AUV-Net: Learning Aligned UV Maps for Texture Transfer and Synthesis

Zhiqin Chen1,2 Kangxue Yin1 Sanja Fidler1,3,4
NVIDIA1 Simon Fraser University2 University of Toronto3 Vector Institute4

Abstract

In this paper, we address the problem of texture representation for 3D shapes for the challenging and under-explored tasks of texture transfer and synthesis. Previous works either apply spherical texture maps which may lead to large distortions, or use continuous texture fields that yield smooth outputs lacking details. We argue that the traditional way of representing textures with images and linking them to a 3D mesh via UV mapping is more desirable, since synthesizing 2D images is a well-studied problem. We propose AUV-Net which learns to embed 3D surfaces into a 2D aligned UV space, by mapping the corresponding semantic parts of different 3D shapes to the same location in the UV space. As a result, textures are aligned across objects, and can thus be easily synthesized by generative models of images. Texture alignment is learned in an unsupervised manner by a simple yet effective texture alignment module, taking inspiration from traditional works on linear subspace learning. The learned UV mapping and aligned texture representations enable a variety of applications including texture transfer, texture synthesis, and textured single view 3D reconstruction. We conduct experiments on multiple datasets to demonstrate the effectiveness of our method.

1. Introduction

The field of 3D shape reconstruction and synthesis has witnessed significant advancements in the past few years. By utilizing the power of deep learning, several works reconstruct 3D shapes from voxels, point clouds, single and multi-view images, with a variety of output shape representations [12, 14, 15, 20, 24, 30, 51]. 3D generative models have also been proposed to synthesize new shapes [4, 11, 19, 34, 39], with the aim of democratizing 3D content creation. However, despite the importance of textures in bringing 3D shapes to life, very few methods have tackled semantic-aware texture transfer or synthesis for 3D shapes [5, 10, 18, 22, 35, 36, 50].

Previous work on texture generation mostly relies on warping a spherical mesh template to the target shape [5, 10, 22, 35], therefore obtaining a texture map defined on the sphere’s surface, which can be re-projected into a square image for the goal of texture synthesis. NeuTex [47] generates 3D shapes with a neural implicit representation for arbitrary surface topology, yet embeds the surface of the shape onto a sphere, which also results in a spherical texture map. Spherical texture maps can only support limited topology, and may introduce severe distortions for thin parts such as animal limbs [27, 43]. Another line of work uses implicit texture fields for texture synthesis [32], without relying on explicit texture mapping. Although texture fields were successfully applied to multi-view image reconstruction [31], they have primarily been used for fitting a single object or scene. Generative models usually suffer from overly smoothed synthesized textures [37, 48].

In contrast, the traditional UV mapping in computer graphics handles arbitrary shape topology and avoids heavy distortions by cutting the surface into pieces and mapping different pieces to different regions on the 2D UV plane. It further preserves texture details by storing the texture in a
In this paper, we propose to train a neural network to predict the UV mapping and the texture image jointly, aiming at high-quality texture transfer and synthesis without needing to conform to a pre-defined shape topology. Specifically, our network learns to embed 3D coordinates on mesh surfaces into a 2D aligned UV space, where corresponding parts of different 3D shapes are mapped to the same locations in the texture image, as shown in Figure 1. Such alignment is enabled by a simple yet effective texture alignment module inspired by traditional linear subspace learning methods such as Principal Component Analysis (PCA), as shown in Figure 3. The network generates a basis shared by all shape textures, and predicts input-specific coefficients to construct the texture image for each shape as a linear combination of the basis images. This forces the texture images to be aligned so that they can be effectively decomposed into combinations of basis images, as visualized in Figure 2. Afterwards, the network reconstructs the colors of the input shape by learning a UV mapping to index the aligned texture image. To unwrap 3D shapes of complex structure or topology, we further introduce a masking network that cuts the shape into multiple pieces to reduce the distortion in the UV mapping.

Our method effectively aligns textures across all shapes, allowing us to swap textures between different objects, by simply replacing the texture image from one object with another. The aligned high-quality texture images produced by our method make it significantly easier to train generative models of textures, since they are aligned and disentangled from geometry. They also enable textured 3D shape reconstruction from single images. We perform extensive experiments on multiple categories including human heads, human bodies, mammals, cars, and chairs, to demonstrate the efficacy of our approach.

2. Related work

We discuss previous work that is most relevant to ours in the fields of texture transfer and synthesis for 3D shapes.

**Template-based methods** assume that all target shapes can be represented by deforming a template mesh, usually a sphere [5, 10, 22, 27, 35, 43] or a plane [33, 46]. The UV mapping of the template mesh is given and transferred to the target shape after deformation. However, by imposing a mesh template, these methods often cannot capture details, especially when the topology or the structure of the target shape is complex. For example, when deforming a sphere into a human body, it is hard to accurately reconstruct the fingers. Even if the deformation is successful, the texture of the fingers, when projected from the human body to a sphere and then to its texture image, is typically heavily distorted.

**UV map from artists.** Another line of work [8, 50] does not assume template meshes are given, but instead assumes that the UV maps are provided with the 3D shapes. The UV maps and textures are typically modeled by artists and can be in arbitrary layouts. To address this issue, these methods usually require ground-truth semantic segmentation of the texture image or the 3D shape for semantic-aware texture synthesis. In our work, we aim to perform texture synthesis without such supervision. Automatic UV mapping has also been extensively studied in computer graphics, though for single shapes. It includes mesh parameterization with certain constraints, and surface cutting to generate charts with disk topology. We refer to [38] for a survey of related techniques. Different from these traditional methods, we learn aligned UV maps for a set of shapes.

**Discretization and colorization.** Instead of adopting UV maps to reduce the dimensionality of textures from 3D to 2D, some methods discretize the 3D shapes into “atoms” and then colorize each “atom”. When a shape is represented as a voxel grid, the shape can be textured by predicting the color of each voxel [9, 41]. For triangle meshes, the color of each vertex can be predicted [17]. However, since discretization is in 3D rather than in 2D (pixels), these approaches either cannot scale up, or cannot predict the color efficiently due to the irregularity of the representation.

**Texture fields** [32] predict the color for each 3D point in a continuous 3D space. The NeRF family [31] also adopts this approach, by using the viewing direction as an additional condition for predicting the color of each point. Since the NeRF family does not directly generate textures for 3D shapes, we mainly discuss and compare with Texture Fields in this paper. One major issue of Texture Fields is that it is unable to represent high-frequency details, which is a property of the MLPs that it uses. Positional encoding [31] and SIREN [40] are proposed to alleviate this issue, which works well on overfitting of single shapes. However, performance degrades significantly in generative tasks. The results of implicit methods tend to be smooth and lack high-frequency details [6, 40].

**Shape correspondences.** There is a large body of work that finds dense correspondences among shapes [16, 28], which can also enable texture transfer. However, these methods do not take color into account when finding correspondences, which may hinder their performance.

3. Our Approach

In this section, we first introduce our core alignment module in 2D and verify it with a 2D-to-2D image alignment experiment in Sec 3.1. We then explain our network architecture for learning aligned UV maps for 3D shapes in Sec 3.2. Finally, in Sec 3.3 we show applications enabled by
our approach, including texture transfer, texture synthesis, and texture prediction from single images.

3.1. Texture alignment module

Learning aligned textures for a set of shapes is a complex task. However, we show that a simple alignment module performs surprisingly well. Before we introduce our network for 3D shapes, we will use a toy 2D experiment to demonstrate how the alignment module works.

Task. Given a set of face images in random poses, as shown in Fig. 2(a), we aim to align them into a canonical pose, as shown in Fig. 2(e). We take 1,000 face images from CelebA-HQ [25, 29] and perform random perspective transformations to obtain 128x128 training images, such as those in Fig. 2(a). To link this task with texture mapping, one could consider training images in Fig. 2(a) as square shapes in 2D, and Fig. 2(e) are their aligned texture images. The color of each pixel in the square shape must be retrieved from the shape’s texture image, by mapping the pixel’s coordinates into the UV space to get the UV coordinates, and then indexing the texture image with those UV coordinates.

Insight. We take inspiration from classic linear subspace learning methods such as eigenfaces [44], where a basis is computed via PCA for a set of face images, so that each face is decomposed into a weighted sum of the eigenfaces. Note that PCA works best when the images are aligned. Therefore, if a network is designed to decompose the input images into weighted sums of basis images, and is allowed to deform the input images before the decomposition, the network should learn to align the input images into a canonical pose, and decompose the aligned images so as to minimize the reconstruction error.

Framework. Fig. 3 illustrates our alignment module operating for 2D images. It is composed of three neural networks: a basis generator to predict a set of basis images; an encoder to predict the coefficients to weigh the basis images; and a UV mapper to predict the UV coordinates for each query point. The encoder also predicts a shape code to condition the UV mapper.

Basis generator. Our basis generator is a Multilayer Perceptron (MLP) that takes a 2D point \((x, y)\) as input and outputs the color of this point. For \(N\) basis images, the network outputs \(N\) gray-scale colors. In this toy 2D experiment, we use \(N = 128\). We adopt an MLP for generating the basis because it is fully differentiable with respect to both the colors of the basis and the input point coordinates. In contrast, if a Convolutional Neural Network (CNN) or a grid of learnable weights is applied to generate the basis, one would need to index the output grid with query points, limiting the gradients from the output color to the basis and the query point coordinates to small neighborhoods.

UV mapper. Once trained, we can input a regular grid of query points to the basis generator to obtain the aligned basis images such as those in Fig. 2(d), and the aligned texture images shown in Fig. 2(c) by multiplying the basis images with the coefficients. However, at training time, we need to “deform” the texture images to reconstruct the input images in Fig. 2(a). This is achieved by the UV mapper, which maps the query points sampled from the square shape into UV coordinates to index the texture image, as shown in Fig. 3. The final deformed outputs are shown in Fig. 2(b). The UV mapper is an MLP conditioned on a shape latent code. It takes 2D point coordinates concatenated with the shape code as input, and outputs UV coordinates.

Encoder and loss function. Since the inputs are images, we use a 2D CNN as our encoder to predict shape codes.
and coefficients. The deformed basis produced by the basis generator is multiplied with the coefficients to produce the final output, as shown in Fig. 3. We use Mean Squared Error (MSE) between the output and the input image as the reconstruction loss. During early stage of training (first few epochs), we apply a prior loss, i.e., MSE between the query point coordinates and their corresponding UV coordinates, to encourage the UV mapper to perform an identity mapping, so that the basis is initialized with appropriate orientation, scale, and position.

Obtaining high-quality texture images. After training, our network produces aligned texture images as shown in Fig. 2(c). However, the images are in low quality since they are constructed by a limited number of over-smoothed basis images. To obtain a high-quality texture image with details, we can deform the input image into the UV space. We sample points from the input image, feed those points to the UV mapper to obtain UV coordinates, use the UV coordinates and colors of the sampled points to fill a blank image, and finally inpaint the missing regions. We use [42] for inpainting. The results are shown in Fig. 2(e).

3.2. Learning aligned UV maps for 3D shapes

Our network for 3D shapes, dubbed AUV-Net, is built upon the alignment module in Fig. 3. It can be thought of as a 3D-to-2D version of the 2D-to-2D alignment module, with modifications to address issues caused by the properties of 3D shapes. The architecture of AUV-Net is shown in Fig. 4. In the following, we will first describe the notable changes in AUV-Net compared to the 2D alignment module, and then the loss functions and training details.

Predicting color, normal, and coordinate maps. 3D shapes tend to have textures with large areas of pure color or similar colors. Those featureless regions make it hard for our network to align the textures with only a loss function defined on colors. Therefore, in addition to the color maps, our network also produces normal maps and coordinate maps, as shown in Fig. 4 top-right. The normal maps are used for predicting the unit normals of input points. The coordinate maps are used for predicting the positions of the input 3D points, therefore forming a 3D-2D-3D cycle in our network, which encourages injective UV mapping.

Cutting surfaces with a masking network. Unlike images, it is usually impossible to embed a 3D shape onto a 2D plane without overlap or severe distortion. Therefore, we introduce a masker, to generate a segmentation mask for the input shape, as shown in Fig. 4 bottom-left. This can be considered as cutting the 3D shape into multiple pieces, so that each piece can be represented by a single texture image. The input to the masker contains point coordinates, point normals, and the shape code. The normals are essential to segmenting the shapes, since thin parts such as fingers on a human body mesh are very hard to segment with only point coordinates. The predicted segmentation mask \( M \) is used to mask the outputs of the two basis generators \( A \) and \( B \), as \( M \cdot A + (1 - M) \cdot B \), to produce the final output.

Multiple basis generators. We introduce two basis generators to represent the “front” and the “back” part of a shape, respectively. The “front” does not have to literally denote the front-facing part of a shape. It simply refers to a part of the shape so that the union of the “front” and “back” covers the entire shape. The outputs of the two basis generators are shown in Fig. 4 right, where the head is being represented with two texture maps. Note that the number of basis generators does not have to be two; we use four basis generators for chairs in our experiments.

Shared UV mapper. We use one shared UV mapper for both the front and the back basis generators, instead of two
independent UV mappers. This is based on a careful consid-
eration. In our experiments, one of the most prominent is-
ues when we transfer the texture from one shape to another
is that we inevitably obtain seams between the two pieces
of shapes using two different texture images. A shared UV
mapper alleviates this issue by forcing the two pieces to
share the same boundary in the texture images. It does not
fully resolve the seam issue, but it is very helpful in practice.
After we inpaint the texture images, the seams are barely
visible in most cases.

**Loss functions.** To train AUV-Net, the meshes are con-
verted into point clouds with normals and colors, as input
to our network. We also voxelize the point clouds to obtain
colored voxel grids as input to the 3D CNN encoder. The
overall loss function is composed of five terms:

\[
L = w_c L_c + w_n L_n + w_x L_x + w_s L_s + w_p L_p
\]  

(1)

where \(L_c, L_n, L_x\) denote the color loss, the normal loss, and
the cycle consistency loss on the 3D coordinates, respec-
tively. They are defined as MSEs between the predictions
and the ground truth.

\(L_s\) is the smoothness loss. For a subset of input points,
we find their neighbors within a distance \(\sigma = 0.02\), and
use the distances between the points and their neighbors
to regularize the corresponding distances in the UV space:

\[
L_s = \frac{1}{MN} \sum_{i=1}^{M} \sum_{j=1}^{N} |D(p_i, p_j) - D(q_i, q_j)| \cdot T(p_i, p_j)
\]  

(2)

where \(N\) is the number of input points, \(M\) is the size of
the subset, \(p_i\) is the \(i\)-th input 3D point, \(q_i\) is the 2D UV
point predicted for \(p_i\) by the UV mapper. \(D(a, b)\) is the
Euclidean distance between point \(a\) and \(b\). \(T(a, b)\) is defined
as \(1\) if \(D(a, b) < \sigma\), and \(0\) otherwise. In each mini-batch,
we process one shape, with \(N = 16,384\) and \(M = 2,048\).

\(L_p\) is the prior loss to initialize the UV coordinates and
the masks. It may vary per category of the training shapes.
For the human head dataset shown in Fig. 1 and Fig. 4,
where all the heads are facing z direction, we have:

\[
L_p = \frac{1}{N} \sum_{i=1}^{N} (p_i^x - q_i^x)^2 + (p_i^y - q_i^y)^2 + (m_i - n_i)^2
\]  

(3)

where \(p_i^x\) is the x coordinate of \(p_i\), \(q_i^x\) is the x coordinate
of \(q_i\), \(m_i\) is the masking value predicted by the masker for \(p_i\).
\(n_i\) is defined as \(1\) if the unit normal of \(p_i\) in the z direction
is greater than \(-0.5\), and \(0\) otherwise. This prior loss initial-
izes the UV mapping by projecting the 3D points onto the
xy-plane. To cut the shape into two pieces, this prior loss
follows our prior that: if the angle between a point’s normal
and z axis is less than 120 degrees, the point belongs to the
“front” part. Similar to Sec. 3.1, prior loss is only used in
the first few epochs of training to initialize the mask and the
UV coordinates. We provide the definitions of prior losses
for other categories in the supplementary.

**Assumptions on the training set.** Note that the above
loss terms assume certain properties of the training dataset.
First, the shapes need to have part-level correspondences,
as the network actually assigns dense correspondences be-
tween shapes when it maps all shapes into the same UV
space. Therefore, we only train our model on 3D shapes
of the same category. Second, the shapes need to be pose
aligned, e.g., heads should all face z direction in the afore-
mentioned human head dataset. We also normalize all
shapes to unit boxes before training, to avoid interference
of drastically different scales.

**Multi-stage training.** We train the network in three
stages, due to a trade-off between the quality of the tex-
ture alignment and the level of distortion. In some cases,
aligning textures requires heavy distortion in the texture
images, e.g., when aligning a sedan with a van (Fig. 5).
However, less distortion is a desirable feature that reduces
aliasing effect when rendering the textures, and makes post-
processing easier, e.g., when being edited by an artist. We
find that if the network is trained with fixed weighting
of the loss terms, we cannot get both the alignment and
minimal distortion. Therefore, we first initialize the net-
work with prior loss \(L_p\) and a set of weights focused on
minimal distortion. In the second stage, we remove \(L_p\),
and use weights focused on alignment. In the final stage,
we use weights focused on minimal distortion. For the
human head dataset, the first stage has 10 epochs, with
\(\{w_c, w_n, w_x, w_s, w_p\} = \{1, 0.5, 100, 1, 1\}\); second stage
has 2,000 epochs, with \(\{1, 0.5, 1, 0, 0\}\); third stage has
2,000 epochs, with \(\{1, 0.5, 100, 100, 0\}\). Training takes 2
days on one NVIDIA RTX 3080 Ti GPU. Other training
details are in the supplementary.

**3.3. Applications**

**Texture transfer.** After training AUV-Net, we obtain
aligned high-quality texture images (1024\(^2\) in our experi-
ments) for all training shapes, as shown in Fig. 1. The fact
that these texture images are aligned allows us to transfer
textures between two training shapes by simply swapping
their texture images, as shown in Fig. 1 and 5. We denote
this application as **Tsf (transfer)**. Given a new shape that is
not in the training set, we can also texture it by mapping its
vertices into the aligned UV space. This is done via a post-
training optimization stage, in which we add the new shape
into the training set, and continue training the network for
a few epochs. During the optimization, we fix the weights
of the basis generators to reuse the well-learned texture basis.

**Texture synthesis.** A great advantage of having aligned
texture images is that it allows us to utilize existing 2D gen-
erative models to synthesize new textures for 3D shapes.
We train StyleGAN2 [26] in experiments and show results in Fig. 9. We denote this application as Gen (generation).

**Single-image 3D reconstruction.** We can condition texture synthesis on a variety of inputs, for example, reconstructing textured 3D shapes from single images, as shown in Fig. 10. To this end, we add a 2D ResNet [21] image encoder to predict the texture latent code and the shape code from an input image, a CNN decoder to predict the aligned texture images from the texture latent code, and an IM-Net decoder [14] to predict the geometry of the shape conditioned on the shape code. We denote this application as SVR (single view reconstruction). Implementation details are provided in the supplementary.

### 4. Experiments

**Datasets.** We use six datasets in our experiments, as listed in Table 1. Information about dataset licenses is in the supplementary. We mainly perform generative tasks (Gen, SVR) on ShapeNet [7] categories since other datasets have too few training shapes. Note that the original shapes in ShapeNet usually have complex geometry but simple textures. We create a new version of ShapeNet Cars and Chairs better suited for the texture transfer/synthesis task, by simplifying the meshes to reduce geometric details and baking the geometric details into textures.

### 4.1. Texture Transfer

We show texture transfer results in Fig. 1 and 5. Only non-ShapeNet categories are shown due to page limit. More results can be found in the supplementary. Our method uses cues such as colors, normals, and positions when learning aligned UV mapping, and therefore performs well on aligning facial orifices, car windows and wheels, fingers, and animal limbs. Previous methods find dense correspondences among shapes by deforming geometry [16, 28]. However, they do not utilize color information, and may thus misalign regions with fewer geometric cues, as shown in Fig. 6. We show results on transferring textures to new shapes that are clearly different from the training shapes in Fig. 5. Our method is able to correctly texture an over-simplified texture-less car model, and transfer textures to a cartoon character model.

**Quantitative evaluation.** To evaluate the alignment quality, we label one texture image with a different color per semantic part, as shown in Fig. 7 (b). Since the texture image is aligned across shapes, we get semantic segmentation of 3D shapes with a single labeled example. We evaluate our part segmentation of shapes with ground truth segmentation provided in the ShapeNet part dataset [49]. We compare with BAE-Net [13] that performs one-shot shape segmentation and DIF-Net [16] that learns dense correspondences, and report Intersection Over Union (IOU) in Table 2. Our method outperforms alternatives.

**Ablation study.** We provide the ablation study in Table 3 and Fig. 8, where we remove one of the five loss terms in

<table>
<thead>
<tr>
<th>Dataset name</th>
<th>Number of Shapes</th>
<th>Applications</th>
</tr>
</thead>
<tbody>
<tr>
<td>ShapeNet [7] cars</td>
<td>7,497</td>
<td>Tsf, Gen, SVR</td>
</tr>
<tr>
<td>ShapeNet [7] chairs</td>
<td>6,778</td>
<td>Tsf, Gen, SVR</td>
</tr>
<tr>
<td>Turbosquid [3] cars</td>
<td>436</td>
<td>Tsf</td>
</tr>
<tr>
<td>RenderPeople [1] human bodies</td>
<td>500</td>
<td>Tsf</td>
</tr>
<tr>
<td>Triplegangers [2] heads</td>
<td>515</td>
<td>Tsf, Gen</td>
</tr>
<tr>
<td>Turbosquid [3] animals</td>
<td>442</td>
<td>Tsf</td>
</tr>
</tbody>
</table>

Table 1. Datasets used in our experiments. Tsf, Gen, and SVR refer to the applications listed in Sec. 3.3.
Figure 6. Comparison with DIF-Net [16] on texture transfer. The texture is transferred from (a) to (b). In (c), the eyes’ shapes are not changed with respect to (a), the lips are misaligned, and the hat is lower than it should be compared to (b). Those details are mostly represented in colors rather than geometry.

Figure 7. Sample texture images and segmentation on ShapeNet cars and chairs. (a) shows texture images before inpainting. Note that there are 2 texture images for each car and 4 for each chair. In (b), we show the segmentation we used to produce Table 2. A visualization on 3D shapes is shown in (c).

Table 2. Semantic segmentation results in IOU, comparing with BAE-Net [13] and DIF-Net [16].

<table>
<thead>
<tr>
<th>Dataset (#parts)</th>
<th>ShapeNet cars (4)</th>
<th>ShapeNet chairs (4)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Segmented parts</td>
<td>Wheel, body, hood, roof</td>
<td>Back, seat, leg, arm</td>
</tr>
<tr>
<td>BAE-Net</td>
<td>59.3</td>
<td>85.2</td>
</tr>
<tr>
<td>DIF-Net</td>
<td>69.0</td>
<td>80.3</td>
</tr>
<tr>
<td>AUV-Net</td>
<td>72.7</td>
<td>85.8</td>
</tr>
</tbody>
</table>

Table 3. Ablation study: semantic segmentation results in IOU.

<table>
<thead>
<tr>
<th></th>
<th>No $L_c$</th>
<th>No $L_n$</th>
<th>No $L_x$</th>
<th>No $L_s$</th>
<th>No $L_p$</th>
<th>No masker</th>
<th>Full model</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cars</td>
<td>68.5</td>
<td>71.7</td>
<td>73.0</td>
<td>72.8</td>
<td>70.6</td>
<td>72.0</td>
<td>72.7</td>
</tr>
<tr>
<td>Chairs</td>
<td>85.2</td>
<td>84.6</td>
<td>83.7</td>
<td>87.1</td>
<td>85.7</td>
<td>71.1</td>
<td>85.8</td>
</tr>
</tbody>
</table>

Table 4. Quantitative results of generative models in FID.

Table 5. Ablation study: learned texture images with different settings. Each chair has four texture images (shown vertically).

4.2. Texture Synthesis

We show texture synthesis results in Fig. 9, and compare with Texture Fields [32] (TF). Our method generates more details and gets correct alignments, while TF outputs smooth color chunks that are sometimes misaligned. This is because TF uses an MLP to map continuous 3D coordinates into colors, and does not properly disentangle texture and geometry. In contrast, our method uses a sophisticated 2D generative model trained on aligned texture images to generate the outputs. The texture images aligned by AUV-Net are mostly independent from the actual mesh geometry.

4.3. Textured Single-View 3D Reconstruction

We show results in Fig. 10, and compare with TF. For a fair comparison on texture prediction, we use the mesh generated by our method as the output mesh of TF, so that TF only needs to predict textures of the shapes. The results show that our method produces sharper boundaries and more details in the textures.

In addition to FID evaluated on a set of shapes, we use Structural Similarity Index Measure (SSIM) [45] and feature-l1 [32] to evaluate the quality of each rendered view and then average them, as proposed in TF. Table 5 reports
the results, with AUV-Net outperforming the baseline. We observe that both methods overfitted on ShapeNet chairs, and are unable to recover the correct textures of complex test shapes. This is likely due to the fact that the dataset has significant variation for chairs, but has insufficient training examples. In addition, we find that SSIM, which is not semantics-aware, may not be a good evaluation metric when the results are not very close to the ground truth. As shown in Table 5, the SSIM of ours and the baseline are very close, though other metrics show clear differences.

5. Conclusion, Limitations, and Future Work

We introduce the first method to learn aligned texture maps for a set of shapes in an unsupervised manner. We show that alignment can be done with a simple alignment module inspired by PCA. The resulting texture images of our method are well aligned and disentangled from geometry. They have enabled several applications including texture transfer, texture synthesis, and textured single view 3D reconstruction, which we showcase in our experiments.

There are three main limitations of our approach. First, our method does not handle seams that arise when texturing the shape, and only exploits a shared UV mapper network to alleviate the issue. Therefore, seams may become conspicuous for some shapes after transferring textures, as shown in Fig. 11. Second, our method does not always find correct correspondences, especially when the textures are messy, e.g., the animal dataset - one can observe that the eyes are not properly aligned in Fig. 5. Adding weak supervision could help, e.g., annotating two points for the eyes in all training shapes. Third, our method does not handle shapes with complex topology well. In fact, ShapeNet chairs pose a significant difficulty for our method with two basis generators, and we had to use four to avoid overlapping textures. We leave these challenges to future work.

Acknowledgements. We thank the reviewers for their valuable comments. This work was completed when the first author was carrying out an internship at NVIDIA.
References

[22] Paul Henderson, Vagia Tsiminaki, and Christoph Lampert. Leveraging 2D data to learn textured 3D mesh generation. In *CVPR*, 2020.  1,  2
[27] Xue ting Li, Sifei Liu, Shalini De Mello, Kihwan Kim, Xiaolong Wang, Ming-Hsuan Yang, and Jan Kautz. Online adaptation for consistent mesh reconstruction in the wild. In *NeurIPS*, 2020.  1,  2


[49] Li Yi, Vladimir G Kim, Duygu Ceylan, I-Chao Shen, Mengyan Yan, Hao Su, Cewu Lu, Qixing Huang, Alla Sheffer, and Leonidas Guibas. A scalable active framework for region annotation in 3D shape collections. Siggraph Asia, 35(6), 2016. 6

[50] Kangxue Yin, Jun Gao, Maria Shugrina, Sameh Khamis, and Sanja Fidler. 3dstylenet: Creating 3d shapes with geometric and texture style variations. In ICCV, 2021. 1, 2