. table surface for table). While modelling other parts’ texture distribution, we append the central part’s texture’s VGG feature to the geometry condition. So the central part’s texture generation is totally conditional on the central part geometry, while other parts’ texture generation depends on the geometry and the central part’s texture feature to make sure the texture generation for different parts coherent with the central part.

3.1 Part based Texture Representation

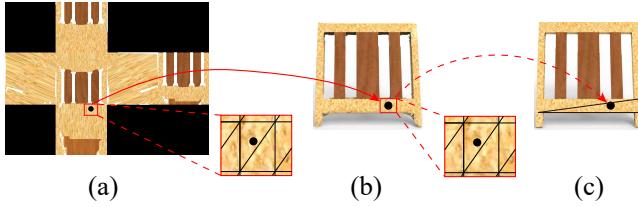


Fig. 5. Illustration of the procedure that maps the ground truth texture to the unfolded box to form a texture image. (a) the texture image to be filled, (b) the deformed box that approximates the input model part, (c) the input model containing ground truth texture.

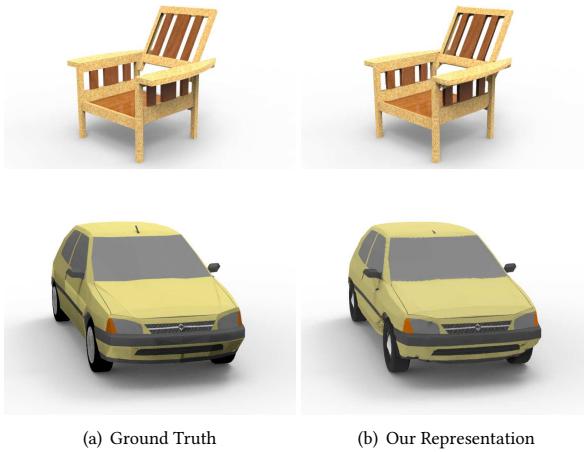


Fig. 6. Some examples demonstrating our part-based texture representation, (a) ground truth shapes with texture, (b) shapes with texture in our representation. Our method is able to retain most fine details, and visually reproduces rich structures that cannot be represented using underlying the geometric representation.

To represent the texture on meshes, a key step is to build texture atlases for meshes. Traditionally, the texture atlas is generated individually for each mesh. However, this makes learning difficult. Our method is based on SDM-NET [Gao et al. 2019], which provides a set of consistently segmented parts across each category. The geometry details of each part are represented by the deformations of a template box. In our setting, template boxes are also served as the domains for surface textures with predefined UV mapping. To minimize the mapping distortion from the template box to the 2D texture maps, the template box is unfolded in a straightforward manner as illustrated in Fig. 2 (b). Let $I_{i,j}$ be the texture image of the j^{th} part on the i^{th} shape. By the unfolding operation, the template box box_0 is unfolded and parameterized to the 2D mesh in the UV space to obtain \tilde{box}_0 as shown in Fig. 2, with texture image size $4l \times 3l$ where l is the length of the edge of the template box when mapped to the UV space. $l=256$ is fixed for all our experiments. Although larger l leads to higher resolution texture images that

better represent texture details, the network also requires more data for training. Our setting reaches a good balance. This defines the UV mapping between the deformed box $b'_{i,j}$ and the corresponding texture image $I_{i,j}$.

The next step is to fill in each pixel \tilde{u} on the texture image $I_{i,j}$. As illustrated in Fig. 5, the 2D mesh \tilde{box}_0 gives a partition of the texture image into triangles. Assuming that \tilde{u} belongs to a 2D triangle \tilde{f} on \tilde{box}_0 with barycentric coordinates $bc(\tilde{u})$. We can find the corresponding 3D face f' located on the deformed box $b'_{i,j}$, as well as the corresponding 3D position p' based on barycentric coordinates $bc(\tilde{u})$ on f' . We then project p' onto the input shape to obtain the ground truth color for the pixel \tilde{u} .

Note that a genus-zero box cannot represent a high-genus part barely through deformation. We make use of the texture transparency (the alpha channel) to represent a high-genus part. If the projection of p' is not on the surface of the input shape, then the alpha channel for the pixel \tilde{u} is set to 0 representing total transparency, otherwise 1 representing opaque. With such a setting, textures make up for topology loss in the part-based geometric representation. Some examples are shown in Fig. 6. Parts such as the chair back contain high-genus structures which cannot be represented using SDM-NET [Gao et al. 2019]. Our texture-based solution produces a visually faithful representation, indistinguishable from the rendering of the ground truth shapes.

3.2 Part-level Texture and Geometry Encoding

We first present how the geometry and texture are encoded for a given object part. For the geometry, we use the mesh-based variational autoencoder (PartVAE) in SDM-NET [Gao et al. 2019] to encode the geometric details. Please refer to [Gao et al. 2019] for its network structure. As for the texture, we introduce a TextureVAE for encoding texture images. Different networks are needed to address their fundamentally different characteristics.

Texture often contains high-frequency details but shares similar patterns, which is difficult to effectively encode using traditional VAEs. Our TextureVAE is inspired by VQ (Vector Quantization)-VAE [Razavi et al. 2019a,b] and its architecture is illustrated in Fig. 7. The input of TextureVAE is a four-channel texture patch in size 256×256 , including three channels of RGB values and one channel of alpha values.

The encoder demonstrated in VQ-VAE2 [Razavi et al. 2019a,b] capable of encoding 256×256 image is a two-level image pyramid extracting two feature maps, referred to as top, and bottom level feature maps z_e^* , where the superscript * can be t or b , indicating top and bottom levels. For simplicity, we use this notation in the following description when appropriate. Each feature map z_e^* goes through a vector quantization operation which maps each feature vector to its closest vector in the dictionary e^* to obtain the discretized feature map z_q^* . The dictionary is updated during the training process as in [Razavi et al. 2019a,b], to capture the data distribution. The decoder takes the discrete feature maps as input to produce the reconstructed image. For more information about vector quantization and discrete vectors for quantization, please refer to [Razavi et al. 2019a,b]. This architecture captures multiscale texture characteristics and allows high-frequency details often present in textures to be

faithfully represented using a low-dimensional latent vector, thanks to the dictionary-based vector quantization and low-dimensional embedding.

With PartVAE encoding the geometry and TextureVAE encoding the texture, our method can represent meshes with fine geometry and fine textures. Moreover, our texture contains transparency information so the texture is also leveraged for its geometric representation ability, enabling our method to represent a single part with many holes on it.

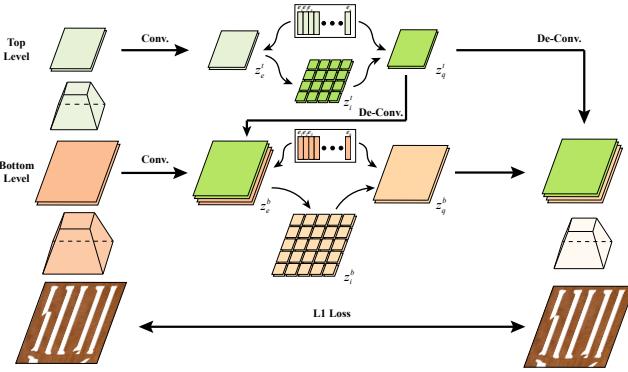


Fig. 7. The architecture of TextureVAE. The encoder maps the input image patch onto two continuous feature maps. Then the dictionary-based vector quantization is performed to make these feature maps discrete. The decoder takes the discrete feature maps as input and reconstructs the image.

Since geometry and texture distributions lie in their own manifold spaces, PartVAE and TextureVAE are trained separately. Let $Enc_P(\cdot)$ and $Dec_P(\cdot)$ be the encoder and decoder of part deformations for representing geometric details.

$Enc_T(\cdot)$ and $Dec_T(\cdot)$ denote the encoder and decoder for the associated texture images, x_p and x_t are the input deformation representation and texture image respectively, $z_p = Enc_P(x_p)$ and $z_t = Enc_T(x_t)$ are the encoded latent vectors of geometry and texture, respectively, where $z_t = (z_q^t, z_q^b)$ is the concatenation of discrete featuremaps from the VAEs of the two layers. $x'_p = Dec_P(z_p)$ and $x'_t = Dec_T(z_t)$ are the reconstructed deformation feature vector and texture image.

The loss function of PartVAE follows [Gao et al. 2019]. Please refer to [Gao et al. 2019] for more detail. TextureVAE minimizes the following loss:

$$L_{TextureVAE} = L_{recon} + L_{seam} + \alpha_1 L_{embedding} \quad (1)$$

Here, $L_{recon} = \|x'_t - x_t\|_1$ is the reconstruction loss to ensure faithful reconstruction. L_{seam} is used to penalize the differences between texture boundary pixels that are in the same template box edge before unfolding to promote synthesizing seamless textures.

$L_{embedding}$ is the embedding loss, which is defined as:

$$L_{embedding} = \sum_{e \in \{t, b\}} \|sg[z_e^*] - e^*\|_2^2 + \beta_1 \|sg[e^*] - z_e^*\|_2^2 \quad (2)$$

where sg stands for stopgradient operator which does not influence the forward pass computation but has zero partial derivatives in

the backward pass. The first term aims at updating the embedding space, i.e., making the dictionary e^* closer to the feature maps z_e^* extracted by the encoder. The second term updates the encoder's parameters to make the feature map move towards the embedding space or the dictionary e^* . Please refer to [Razavi et al. 2019a,b] for more detail.

L_{reg} is an l_2 norm regularization term of the trainable parameters in the network to avoid overfitting. α_1 and β_1 control the relative importance of each loss term.

3.3 Geometry Guided Texture Generation

The geometry and texture can be encoded by PartVAE and TextureVAE, respectively. In many cases, there are strong correlations between the geometry and the texture, e.g., in the front of the car are usually the beams with white color and tires are usually black, while in other cases, the same geometry may be textured in different, plausible ways, e.g. the car body may be painted with different colors even with nearly identical geometry.

With these observations, we can see the texture generative model should generate different textures for a single input shape. Autoregressive models predict the conditional probability distribution of each pixel in the texture image, and sample different textures from the predicted distribution. The autoregressive generative model PixelSNAIL [Chen et al. 2018] has been successful in discrete feature map generation. Our generative network is inspired by PixelSNAIL [Chen et al. 2018]. As Fig. 8 shows, two different conditional PixelSNAIL networks learn the high-dimensional distributions of the two discrete index matrices z_i^t and z_i^b . It should be clarified that we learn the whole texture image's data distribution instead of the single texture patch. That is to say we concatenate the discrete index matrices of different patches in a texture image together and feed it into the PixelSNAIL network. The top level PixelSNAIL takes the top level index matrix z_i^t as input and the geometry latent vector z_p (for the central part) or concatenation of geometry latent vector z_p and the central part texture's VGG feature (for other parts) as condition, while the bottom level PixelSNAIL takes the bottom level discrete index matrix z_i^b as input and the top level discrete index matrix z_i^t as conditional. Let x and c denote the input and the condition of the PixelSNAIL network f . In the training process, the PixelSNAIL network minimizes following loss:

$$L_{PixelSNAIL} = L_{CrossEntropy}(x, f(x, c)) \quad (3)$$

We use SP-VAE from SDM-NET [Gao et al. 2019] to jointly encode the geometry and structure of different parts to represent the whole shape.

3.4 Applications

We now present how our method can be applied to various applications with textured meshes.

Automatic Texture Generation for Given Shapes. Our network can be used to generate texture for a given shape automatically. Given a new shape, we first convert it into our representation. Then we can obtain the geometry latent vector of each part in the latent space using PartVAE. The autoregressive generative model PixelSNAIL

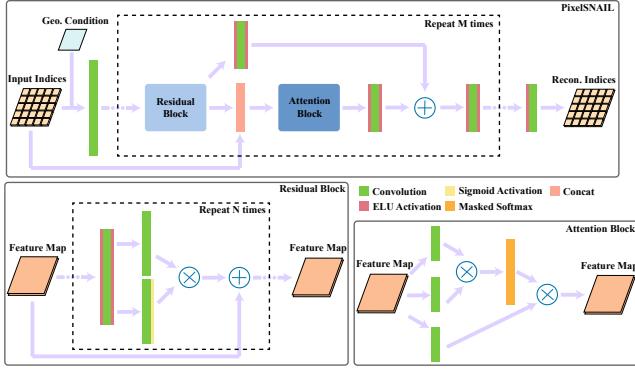


Fig. 8. The architecture of auto-regressive generative model PixelSNAIL [Chen et al. 2018].

for the central part models the distribution of the two-level index matrices. Then we sample on the distribution for several times to generate different index matrices z_i^t and z_i^b , which are later used as index to search corresponding code vector in the codebook to form discrete featuremaps z_q^t and z_q^b . Finally, the decoder of TextureVAE decodes z_q^t and z_q^b to generate textures. Once the texture of central part is generated, we extract the central part texture's VGG feature and concatenate it with other parts' geometry as condition for other parts' autoregressive models, which will lead to coherent texture for different parts. This supports automatic texture generation for given shapes but also allows generating diverse plausible textures.

Generation and Interpolation. Given our deep generative model TM-NET, we can randomly generate textured mesh models and interpolate mesh models in the latent space.

In the textured mesh generation task, we first generate the structured meshes with fine geometry by randomly sampling from the SP-VAE latent space. For each part we encode the geometry into the latent space of PartVAE. Then the problem is converted to automatically generating textures for the generated shape as mentioned above.

For interpolation, the geometry and structure of two input shapes are interpolated in the latent space of SP-VAE, and the textures are interpolated in the continuous feature maps z_e^* of TextureVAE for generating in-between textured meshes.

4 DATASETS AND NETWORK IMPLEMENTATION

In this section, we give the detailed network architecture and the training process. Our experiments were performed on a computer with an i7 6850K CPU, 64GB RAM, and 12 RTX 2080Ti GPU.

4.1 Dataset

[Gao et al. 2019] provides part segmentation for objects in the subset from ShapeNet Core V2 [Chang et al. 2015]. We select mesh models with both segmentation and texture as the dataset in this paper. In addition, we segment the recently released dataset 3D-Future [Fu et al. 2020]. The number of textured meshes used in our experiment is shown in Table 1.

Table 1. The numbers of segmented textured meshes in our experiments.

category	Car	Chair	Plane	Table
# Segmented Meshes with Texture	1824	6635	2690	7209

4.2 Network Architecture

Our network is composed of four parts including PartVAE for encoding part geometry, TextureVAE for encoding part texture, conditional PixelSNAIL for generating textures, and SP-VAE for jointly encoding both global structure and part geometries. The architectures of PartVAE and SP-VAE follow [Gao et al. 2019]. And as shown in Fig. 7, the encoder of TextureVAE has three convolutional blocks, each followed by one convolution layer to obtain two different levels of continuous feature representations, which then go through vector quantization. The decoder concatenates the discrete latent variables from different levels and outputs the reconstructed texture images through deconvolution blocks. The detailed architecture of encoding and decoding networks can be found in the supplementary material. For the hyperparameters in the loss function of TextureVAE, they are the same as those in [Razavi et al. 2019a]. Further, the conditional texture generation network architecture is same as the original implementation, except that before putting conditional input into PixelSNAIL, we add three fully connected layers to convert the conditional input into the same size of the input discrete index matrix.

The network architecture of SP-VAE follows the SP-VAE in [Gao et al. 2019]. Leaky ReLU is set as the activation function. We first train PartVAE and TextureVAE. After that, we train the conditional PixelSNAIL network. SP-VAE is trained at last.

4.3 Metrics for Evaluation

For evaluation, we use different evaluation metrics in the geometry space and image space. Firstly, we use the Fréchet Inception Distance (FID) [Heusel et al. 2017], which is a widely used metric to measure the distances between different distributions. Learned Perceptual Image Patch Similarity (LPIPS) [Zhang et al. 2018] is used to measure the similarity between the decoded texture image and the input texture image. Moreover, since the texture and geometry are interconnected, we can evaluate the quality of geometry and texture together by rendering the textured shapes. So we render the decoded textured meshes and the input textured meshes in 12 viewpoints and use Structure Similarity Image Metric (SSIM) [Zhou et al. 2004] to measure the difference of rendered images in the same viewpoint and the same lighting. Secondly, to evaluate the realism of generated textures for a given shape, we use fooling rate which is a human judgement approach proposed in [Zhang et al. 2016]. This test shows a real image and a synthesized image to a human participant for one second each and asks them to tell which is ‘real’. The fooling rate is the possibility of synthesized images being mistaken for real images.

4.4 Parameters

Hyper-parameters are fixed for datasets of different object categories. We now demonstrate how the results change while changing hyper-parameters. Each dataset is randomly divided into the training set and test set using the data released from SDM-NET with a ratio of 9 : 1. We use Structural Similarity Index (SSIM) (see more in Sec. 5) to measure image perceptual similarity between rendered images of input textured meshes and those of decoded textured meshes. α_1 and β_1 are set to 1 and 0.25 following VQ-VAE[Razavi et al. 2019a,b].

4.5 Training Details

Our network is trained stage by stage. On the geometry side, similar to SDM-NET, every semantic part has a corresponding PartVAE. All of these PartVAEs are trained separately. After training the PartVAEs, we then concatenate the latent vector of PartVAE and the structure code as the input to train SP-VAE. The PartVAE is trained with 20,000 iterations and SP-VAE with 120,000 iterations with batch size 128. On the texture side, a single TextureVAE is trained for textures of a single part. The TextureVAE is trained with 400,000 iterations with batch size set as 10. After the training process of SP-VAE and TextureVAE are finished, we train the top and bottom level PixelSNAILs. The top level PixelSNAIL is trained with 200,000 iterations and bottom level PixelSNAIL is trained with 300,000 iterations with batch size 6. For all the networks, every training batch is randomly selected from the training dataset. The detailed training process can be seen in Algorithm. 1. Take the car dataset with seven parts as an example, the whole training process takes about 5 days. Once the networks are trained, generating a textured shape takes about 10 minutes.

ALGORITHM 1: Training steps

1. Train PartVAE for each part;
 2. Train SP-VAE for each shape;
 3. Train TextureVAE for each part;
 4. Load checkpoint and extract PartVAE and TextureVAE latents z_p , z_i^t and z_i^b for the central part;
 5. Train conditional PixelSNAIL for the central part on the top and bottom level discrete feature space;
 6. Load checkpoint and extract PartVAE and TextureVAE latents z_p , z_i^t , z_i^b , and VGG feature of the corresponding central part's texture for other parts;
 7. Train conditional PixelSNAIL for other parts on the top and bottom level discrete feature space.
-

5 RESULTS AND EVALUATIONS

In this section we present results of textured shape reconstruction, interpolation, generation, and automatic texturing for a new shape. We also perform ablation studies to show the benefits of each component in our design.

Textured Shape Reconstruction. Here we first compare our method with state-of-the-art geometry representation methods both qualitatively and quantitatively from the viewpoint of geometry, so only the alpha channel is used, while the color texture is ignored. In

Fig. 9, we show some reconstruction results on the test set, compared with those of existing methods. It can be seen that our reconstructed shapes are much closer to the ground truth shapes visually thanks to the power of box deformation and texture transparency. In quantitative comparison, our method also outperforms other methods in the multi-view SSIM metric. As shown in Table 2, our method gives better reconstruction performance compared with existing geometry representation methods in all five categories.

For textured shape reconstruction, we show some qualitative results. In Fig. 10, we show the ground truth shapes with texture, the input textured shapes with our representation and reconstructed textured shapes by our method. The results show that our network is able to preserve the geometry and texture details and structure, thanks to our effective textured shape representation. Further, we compare our method with a retrieval baseline on the test set in the SP-VAE latent space in Table 3. In the baseline setting, for each shape in the test set we retrieve the closest shape in the SP-VAE latent space under the Euclidean distance metric. It can be seen that our reconstructed textured shapes are superior to the baseline in all three metrics (SSIM, FID, and LPIPS).

Table 2. Quantitative comparison of our method and other geometry representation methods on shape reconstruction (without color texture) in the SSIM metric. Our method outperforms other methods in all five categories. **Note:** StructureNET uses PartNet [Mo et al. 2019b] as input which does not contain the following categories: car, guitar, and plane.

SSIM	Dataset			
	Car	Chair	Plane	Table
SDM-NET	0.9011	0.9153	0.9187	0.9247
IM-NET	0.8925	0.9073	0.8739	0.8958
BSP-NET	0.8994	0.9082	0.8678	0.9188
PSG	0.7238	0.6189	0.713	0.693
StructureNet	-	0.8034	-	0.7983
ours	0.9137	0.9232	0.9204	0.9306

Textured Shape Interpolation. Interpolation is a powerful technique to generate a shape sequence between the source shape and the target shape.

The geometry and texture are separately encoded to the latent spaces of SP-VAE and TextureVAE. So we can perform linear interpolation between shapes in both the geometry and texture latent spaces. After that a sequence of shapes with reasonable structure, geometry, and textures are generated by the decoders of the SP-VAE and TextureVAE. Interpolation examples are shown in Fig. 11. The in-between shapes look reasonable in terms of both geometry and texture. This demonstrates that the embedding spaces of SP-VAE and TextureVAE provide a meaningful domain for interpolation.

Automatic Texture Generation. Given a shape without texture, a natural problem is how to texture the shape. For this problem, we want to generate the texture that is compatible with the geometry of the shape, to ensure a plausible visual appearance.

To achieve this, we first convert the raw shape to our representation, and then encode each semantic part of it to the compact

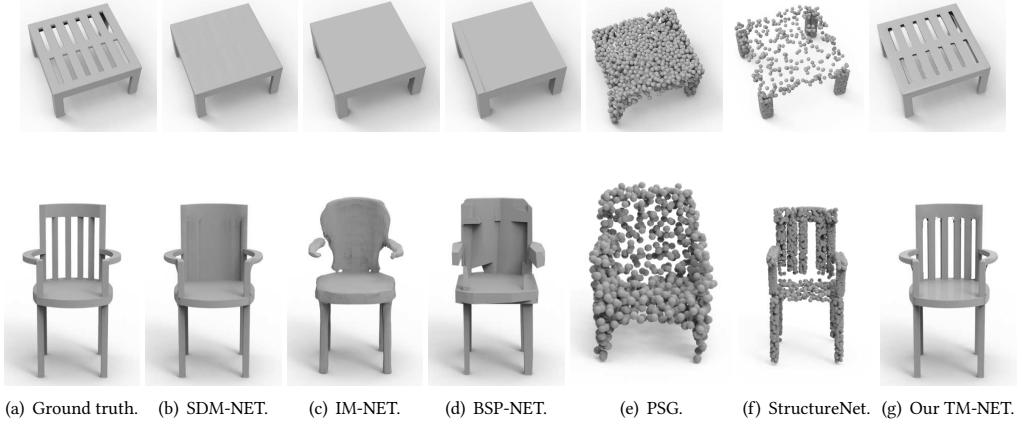


Fig. 9. Comparison of reconstruction results of several state-of-the-art 3D generative models. The part-based SDM-NET tends to produce the best overall geometry construction. However, it is unable to reproduce topological details such as holes on the chair back and tabletop. With texturing and transparency, TM-NET is the only method that can create the appearance of topological details; its reconstruction is almost visually indistinguishable from the input. In this evaluation, we only focus on the geometry, and ignore the color texture for our method.

Table 3. Quantitative comparison of our method and the baseline retrieval method in the SSIM metric between the ground truth shape and reconstructed shape. Our method outperforms the baseline in all five categories in the SSIM metric.

Dataset	Methods	Metrics		
		SSIM	FID	LPIPS
Car	Reconstruction	0.906	166.057	0.120
	Baseline	0.818	314.361	0.227
Chair	Reconstruction	0.912	203.719	0.178
	Baseline	0.816	267.612	0.196
Plane	Reconstruction	0.902	226.316	0.143
	Baseline	0.812	299.138	0.215
Table	Reconstruction	0.915	168.328	0.152
	Baseline	0.805	314.361	0.224

latent space of the corresponding PartVAE. Since we have learned the texture’s conditional distribution of the discrete representation, we can sample multiple reasonable discrete indices matrices for a given geometry latent vector. After that, the decoder of the TextureVAE can decode the inferred discrete indices matrices and produces textures for the given shape.

In Fig. 14, we show automatic texture generation results given the same geometry condition. We also compare the auto-texturing results with Texture Fields [Oechsle et al. 2019] with the same input shape. For quantitative comparison, we render the auto-texturing results in the same viewpoint and shading and compare the LPIPS values between different results generated by the same method (ours and Texture Fields) to demonstrate the diversity, and fooling rate through a user study to evaluate realism. As shown in Table 4, our

auto-texturing results are substantially more realistic and diverse than Texture Fields [Oechsle et al. 2019].

Table 4. We measure diversity using average LPIPS distance between images generated by the same method, and measure realism using the fooling rate, comparing a real rendered image of a textured shape and a fake image of the shape with generated textures. Our method produces textures which have higher realism while maintaining diversity.

Method	Realism	Diversity
	Fooling rate	LPIPS
Texture Fields	23.8%	0.148
Ours	41.5%	0.263

Textured Shape Generation. As a generative model, our method is able to generate new textured shapes. Compared to SDM-Net [Gao et al. 2019], our method not only generates high-quality structural shapes with fine geometry detail but also can generate the texture for each part that is suitable for the geometry.

The major reason for this capability is that SP-VAE encodes the structural information of different parts together with their geometry and TextureVAE not only encodes the texture information but also leverages the representation of geometry to support visually non-zero genus parts. We first sample on the meaningful compact space of SP-VAE to generate deformable shapes with fine geometry and structure but without texture. Then similar to the automatic texture generation procedure, we feed the geometry latent vectors PixelSNAIL to predict TextureVAE indices matrices. Finally the decoder of TextureVAE takes these predicted indices matrices as input to decode reasonable textures. The generated shapes are shown in Fig. 16. We also compare our results with the NVS (novel view synthesis) method VON [Zhu et al. 2018] in Fig. 17. Our generated results are much more consistent across different views while VON is inconsistent.

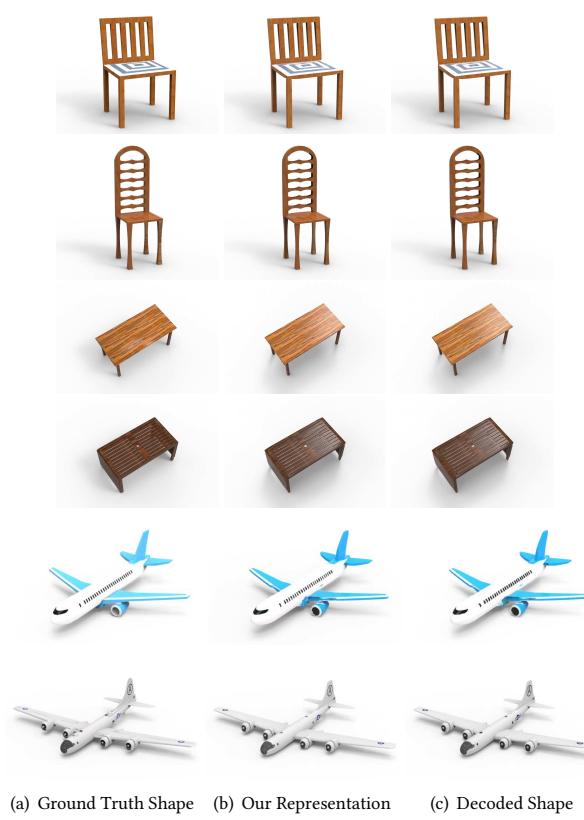


Fig. 10. Examples of decoded shapes with textures using our representation. The figure shows six examples. From left to right are the ground truth shapes, input shapes with our representation and decoded textured shapes, respectively. The results show that our representation clearly exhibits the features on geometry and texture, and the decoded shapes also maintain the realistic and detailed features, demonstrating that our framework constructs a compact and reasonable embedding space.

Evaluation on Texture Image Resolution. Naturally, texture image resolution will influence the results of the network. So we test our network using different resolutions of texture images on the test dataset. As shown in Table 5, when the image resolution is low, the SSIM is low which means the reconstruction error on the test dataset is high. When the image resolution increases, the reconstruction error decreases. But if the image resolution is too high, the generalization ability becomes worse. So we set the length of the edge l of the template box in Sec. 3 as 256 to get the best results.

Table 5. SSIM on rendered images in the chair test dataset in different resolutions. l is the length of the edge of the template box when mapped to the UV space. The table shows $l = 256$ leads to good results.

l	64	128	256	512
SSIM on texture images	0.8374	0.8591	0.9277	0.9046



Fig. 11. Interpolation between two models with different textures in test set. The first column and the last column are the shapes to be interpolated. The other columns are the in-between textured shapes by linear interpolation in the PartVAE and TextureVAE latent spaces.



Fig. 12. Interpolation between two models with different textures in test set of car category. The first column and the last column are the shapes to be interpolated. The other columns are the in-between textured shapes by linear interpolation in the PartVAE and TextureVAE latent spaces.

Ablation Studies. We perform several ablation studies to demonstrate the necessity of key components of our network architecture.

With and without the seam loss. Since our texture representation relies on unfolding boxes (see Fig. 2), pixels on the seam during



Fig. 13. Interpolation between two models with sampled textures. The textures are generated by automatic texture generation given shape geometry. The first column and the last column are the shapes to be interpolated. The other columns are the in-between textured shapes by linear interpolation in the PartVAE and TextureVAE latent spaces.



Fig. 14. Automatic texture generation. The first column is the input shape without texture. The four remaining columns are automatic texture generation results after sampling four times with the same geometry conditional input.



Fig. 15. Comparison of auto-texturing results with Texture Fields [Oechsle et al. 2019]. Every two rows are the results of an example, where the top row shows the results of Texture Fields and the bottom row presents our results.



Fig. 16. Representative results of generated shapes with textures. We first randomly sample on the latent space of SP-VAE to generate structured meshes. We then use geometry latent as condition for PixelSNAIL to generate desired textures.



Fig. 17. Textured shape generation comparison with novel view synthesis method VON [Zhu et al. 2018]. Top row: the results of VON, and bottom row: our results. VON generates multi-view images for a given shape with different view point parameters. The results show inconsistency from different viewpoints since each viewpoint is generated separately with no global consistency regularization. For our method, it can generate realistic shape (see Fig. 16), and when rendered to multiple views, the images are guaranteed to be consistent.

unfolding are mapped to two or more places in the texture space. This may lead to artifacts when pixels on the boundaries are not consistent. So we introduced a seam loss to prevent this in the TextureVAE network. This loss aims to ensure the unfolded boundaries in the texture images are as consistent as possible. Fig. 18 shows the decoded textured results of a chair without and with the seam loss L_{seam} . The results show black line artifacts (on legs and seat and highlighted with red rectangles) exist near the unfolded boundaries without the seam loss, and our result with the seam loss does not have such artifacts.

With and without central part's VGG feature condition. The autoregressive model PixelSNAIL can generate reasonable textures for a single part. But generating textures for different parts individually may lead to inconsistency between different parts. So we first generate the central part's texture, the VGG feature of which is extracted and concatenated with other parts' geometry latent code to be the conditional input of other parts' PixelSNAIL network. The

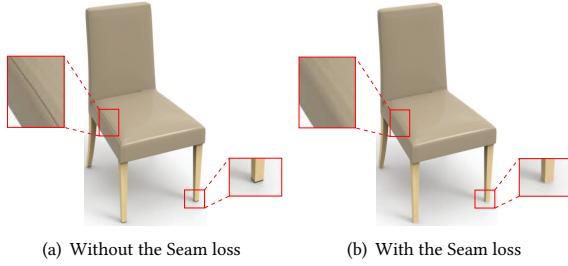


Fig. 18. Comparison of decoded results without and with seam loss L_{seam} . Due to unfolded boundaries of the texture image in our texture representation, the generated texture images may show some artifacts (inconsistency in the unfolded boundaries, as highlighted with red rectangles in (a)). The loss can prevent the artifacts (b). The results demonstrate the seam loss is effective for removing artifacts related to unfolded boundaries.

comparison of automatic texture generation results without and with central part’s VGG feature conditional input is shown Fig. 19. We can see generated textures can be incompatible between parts without central part’s VGG feature condition input.



Fig. 19. Comparison of auto-texturing results without and with the VGG feature condition input. We first generate texture for the central part chair back. Then we generate texture for other parts. It can be seen that when generating textures for the seat without the back’s VGG feature condition input, The overall shape’s texture looks incoherent.

6 CONCLUSION, LIMITATION, AND FUTURE WORK

We present a deep generative model, TM-NET, for textured mesh generation. Our network utilizes a simple and effective consistent texture atlas obtained from unfolding deformable boxes. It separately encodes box deformation and textures using VAEs at the part level and learns one-to-many relationships between geometry and texture. Textures are effectively encoded using dictionary-based vector quantization to allow high-frequency details to be faithfully synthesized. Our generative model is quite generic and supports several typical applications in the realm of realistic 3D shape modeling, e.g., synthesizing textures for a given 3D shape, generation of novel textured meshes, single-view 3D reconstruction, and latent-space interpolation, all with a unified framework.

Although TM-NET can represent shapes containing non-zero genus parts with the alpha channel in texture, it may fail when



Fig. 20. Representation failure case. The first column is the ground truth shape, the second column is our reconstructed shape. The table top has many tiny gaps on it. But in the decode shape, some of these gaps are closed. The tiny caps are achieved with the help of the alpha channel of the texture image. Those transparency values only occupy a small region on the image, which are difficult to be reconstructed by the network.

some parts have super tiny gaps on them. As shown in Fig. 20, the table top has many tiny gaps on it. Our method does not reconstruct the table top fully successfully since some of the gaps are closed in the decoded shape. In general, while the texturing trick can create the illusion of topological details such as holes in 3D shapes, our network is not always able to reconstruct or generate the actual geometric details. Still, we believe that the textures can provide sufficient constraints to help alter the geometry in a postprocess. To train a network to accomplish this in an end-to-end manner is one of the main challenges to tackle in future work.

As for other possible directions to explore, we would like to extend our method to other applications, e.g. single-view 3D reconstruction. Expansions to other types of input, such as line drawings for both shape and texture, and to other encoding schemes are also interesting to explore. Finally, we would like to extend our current representation with procedural texture generation, e.g., based on Perlin noise style texture [Henzler et al. 2019] with high frequency to improve the generated textures.

REFERENCES

- Panos Achlioptas, Olga Diamanti, Ioannis Mitliagkas, and Leonidas Guibas. 2018. Learning Representations and Generative Models for 3D Point Clouds. In *International Conference on Machine Learning (ICML)*, Vol. 80, 40–49.
- Angel X. Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, Jianxiong Xiao, Li Yi, and Fisher Yu. 2015. *ShapeNet: An Information-Rich 3D Model Repository*. Technical Report arXiv:1512.03012 [cs.GR]. Stanford University — Princeton University — Toyota Technological Institute at Chicago.
- Wenzheng Chen, Jun Gao, Huan Ling, Edward Smith, Jaakkko Lehtinen, Alec Jacobson, and Sanja Fidler. 2019. Learning to Predict 3D Objects with an Interpolation-based Differentiable Renderer. In *Advances in Neural Information Processing Systems (NIPS)*. 9605–9616.
- Xi Chen, Nikhil Mishra, Mostafa Rohaninejad, and Pieter Abbeel. 2018. Pixelsnail: An improved autoregressive generative model. In *International Conference on Machine Learning*. 864–872.
- Zhiqin Chen, Andrea Tagliasacchi, and Hao Zhang. 2020. BSP-NET: Generating Compact Meshes via Binary Space Partitioning. In *IEEE Computer Vision and Pattern Recognition (CVPR)*.
- Zhiqin Chen and Hao Zhang. 2019. Learning implicit fields for generative shape modeling. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 5939–5948.
- Alexey Dosovitskiy and Thomas Brox. 2016. Generating Images with Perceptual Similarity Metrics based on Deep Networks. *CoRR* abs/1602.02644 (2016). arXiv:1602.02644 <http://arxiv.org/abs/1602.02644>

- Alexei A. Efros and Thomas K. Leung. 1999. Texture Synthesis by Non-parametric Sampling. In *IEEE International Conference on Computer Vision (ICCV)*. 1033–1038.
- Huan Fu, Rongfei Jia, Lin Gao, Mingming Gong, Binqiang Zhao, Steve Maybank, and Dacheng Tao. 2020. 3D-FUTURE: 3D Furniture shape with TextURE. arXiv:cs.CV/2009.09633
- Lin Gao, Jie Yang, Tong Wu, Yu-Jie Yuan, Hongbo Fu, Yu-Kun Lai, and Hao(Richard) Zhang. 2019. SDM-NET: Deep Generative Network for Structured Deformable Mesh. *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH Asia 2019)* 38, 6 (2019), 243:1–243:15.
- Leon Gatys, Alexander S Ecker, and Matthias Bethge. 2015. Texture synthesis using convolutional neural networks. In *Advances in Neural Information Processing Systems (NIPS)*. 262–270.
- Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative Adversarial Nets. In *Advances in Neural Information Processing Systems (NIPS)*. Curran Associates, Inc., 2672–2680.
- Thibault Groueix, Matthew Fisher, Vladimir G. Kim, Bryan Russell, and Mathieu Aubry. 2018. AtlasNet: A Papier-Mâché Approach to Learning 3D Surface Generation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 216–224.
- Xiaoguang Han, Chang Gao, and Yizhou Yu. 2017. DeepSketch2Face: a deep learning based sketching system for 3D face and caricature modeling. *ACM Trans. Graph.* 36, 4 (2017), article 126.
- Philipp Henzler, Niloy J Mitra, and Tobias Ritschel. 2019. Learning a Neural 3D Texture Space from 2D Exemplars. arXiv:1912.04158 (2019).
- Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. 2017. GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium. In *Advances in Neural Information Processing Systems (NIPS)*. Curran Associates, Inc., 6626–6637.
- Tao Hu, Geng Lin, Zhizhong Han, and Matthias Zwicker. 2019. Learning to Generate Dense Point Clouds with Textures on Multiple Categories. arXiv:1912.10545 (2019).
- Angjoo Kanazawa, Shubham Tulsiani, Alexei A. Efros, and Jitendra Malik. 2018. Learning Category-Specific Mesh Reconstruction from Image Collections. In *Proceedings of the European Conference on Computer Vision (ECCV)*. 371–386.
- Diederik P Kingma and Max Welling. 2013. Auto-encoding variational bayes. arXiv:1312.6114 (2013).
- Johannes Kopf, Daniel Cohen-Or Chi-Wing Fu, Oliver Deussen, Dani Lischinski, and Tien-Tsin Wong. 2007. Solid texture synthesis from 2D exemplars. *ACM Transactions on Graphics* 26, 3 (2007), 2.
- Jun Li, Kai Xu, Siddhartha Chaudhuri, Ersin Yumer, Hao Zhang, and Leonidas J. Guibas. 2017. GRASS: Generative Recursive Autoencoders for Shape Structures. *ACM Trans. Graph.* 36, 4 (2017), 52:1–52:14.
- Shichen Liu, Tianye Li, Weikai Chen, and Hao Li. 2019. Soft Rasterizer: A Differentiable Renderer for Image-based 3D Reasoning. (2019), 7708–7717.
- Zhaoliang Lun, Matheus Gadelha, Evangelos Kalogerakis, Subhransu Maji, and Rui Wang. 2017. 3D Shape Reconstruction from Sketches via Multi-view Convolutional Networks. In *Proc. of the International Conference on 3D Vision (3DV)*.
- Ricardo Martin-Brualla, Rohit Pandey, Sofien Bouaziz, Matthew Brown, and Dan B Goldman. 2020. GeLaTO: Generative Latent Textured Objects. In *European Conference on Computer Vision*.
- Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. 2019. Occupancy networks: Learning 3D reconstruction in function space. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 4460–4470.
- Kaichun Mo, Paul Guerrero, Li Yi, Hao Su, Peter Wonka, Niloy Mitra, and Leonidas Guibas. 2019a. StructureNet: Hierarchical Graph Networks for 3D Shape Generation. *ACM Trans. Graph.* 38, 6 (2019), 242:1–242:29.
- Kaichun Mo, Shilin Zhu, Angel X Chang, Li Yi, Subarna Tripathi, Leonidas J Guibas, and Hao Su. 2019b. PartNet: A Large-scale Benchmark for Fine-grained and Hierarchical Part-level 3D Object Understanding. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 909–918.
- Michael Oechsle, Lars Mescheder, Michael Niemeyer, Thilo Strauss, and Andreas Geiger. 2019. Texture Fields: Learning Texture Representations in Function Space. In *Proceedings of IEEE International Conference on Computer Vision (ICCV)*. 4531–4540.
- Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. 2019. DeepSDF: Learning Continuous Signed Distance Functions for Shape Representation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 165–174.
- Dario Pavillo, Graham Spinks, Thomas Hofmann, Marie-Francine Moens, and Aurelien Lucchi. 2020. Convolutional Generation of Textured 3D Meshes. arXiv preprint arXiv:2006.07660 (2020).
- Charles Ruizhongtai Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. 2017a. PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation.. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 77–85.
- Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J. Guibas. 2017b. PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space.. In *Advances in Neural Information Processing Systems (NIPS)*. 5105–5114.
- Amit Raj, Cusuh Ham, Connelly Barnes, Vladimir Kim, Jingwan Lu, and James Hays. 2019. Learning to Generate Textures on 3D Meshes. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*.
- Ali Razavi, Aaron van den Oord, and Oriol Vinyals. 2019a. Generating diverse high-fidelity images with vq-vae-2. In *Advances in Neural Information Processing Systems*. 14866–14876.
- Ali Razavi, Aäron van den Oord, and Oriol Vinyals. 2019b. Generating Diverse High-Resolution Images with VQ-VAE. In *Deep Generative Models for Highly Structured Data, ICLR 2019 Workshop, New Orleans, Louisiana, United States, May 6, 2019*. Open-Review.net. https://openreview.net/forum?id=ryeBN88Ku4
- Stephan R. Richter and Stefan Roth. 2018. Matryoshka networks: Predicting 3D geometry via nested shape layers. In *IEEE Conference on Computer Vision and Pattern Recognition*.
- Olaf Ronneberger, Philipp Fischer, and Thomas Brox. 2015. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*. Springer, 234–241.
- Shunsuke Saito, Lingyu Wei, Liwen Hu, Koki Nagano, and Hao Li. 2017. Photorealistic facial texture inference using deep neural networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Vol. 3. 5144–5153.
- Karen Simonyan and Andrew Zisserman. 2014. Very deep convolutional networks for large-scale image recognition. arXiv:1409.1556 (2014).
- Vincent Sitzmann, Michael Zollhöfer, and Gordon Wetzstein. 2019. Scene Representation Networks: Continuous 3D-Structure-Aware Neural Scene Representations. In *Advances in Neural Information Processing Systems*.
- Xavier Snelgrove. 2017. High-Resolution Multi-Scale Neural Texture Synthesis. In *SIGGRAPH ASIA 2017 Technical Briefs (SA '17)*. ACM.
- Shao-Hua Sun, Minyoung Huh, Yuan-Hong Liao, Ning Zhang, and Joseph J Lim. 2018. Multi-view to Novel View: Synthesizing Novel Views with Self-Learned Confidence. In *European Conference on Computer Vision*.
- Qingyang Tan, Lin Gao, Yu-Kun Lai, and Shihong Xia. 2018. Variational Autoencoders for Deforming 3D Mesh Models. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 5841–5850.
- Shubham Tulsiani, Tinghui Zhou, Alexei A. Efros, and Jitendra Malik. 2017. Multi-view Supervision for Single-view Reconstruction via Differentiable Ray Consistency. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2626–2634.
- Nanyang Wang, Yinda Zhang, Zhiwen Li, Yanwei Fu, Wei Liu, and Yu-Gang Jiang. 2018b. Pixel2mesh: Generating 3d mesh models from single rgb images. In *European Conference on Computer Vision (ECCV)*. Springer, 52–67.
- Peng-Shuai Wang, Chun-Yu Sun, Yang Liu, and Xin Tong. 2018a. Adaptive O-CNN: A Patch-based Deep Representation of 3D Shapes. *ACM Trans. Graph.* 37, 6 (2018), 217:1–217:11.
- Jiajun Wu, Chengkai Zhang, Tianfan Xue, William T Freeman, and Joshua B Tenenbaum. 2016. Learning a probabilistic latent space of object shapes via 3d generative-adversarial modeling. In *Advances in Neural Information Processing Systems (NIPS)*. 82–90.
- Zhijie Wu, Xiang Wang, Di Lin, Dani Lischinski, Daniel Cohen-Or, and Hui Huang. 2019. SAGNet: Structure-aware Generative Network for 3D-Shape Modeling. *ACM Trans. Graph.* 38, 4 (2019), 91:1–91:14.
- Richard Zhang, Phillip Isola, and Alexei A Efros. 2016. Colorful Image Colorization. In *ECCV*.
- Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. 2018. The Unreasonable Effectiveness of Deep Features as a Perceptual Metric. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 586–595.
- Kun Zhou, John Synder, Baining Guo, and Heung-Yeung Shum. 2004. Iso-charts: stretch-driven mesh parameterization using spectral analysis. In *Proceedings of the 2004 Eurographics/ACM SIGGRAPH symposium on Geometry processing*. 45–54.
- Yang Zhou, Zhen Zhu, Xiang Bai, Dani Lischinski, Daniel Cohen-Or, and Hui Huang. 2018. Non-stationary Texture Synthesis by Adversarial Expansion. *ACM Trans. Graph.* 37, 4 (2018).
- Jun-Yan Zhu, Zhoutong Zhang, Chengkai Zhang, Jiajun Wu, Antonio Torralba, Joshua B. Tenenbaum, and William T. Freeman. 2018. Visual Object Networks: Image Generation with Disentangled 3D Representations. In *Advances in Neural Information Processing Systems (NIPS)*. 118–129.