3DStyleNet: Creating 3D Shapes with Geometric and Texture Style Variations

Kangxue Yin[1] Jun Gao[1,2,3] Maria Shugrina[1] Sameh Khamis[1] Sanja Fidler[1,2,3]

NVIDIA[1] University of Toronto[2] Vector Institute[3]

{kangxuey, jung, mshugrina, skhamis, sfidler}@nvidia.com

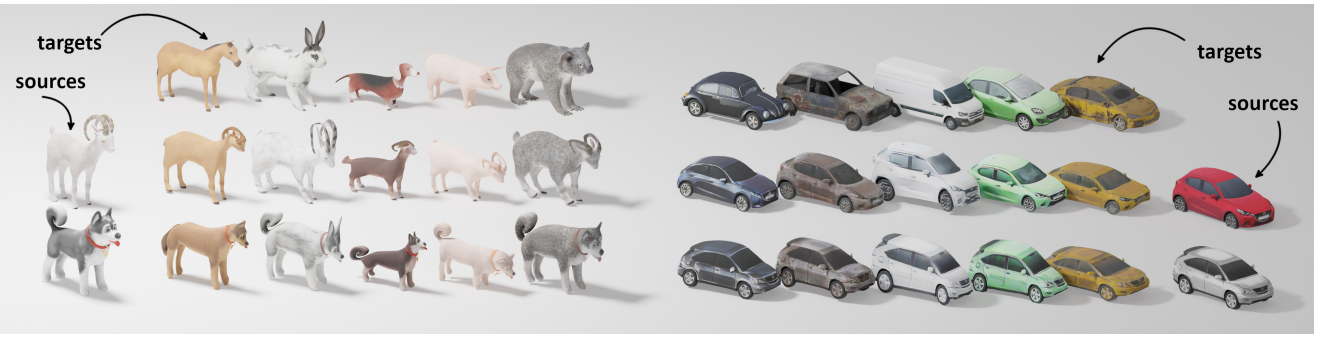

Figure 1: We propose 3DSTYLENET, a neural stylization method for 3D textured shapes. Our method creates novel geometric and texture variations of 3D objects by transferring the shape and texture style from one 3D object (target) to another (source).

Abstract

We propose a method to create plausible geometric and texture style variations of 3D objects in the quest to democratize 3D content creation. Given a pair of textured source and target objects, our method predicts a part-aware affine transformation field that naturally warps the source shape to imitate the overall geometric style of the target. In addition, the texture style of the target is transferred to the warped source object with the help of a multi-view differentiable renderer. Our model, 3DSTYLENET*, is composed of two sub-networks trained in two stages. First, the geometric style network is trained on a large set of untextured 3D shapes. Second, we jointly optimize our geometric style network and a pre-trained image style transfer network with losses defined over both the geometry and the rendering of the result. Given a small set of high-quality textured objects, our method can create many novel stylized shapes, resulting in effortless 3D content creation and style-ware data augmentation. We showcase our approach qualitatively on 3D content stylization, and provide user studies to validate the quality of our results. In addition, our method can serve as a valuable tool to create 3D data augmentations for computer vision tasks. Extensive quantitative analysis shows that* 3DSTYLENET *outperforms alternative data augmentation techniques for the downstream task of single-image 3D reconstruction. Project page:* `https://nv-tlabs.github.io/3DStyleNet/`

1. Introduction

The remarkable success of neural image style transfer has demonstrated deep learning as a powerful tool to create artistic images [13, 21, 44, 29, 30, 14, 28], with both casual and professional applications [24]. Although editing 3D content is arguably a more arduous and time-consuming task, which makes automatic tools especially attractive, equally successful formulation of style transfer for the 3D domain has not been proposed. At the same time, the demand for 3D content is growing due to the popularity of gaming, AR/VR, 3D animated films, and simulation of virtual worlds. In our work, we seek a style transfer formulation applicable to *3D content* creation, including both the geometric shape of objects and their color texture.

Prior works in 3D style transfer address either shape or color stylization, but do not extend to both of these critical attributes of 3D objects. Classical methods consider deformation transfer from one object to another with the guidance of shape correspondence [49] and the transfer of the texture map from one shape to another by minimizing distortion [41, 40]. Deep learning methods likewise addressed either geometry [33, 35, 58] or texture stylization [39]. For example, [33] proposes an energy optimization framework based on surface normals for geometry cubification, while [58] deforms a shape by utilizing a neural network to predict a cage defining a smooth warp of the shape. In [39] and [25], the style of an artistic painting is transferred to the texture or surface of a 3D object. None of the above

methods is able to perceive and transfer the geometric and texture style jointly from one 3D object to another.

In this paper, we aim to create novel geometric and texture variations of 3D objects by transferring the geometric and texture style from one 3D object to another. Unlike existing approaches, our method performs joint optimization over shape and texture to ensure consistency of the final 3D object. Our method, referred to as 3DSTYLENET, treats simple geometric relationships (e.g., relative scales, positions, rotations) between the semantic parts of a 3D object as a global shape style, which can be abstracted with a set of ellipsoids. We model geometric style transfer with a part-aware affine transformation field, defined based on the ellipsoids, that warps the semantic parts of source shape to be in similar relationships to these parts in target shape. We design a neural network to perform this task, which we train on a large dataset of 3D shapes. In order to achieve high-quality texture style transfer, a proper alignment of the object geometry is required. We therefore couple and jointly optimize our geometric style network with a pre-trained image style transfer network [28] for joint geometric and texture style transfer using losses defined over multi-view rendering produced by a differentiable renderer [27].

Our 3DSTYLENET creates variations of shapes in their style space, yielding a shape creation tool which can be used by naive users for 3D content creation. To validate the quality of our results, we conduct a user study which shows that our method can produce higher-quality results than a strong baseline that combines a SOTA shape deformation method and a SOTA image style transfer method. Furthermore, we show that our 3DSTYLENET can also serve as a 3D data augmentation method for improving the performance of downstream computer vision models. We showcase our approach as a data augmentation strategy for the task of single image 3D reconstruction, demonstrating boosts in performance over strong baseline augmentation techniques.

2. Related work

2.1. Image Stylization

Artistic stylization is a long-standing research area with a large body of work for images and videos [26, 15]. Traditional approaches rely on stroke-based approximations [17], region-based motion transfer [2], and hand-crafted local image features [18]. The classic *Image Analogies* method of Hertzmann *et al*. [18] laid the groundwork for Neural Style Transfer pioneered by Gatys et al. [13]. This has become a popular research direction in recent years [23], with the majority of Neural Style Transfer work targeting images and videos. We build on methods in the image domain [28] to target style transfer for textured 3D shapes.

2.2. Shape Editing and Stylization

Shape Deformation and Modeling. A broad class of techniques allow manipulation of 3D surface geometry, for example, using controllable handles and skeletons [47, 50, 9], governed by closed-form solutions such as linear blend skinning (LBS) [32] or energy minimization techniques [47, 50]. Unlike these interactive methods, deformation transfer seeks to automatically transfer a series of poses of a source shape to the target shape, using provided correspondences and optimization [49] or by fitting a neural network [58, 12]. The goal of our work is not to transfer deformations, but rather to stylize the input shape via a part-aware affine transformation on geometry. Compared to the recent method Neural Cage [58] (which stylizes geometry only), we achieve superior performance experimentally.

Geometric Stylization. Several approaches seek to stylize geometric features of a shape. Liu and Jacobson [33] adapt the As-Rigid-As-Possible energy [47] with L_1 regularization on surface normals to warp an input mesh into the style of a cube while maintaining shape details. Others propose automatic stylization in the style of 3D collages [10, 52], *LEGO* [37], furniture [55, 20] or Manga [43]. Style has also been considered on a more local level. Mesh smoothing or filtering, using classical operators [46] or neural networks [51], can also be viewed as a style where one progressively changes the level-of-detail while maintaining geometry-aware features [19, 48]. Recent works also propose to learn local 3D textures [16] or train style-specific mesh refinements [34]. Our approach learns deformations that stylize the global object shape in a plausible way given a target shape as a guidance.

Image-Based Approaches. Various recent works follow a 2D-to-3D approach, leveraging a pretrained style network [13] and differentiable rendering to minimize the style energy in the rendered image space [35, 25]. Parparazzi [35] focuses on editing the local geometric details, while ignoring the style of global deformations. N3MR [25] edits both texture and geometry towards the style of a target 2D image. Our aim in this work is to transfer the style from one 3D object to another.

2.3. Texture Transfer

The problem of transferring texture from one 3D shape to another can be solved by finding a dense mapping from one geometry to another. A number of geometry processing methods for computing such maps have been proposed [42, 7], but often require user-provided correspondences [8] or place strong assumptions on the input geometry. For example, most meshes found in the wild are not manifold [7], watertight [42] or homeomorphic to a disk [41]. More importantly, these purely geometric approaches lack awareness of semantic features, like eyes of a character, or the ability to modify the texture itself in order to preserve geometric patterns when applied to a vastly different geometry. In contrast, our method employs a differentiable renderer and content-aware losses that allow robust texture transfer that respects both the content and the higher frequency pat-

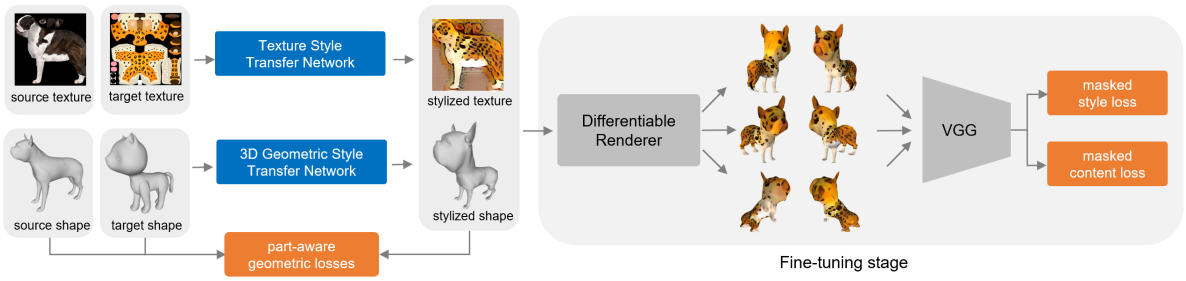

Figure 2: An overview of 3DSTYLENET, comprised of two main modules: a geometric style transfer network and a texture style transfer network. Each module is pre-trained on either 3D shape transfer or image stylization. We then perform joint geometry and texture optimization by utilizing a differentiable renderer.

terns of the texture. Our method is applicable to any triangle mesh including non-manifold meshes with holes and multiple objects. For a related task of image to 3D model texture transfer, our method can be used to directly replace the need for a 3D proxy or lighting-texture decomposition as proposed in [54].

3. Method

The goal of our method is to synthesize variations of 3D objects in their geometric and texture style space. Our method takes as input a *source* 3D model P representing the "content", and a *target* 3D model Q representing the desired "style". Unlike Neural Style Transfer for images, we *edit* the source shape rather than generating new content, thus preserving the level of detail of the source. In addition to surface geometry, our method requires a texture map of the target and, if available, of the source. Unlike prior works, 3DSTYLENET performs both geometric and texture stylization to achieve a wide range of 3D variations.

There is no single way to define the style of a 3D object. In our work, we consider global geometric style on the semantic part-level. For example, a cartoon dog may have a much larger head than its realistic counterpart. Our stylization leaves local geometric details of the source unchanged, which is often desirable for high-quality models. Instead, we treat geometric relationships between semantic parts, such as relative scales and positions, as the object's style. We abstract this style with a set of ellipsoids and model the geometric style transfer with a learned 3D part-aware affine transformation field that warps the semantic parts. We define texture style of a 3D object similarly to modern image style transfer techniques, and perform stylization in the rendered image space by leveraging differentiable rendering.

The 3DSTYLENET is comprised of two main network components, the Geometric and Texture Style Transfer Networks, as shown in Figure 2. We first pre-train the 3D Geometric Style Transfer Network (§3.1) on a set of untextured shapes and the Texture Style Transfer Network (§3.2) on image datasets. To make them work jointly for geometric and texture style transfer, we perform joint geometry and texture optimization (§3.3). A large set of 3D object variations can be created automatically by applying the joint optimization for all object pairs in a set of 3D objects, or in a user-driven design tool.

3.1. 3D Geometric Style Transfer Network

We define the geometric style of a 3D shape by focusing on relationships between semantic parts. For example, an adult tiger has a relatively smaller head and longer legs than a baby tiger. Such global geometric style is well represented by approximate shapes of semantic parts. To modify geometric style given target guidance, our 3D Geometric Style Transfer Network (Figure 3) learns to predict a part-aware affine transformation field regressed from a source and target shape pair. This predicted affine transformation field can then be used to smoothly deform the vertices of the source shape while preserving fine geometric details.

Network Architecture: Our geometric network takes point cloud samples of the two shapes as input. We use PVCNN [36] to encode the point clouds, and feed the concatenated embeddings to an MLP with skip connections. The network is defined for a fixed number of semantic parts N. For each semantic part i in the source shape, the MLP outputs parameters of an ellipsoid E_i that best approximates the part (see Figure 4), and a 3D affine transformation A_i for warping the part. This output is then used to compute a smooth affine transformation field to deform the source geometry, while preserving local details.

Part-Aware Transformations: We use the parameters of the predicted ellipsoids $E_1, \ldots, E_N$ to compute smooth skinning weights for any points on the source shape. Using these skinning weights and the N predicted affine transformations, we compute the deformation of any source point using the LBS model [32]. Abstractly, the predicted ellipsoids represent "what" to deform, and the affine transformations represent "how" to deform it. To derive the skinning weights, we observe that an ellipsoid E_i can also be represented by a 3D affine transformation composed of S_i, R_i and T_i that scales, rotates and translates a unit sphere to turn it into an ellipsoid. With this decomposed representation of the ellipsoid E_i, we can define a 3D Gaussian that aligns with it:

$$g_i(p) = G(p|T_i, \lambda S_i R_i (S_i R_i)^T) \quad (1)$$

where p is a 3D point, T_i is the mean of the Gaussian and

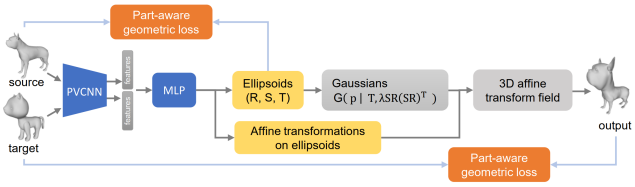

Figure 3: Our part-aware 3D Geometric Style Transfer Network.

center of the ellipsoid, $\lambda S_i R_i (S_i R_i)^{T_i}$ is the covariance matrix, and λ is a fixed scalar for controlling the spread (we use $\lambda = 4.0$). The N functions g_i derived from N part ellipsoids define an N-channel 3D blending field. We use this normalized blending field to interpolate the affine transformations A_i of all semantic parts to obtain a single 3D affine transformation field ϕ. We warp the source shape P with the affine transformation field to obtain the stylized output shape $\phi(P)$. In practice, Gaussians and the blending field are only evaluated for every vertex of the source mesh.

Part-Aware Losses: We train the 3D Geometry Style Transfer Network using part-aware geometric losses, requiring semantic part labeling of the input dataset. To this end, we manually labeled a small set of training shapes, and train a BAE-NET [5] under hybrid supervision to predict the semantic part labels for all shapes. See Supplement. for details in training the segmentation network. In Fig 4, we visualize segmentation results on a few samples. In total, we annotated $N = 11$ semantic parts for animals, 7 parts for cars, and 6 parts for people. Overall, we observed the predicted segmentation quality to be high, but smaller parts like ears were sometimes noisy.

The main component of our loss function is the part-aware distance between shapes P and Q, defined as:

$$D_{part}(P,Q) = Ch_{L1}(P,Q) + \sum_{i \in L} Ch_{L_1}(P_i, Q_i) \quad (2)$$

where P, Q are the sampled point sets and P_i and Q_i denote the point subsets for part i, and $Ch_{L1}(P,Q)$ is the averaged L_1-Chamfer distance. Including both global (first term) and part-wise (second term) Chamfer distances makes this loss more robust to segmentation noise.

The final loss function for our Geometric Style Transfer Network is:

$$\begin{aligned} Loss(P,Q,\phi) = D_{part}(\phi(P),Q) + D_{part}(\xi(P),P) \\ + \alpha(D_{sym}(\phi(P)) + D_{sym}(\xi(P))) \end{aligned} \quad (3)$$

where $\xi(P)$ is the sampled surface points of all ellipsoids $\{E_i\}$ predicted for source shape P, and $\phi(P)$ is the warp of the source shape as described above. $D_{sym}(\phi(P))$ is an optional symmetry regularization term defined as the averaged $L1$-Chamfer distance between $\phi(P)$ and its reflection over its plane of symmetry (we use $\alpha = 0.1$).

Training: Due to the limited availability of high-quality textured 3D shapes, we designed the Geometric Style Transfer Network to be trained on a large set of shapes without textures. Similarly to [58], we run self-supervised training by enumerating all possible pairwise combinations of the training shapes. We use the Adam optimizer for 40,000 iterations, with a batch size of 32. The initial learning rate is 0.001, and is halved after every 20% iterations.

Figure 4: Example part segmentation and predicted ellipsoids for the same shapes.

3.2. Texture Style Transfer Network

To transfer texture style, we reuse a linear image style transfer network [28], with texture images (Figure 2) as the source and target. The innovation of 3DSTYLENET is applying this component to 3D models using a differentiable renderer in the fine-tuning stage (§3.3). This allows the Texture Style Transfer Network to gain awareness of geometric properties, absent in unlabeled texture images alone.

Training: This network is pre-trained on images. Specifically, the encoder is part of the VGG-19 [45] trained on ImageNet [6], and the decoder is trained on MS-COCO [31]. The linear transformation module is trained on MS-COCO and WikiArt [1] as the content and style image sets, respectively. Please refer to [28] for details.

3.3. Joint Geometry and Texture Optimization

Due to the lack of textured training shapes, the Geometric and Texture Style Transfer networks are trained separately on different datasets. As the texture network is not 3D-aware, it is not able to avoid seams at the 3D surface regions where uv mapping is discontinuous, and is not able to learn to ignore the black background in the texture image (e.g., see input texture images in Fig. 2). To overcome these issues, we render the shapes with textures, and jointly optimize the geometry and texture networks at test time.

Specifically, at test time, we fine-tune both networks for a specific source and target pair. To accomplish this, we render the textured stylized object in multiple views with a rasterization-based differentiable renderer, Nvdiffrast [27], and evaluate the masked content and style loss on the multi-view rendering for joint optimization of geometry and texture. The two networks learn to work together to both hallucinate textures that respect target patterns and colors and source boundaries, and to adjust geometric transformations to make texture transfer easier.

Concretely, we jointly optimize the parameters of the MLP in the geometry style network (§3.1) and of the linear transform module of [28] (§3.2) to minimize losses defined over both the geometry and multi-view rendering of

the stylized object, using the following objective function:

$$f(\phi, \hat{m}_P) = Loss(P, Q, \phi) + \beta \sum_v L_{content}[F_v(\phi(P), \hat{m}_P), F_v(P, m_P)] + \gamma \sum_v L_{style}[F_v(\phi(P), \hat{m}_P), F_v(Q, m_Q)] \quad (4)$$

where m_P and m_Q are the texture images of the source P and target Q, and $\hat{m}_P$ is the stylized texture image (the source uv map is kept fixed). $F_v(P, m_P)$ is the set of multi-level VGG features of the rendered pixels for shape P with texture m_P under camera view v. Crucially, we use background mask output by the renderer to mask out features computed from irrelevant background pixels. $Loss$ is from Eq.3, and style and content losses on the pixel features are defined in [28]. We use $\beta = 0.01$ and $\gamma = 0.001$ in our experiments. More details about the joint optimization is provided in Supplement.

Fine-tuning Time: This fine-tuning is fast and typically converges after about 20 steps for each input pair at test-time. Given meshes with 20K faces and 512×512 texture images, it takes roughly 9$\sim$10 seconds to get the result on an Nvidia RTX 2080ti GPU.

4. Experiments

In this section, we evaluate our 3DSTYLENET qualitatively and quantitatively, and compare it to alternative baseline approaches. Throughout the experiments, we use the same default hyper-parameters for our networks. We showcase our method in stylizing 3D shapes for the animal, car and people categories. We also evaluate our method as a 3D data augmentation technique.

4.1. Dataset

Animals: The animal dataset is collected from TurboSquid[1]. We collected 1,120 non-textured shapes for training our geometric style transfer network. As described in §3.1, we selected 32 shapes which we manually labeled with semantic parts. We then train a BAE-NET with hybrid supervision to segment all 1,120 shapes into semantics parts. More details of the segmentation method can be found in the Supplement. In addition to the 1,120 non-textured shapes, we also collect 442 textured animal shapes from TurboSquid for joint geometry and texture style transfer. We screened the dataset to make sure there are no overlapping examples between the non-textured and the textured sets. We use half of the 442 animal set for generating style variations with our method, and use the remaining 221 objects for quantitative evaluation in the data augmentation experiment in §4.3.

Cars: We use 898 cars from the ShapeNet part segmentation challenge [57, 3] for training our geometric style network. In order to get desired part segmentation, we manually labeled 8 examples, and train a BAE-NET in a similar way to Animals. For joint geometric and texture style transfer, we collected 436 textured cars from TurboSquid since objects on this website come with much higher quality textures than ShapeNet objects. We use 218 for synthesizing style variations, and reserve the remaining 218 for quantitative evaluation for the data augmentation experiment.

People: We collected 500 textured 3D human models in T-pose from RenderPeople[2], and randomly split them into a training set of size 400 and a test set of size 100. To test our method in transferring cartoon style to real human shapes, we further collected 20 textured cartoon character shapes from TurboSquid for testing. We manually labeled 4 examples in the training set for segmentation with BAE-NET.

4.2. 3D Style Transfer Results

3D style transfer is one of the main applications of 3DSTYLENET. We compare our method to a strong baseline method which combines NeuralCage [58] and linear image style transfer network [28]. Specifically, in the baseline method, we train NeuralCage on our training shapes, and use it to deform the source shape to match the target shape. We then feed the source and target texture images to the pre-trained linear image style transfer network to transfer the texture style as well.

The results of the baseline and our method are shown in Fig. 5. Our method can better approximate both the geometric and texture style of the target object. Notice that the baseline largely preserves the source shape for the car category, while ours produces results closer to the target shape. Likewise, for animals, our method captures style of the target shape and texture, which is especially evident for the 5th column – while the baseline simply enlarges the dog's head, our method jointly stylizes both geometry and texture to achieve the cartoon look of the target shape. We also observe some failure cases. For example, the elephant nose in the 6-7th columns is unnaturally warped. This is because we do not define a nose part in our segmentation, and our method relies on semantic part segmentation of the source object. The human cartoon style transfer results show that our method does a better job in semantic-ware style transfer than baseline. For example, in the last column, the baseline mistakenly transfers the color of hair to human faces while our method does not.

User Study. We conducted a user study through Amazon Mechanical Turk where users were asked to compare our results against using Neural Cage [58] and Style Transfer [28] combined. We generated 2000 videos of the source and target models, which we labeled original animals *A* and *B*, and the two stylized models, which we labeled hybrid animals *C* and *D*, where in half the videos our model was hybrid *C* and in the other half it was hybrid *D* to overcome what is known as left-side selection bias. When asked whether

[1] https://d ngmj xfkvpqj jfc d .jollibeefood.rest/

[2] https:// bn xtruvt .jollibeefood.rest/

Figure 5: **Qualitative comparison:** Our method v.s. NeuralCage [58] + Linear Image Style Transfer [28]. Notice that our method better captures the style in both geometry and texture. See for example the 5th column of the animal subset. While the baseline simply enlarges the dog's head, our method jointly stylizes both geometry and texture to achieve the cartoon look of the target object.

the stylized animal models are closer to the target shape or closer to the source shape, 41.9% of users reported that our model is closer to the target than it is to the source, compared to only 26.8% for the baseline model. 69.6% of users thought our generated colors and patterns are closer to animal B than those of the baseline, and 63.4% reported that our model has more similar shape and body proportions to the target than those of the baseline. However, only 51.2% reported that our model is more unique than the baseline and 53.1% reported that our overall shape is more of a blend of the two shapes compared to the baseline. The questions we asked in the AMT survey can be found in the Supplement.

Mesh Texture Transfer. If disabling the geometric style transfer, our 3DSTYLENET can also be applied to the task of mesh texture transfer (see §2.3). Unlike purely geometric approaches that seek to find a distortion-minimizing map-

Figure 6: **Texture Transfer:** Transferring giraffe texture to a goat model (a), using method of [41] (b), and our method (c).

ping between geometries, our approach is able to hallucinate new texture images. As a result, our method avoids distortions of the source texture caused by widely different target geometry (see close ups in Fig. 6), and tends to preserve salient features of the target (such as the face of the goat). Compared to the baseline, our method is orders of magnitude faster (9$\sim$10 seconds on RTX 2080ti) and robust to non-manifold non-watertight meshes with diverse topologies, which makes it practical for processing meshes in the wild. In contrast, competitive methods in §2.3 place many requirements on the input geometry – for example, only half of the goat is available for [41] in Fig.6 because this method is designed for geometries that are homeomorphic to a disk.

4.3. 3D Data Augmentation via 3DStyleNet

Training neural networks to reconstruct 3D objects from partial observations such as from a single monocular image, requires large amounts of training data. However, obtaining very large-scale 3D object datasets with high quality geometry and texture is hard and expensive. To augment the 3D training set, the most widely used technique is domain randomization [53] which randomizes colors/textures of objects when training downstream models. In our work, we propose to use our method as a way to perform 3D synthetic data augmentation. In particular, we here focus on the task of single-image 3D reconstruction of single objects. We propose to use both geometry and texture style transfer to augment the 3D training data, and validate the performance of our model with comparisons to strong baselines.

Experimental Settings: As described in §4.1, we use 221 animal objects randomly selected from the 442 collected animals from Turbosquid as our training set for single image 3D reconstruction. We use our 3DStyleNet to augment the training set by transferring the geometry and texture from one 3D object to another, which yields 221×221 objects in total. We render all objects into 24 different views, and train DISN [56] to predict 3D shapes from images. We additionally predict RGB color for each 3D coordinate. Training details can be found in the Supplement. To evaluate performance, we use the remaining 221 objects from Turbosquid as test set, and split them into three categories: Seen Shapes, Similar Shapes, and Unseen Shapes according to the chamfer distance to the closest shape in the training set(before augmentation). Note that while the shape of the objects in Seen category matches some in the training set, the texture of the objects is distinct. Following past literature [38, 56, 11, 22], we evaluate the 3D reconstruction quality using the Chamfer Distance, Chamfer-L1, and F-score. We provide results in terms of other metrics and results on another category(Cars) in Supplement.

To evaluate our method as a data augmentation strategy, we compare with the following baselines: **a)** no data augmentation, where the network is only trained on (rendered views of) 221 training objects. For texture-based augmentation, we compare with **b)** random texture replacement akin to domain randomization work [53], where for each of the 221 objects we randomly select 221 COCO [31] images as new textures (yielding 221×221 combinations), and **c)** using image style transfer [28] to transfer the texture image style from one training object to another. For geometry-based augmentation, we compare with **d)** random affine transformations, where we randomly apply different 221 affine transformations to each of the training objects (scale of the affine transformation is the same as in our method), and **e)** using a Neural Cage model [58] trained on our 1,120 non-textured shape set to deform the geometry of each training object to another. We further compare with **f)** a combination of Neural Cage for shape deformation and Image Style Transfer [28] for texture augmentation. Finally, we also ablate our own method when using geometry style transfer only, texture style transfer only, and both texture and geometry but without fine-tuning stage. Note that across all baselines but the no-data-augmentation, we use 221×221 textured shapes for training the 3D reconstruction method, making this a fair experiment.

Experimental Results: Quantitative and qualitative results are shown in Table 1 and Fig. 7, respectively. Comparing with no data augmentation which is typically done in 3D reconstruction works, ours and baseline augmentation methods achieve significant improvements. Random Affine transformations achieves the worst results, as it degrades the quality of training data and makes it harder for the networks to learn shape priors. Comparing the Neural Cage [58] Geometry with Ours Geometry Transfer only, we achieve higher quality 3D reconstruction, showing the usefulness of affine transformation field in geometric style transfer. For texture augmentation, compared to both Random COCO [31] texture and Style Transfer [28] Texture, ours texture transfer method achieves comparable or in some metrics worst performance. Note that the texture-based augmentation methods perform best on the Seen category, which we attribute to the network overfitting to the geometry in the training set. Since some test shapes in seen category have the same shape geometry but different texture as in training set, the texture randomization methods are directly optimizing for the downstream network to disregard texture and memorize shapes. The performance of texture-randomization methods drops sig-

Augmentation Method	Chamfer ↓			Chamfer L1 ↓			F-score ↑		
	Seen Shapes	Similar Shapes	Unseen shapes	Seen Shapes	Similar Shapes	Unseen shapes	Seen Shapes	Similar Shapes	Unseen shapes
No Data Augmentation	0.025	0.035	0.065	0.073	0.102	0.187	0.323	0.210	0.085
Random Affine	0.040	0.043	0.053	0.114	0.123	0.152	0.211	0.195	0.144
Neural Cage [58] Geometry	0.017	0.022	0.044	0.050	0.064	0.128	0.415	0.346	0.143
Random COCO [31] Texure	0.014	0.021	0.045	0.040	0.061	0.131	0.572	0.361	0.140
Style Transfer [28] Texture	0.015	0.023	0.047	0.042	0.068	0.136	0.518	0.316	0.122
Neural Cage [58] + Style Transfer [28]	0.019	0.022	0.040	0.053	0.062	0.116	0.423	0.370	0.181
Ours: Geometry Transfer only	0.018	0.021	0.040	0.051	0.062	0.116	0.423	0.358	0.188
Ours: Texture Transfer only	**0.012**	0.024	0.051	**0.034**	0.070	0.148	**0.623**	0.314	0.118
Ours(w/o finetune): Texture + Geometry Transfer	0.019	0.022	0.039	0.054	0.063	0.111	0.414	0.360	0.183
Ours: Texture + Geometry Transfer	0.016	**0.019**	**0.037**	0.047	**0.055**	**0.107**	0.449	**0.394**	**0.218**

Table 1: Quantitative results on the downstream task of **Single Image 3D reconstruction** using DISN [56] as the 3D reconstruction method and 3DSTYLENET compared with baselines as a 3D data augmentation strategy.

Figure 7: Qualitative results on **Single Image 3D reconstruction** using DISN as the 3D reconstruction method and various 3D data augmentation strategies. While none of the results are perfect, some are clearly worse than others. Affine randomization hurts performance. No augmentation produces worst results than the remaining augmentation strategies. Ours produces the most plausible and smooth shapes.

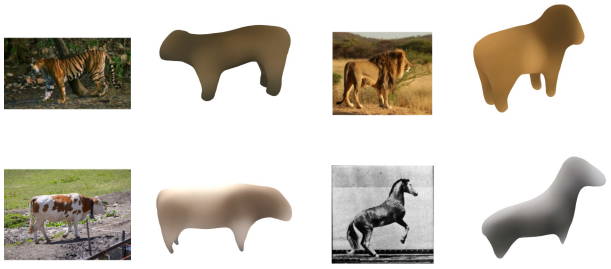

Figure 8: Qualitative results on **Single Image 3D reconstruction** on SMAL [59] Dataset using DISN [56] as the 3D reconstruction method and 3DSTYLENET as a 3D data augmentation strategy. Note that the background is masked out with the provided segmentation in the Dataset.

nificantly on the Unseen category where both textures and shapes are novel. Here our full approach significantly outperforms texture-randomization methods. Comparing Neural Cage [58] + Style Transfer [28] with our full model, we achieve comparable performance in terms of Chamfer and Chamfer-L1, and better performance in terms of F-score and qualitative results. This demonstrates the effectiveness of 3DSTYLENET as a 3D data augmentation technique.

Qualitative Results on Real Data: To evaluate how well the 3D reconstruction model trained using our augmentation strategy generalizes to real images, we directly utilize the trained network to the SMAL [59] dataset without finetuning. Since DISN [56] requires camera pose during inference, we train the occupancy network that does not require the camera views on our augmented 221×221 data. We provide qualitative results in Fig. 8. Although we only train the model on synthetic rendered images and human-created shapes from Turbosquid, the network is able to reconstruct shapes from masked real images to a good precision.

Additional results can be found in Supplement.

5. Conclusion

In this paper, we proposed a novel method for 3D object stylization informed by a reference textured 3D shape. Our 3DSTYLENET predicts a part-aware affine transformation field that warps a source shape to imitate the overall geometric style of the target shape. We also transfer the texture style of the target to the source object with the help of a multi-view differentiable renderer and the geometric alignment after shape stylization. Our method jointly optimizes the geometric style network and an image style transfer network with losses defined over both the geometry and the multi-view rendering of a pair of textured shapes. We showcase our approach on 3D content stylization, as well as a valuable tool to create 3D data augmentations for computer vision tasks. We outperform traditional augmentation techniques, particularly on the challenging shapes unseen at training time. We hope that our work opens an avenue to creative 3D content stylization and creation tooling for both naive and expert users.

References

[1] Painter by numbers, wikiart. `https://www.kaggle.com/c/painter-by-numbers`.

[2] Aseem Agarwala, Aaron Hertzmann, David H Salesin, and Steven M Seitz. Keyframe-based tracking for rotoscoping and animation. *ACM Transactions on Graphics (ToG)*, 23(3):584–591, 2004.

[3] Angel X Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, et al. Shapenet: An information-rich 3d model repository. *arXiv preprint arXiv:1512.03012*, 2015.

[4] Wenzheng Chen, Jun Gao, Huan Ling, Edward Smith, Jaakko Lehtinen, Alec Jacobson, and Sanja Fidler. Learning to predict 3d objects with an interpolation-based differentiable renderer. In *Advances In Neural Information Processing Systems*, 2019.

[5] Zhiqin Chen, Kangxue Yin, Matthew Fisher, Siddhartha Chaudhuri, and Hao Zhang. Bae-net: Branched autoencoder for shape co-segmentation. *Proceedings of International Conference on Computer Vision (ICCV)*, 2019.

[6] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[7] Michal Edelstein, Danielle Ezuz, and Mirela Ben-Chen. Enigma: evolutionary non-isometric geometry matching. *ACM Transactions on Graphics (TOG)*, 39(4), 2020.

[8] Danielle Ezuz, Behrend Heeren, Omri Azencot, Martin Rumpf, and Mirela Ben-Chen. Elastic correspondence between triangle meshes. In *Computer Graphics Forum*, volume 38, pages 121–134. Wiley Online Library, 2019.

[9] Ran Gal, Olga Sorkine, Niloy J Mitra, and Daniel Cohen-Or. iwires: An analyze-and-edit approach to shape manipulation. In *ACM SIGGRAPH 2009 papers*, pages 1–10. 2009.

[10] Ran Gal, Olga Sorkine, Tiberiu Popa, Alla Sheffer, and Daniel Cohen-Or. 3d collage: expressive non-realistic modeling. In *Proceedings of the 5th international symposium on Non-photorealistic animation and rendering*, pages 7–14, 2007.

[11] Jun Gao, Wenzheng Chen, Tommy Xiang, Clement Fuji Tsang, Alec Jacobson, Morgan McGuire, and Sanja Fidler. Learning deformable tetrahedral meshes for 3d reconstruction. In *Advances In Neural Information Processing Systems*, 2020.

[12] Lin Gao, Jie Yang, Yi-Ling Qiao, Yu-Kun Lai, Paul L Rosin, Weiwei Xu, and Shihong Xia. Automatic unpaired shape deformation transfer. *ACM Transactions on Graphics (TOG)*, 37(6):1–15, 2018.

[13] Leon A Gatys, Alexander S Ecker, and Matthias Bethge. Image style transfer using convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2414–2423, 2016.

[14] Leon A. Gatys, Alexander S. Ecker, Matthias Bethge, Aaron Hertzmann, and Eli Shechtman. Controlling perceptual factors in neural style transfer. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3730–3738, 2017.

[15] Bruce Gooch and Amy Gooch. *Non-photorealistic rendering*. CRC Press, 2001.

[16] Amir Hertz, Rana Hanocka, Raja Giryes, and Daniel Cohen-Or. Deep geometric texture synthesis. *arXiv preprint arXiv:2007.00074*, 2020.

[17] Aaron Hertzmann. Painterly rendering with curved brush strokes of multiple sizes. In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, pages 453–460, 1998.

[18] Aaron Hertzmann, Charles E. Jacobs, Nuria Oliver, Brian Curless, and David Salesin. Image analogies. In *SIGGRAPH*, 2001.

[19] Hugues Hoppe. Progressive meshes. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 99–108, 1996.

[20] Ruizhen Hu, Wenchao Li, Oliver Van Kaick, Hui Huang, Melinos Averkiou, Daniel Cohen-Or, and Hao Zhang. Co-locating style-defining elements on 3d shapes. *ACM Transactions on Graphics*, 36(3):33:1–33:15, June 2017.

[21] Xun Huang and Serge J. Belongie. Arbitrary style transfer in real-time with adaptive instance normalization. *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 1510–1519, 2017.

[22] Krishna Murthy Jatavallabhula, Edward Smith, Jean-Francois Lafleche, Clement Fuji Tsang, Artem Rozantsev, Wenzheng Chen, Tommy Xiang, Rev Lebaredian, and Sanja Fidler. Kaolin: A pytorch library for accelerating 3d deep learning research. In *arXiv:1911.05063*, 2019.

[23] Yongcheng Jing, Yezhou Yang, Zunlei Feng, Jingwen Ye, Yizhou Yu, and Mingli Song. Neural style transfer: A review. *arXiv preprint arXiv:1705.04058*, 2017.

[24] Bhautik Joshi, Kristen Stewart, and David Shapiro. Bringing impressionism to life with neural style transfer in come swim. *arXiv preprint arXiv:1701.04928*, 2017.

[25] Hiroharu Kato, Yoshitaka Ushiku, and Tatsuya Harada. Neural 3d mesh renderer. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[26] Jan Eric Kyprianidis, John P. Collomosse, Tinghuai Wang, and Tobias Isenberg. State of the "art": A taxonomy of artistic stylization techniques for images and video. *IEEE transactions on visualization and computer graphics*, 19 5:866–85, 2013.

[27] Samuli Laine, Janne Hellsten, Tero Karras, Yeongho Seol, Jaakko Lehtinen, and Timo Aila. Modular primitives for high-performance differentiable rendering. *ACM Transactions on Graphics*, 39(6), 2020.

[28] Xueting Li, Sifei Liu, Jan Kautz, and Ming-Hsuan Yang. Learning linear transformations for fast image and video style transfer. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[29] Yijun Li, Chen Fang, Jimei Yang, Zhaowen Wang, Xin Lu, and Ming-Hsuan Yang. Universal style transfer via feature transforms. In *NIPS*, 2017.

[30] Yijun Li, Ming-Yu Liu, Xueting Li, Ming-Hsuan Yang, and Jan Kautz. A closed-form solution to photorealistic image stylization. *CoRR*, abs/1802.06474, 2018.

[31] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014.

[32] Erik Lindholm, Mark J Kilgard, and Henry Moreton. A user-programmable vertex engine. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 149–158. ACM, 2001.

[33] Hsueh-Ti Derek Liu and Alec Jacobson. Cubic stylization. *ACM Transactions on Graphics (TOG)*, 38(6):1–10, Nov 2019.

[34] Hsueh-Ti Derek Liu, Vladimir G Kim, Siddhartha Chaudhuri, Noam Aigerman, and Alec Jacobson. Neural subdivision. *ACM Transactions on Graphics (TOG)*, 39(4):124–1, 2020.

[35] Hsueh-Ti Derek Liu, Michael Tao, and Alec Jacobson. Paparazzi: surface editing by way of multi-view image processing. *ACM Trans. Graph.*, 37(6):221–1, 2018.

[36] Zhijian Liu, Haotian Tang, Yujun Lin, and Song Han. Point-voxel cnn for efficient 3d deep learning. In *Advances in Neural Information Processing Systems*, 2019.

[37] Sheng-Jie Luo, Yonghao Yue, Chun-Kai Huang, Yu-Huan Chung, Sei Imai, Tomoyuki Nishita, and Bing-Yu Chen. Legolization: Optimizing lego designs. *ACM Transactions on Graphics (TOG)*, 34(6):1–12, 2015.

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

[39] Alexander Mordvintsev, Nicola Pezzotti, Ludwig Schubert, and Chris Olah. Differentiable image parameterizations. *Distill*, 2018. https://distill.pub/2018/differentiable-parameterizations.

[40] Daniele Panozzo, Ilya Baran, Olga Diamanti, and Olga Sorkine-Hornung. Weighted averages on surfaces. *ACM Transactions on Graphics (TOG)*, 32(4):1–12, 2013.

[41] Patrick Schmidt, Janis Born, Marcel Campen, and Leif Kobbelt. Distortion-minimizing injective maps between surfaces. *ACM Transactions on Graphics (TOG)*, 38(6):1–15, 2019.

[42] Patrick Schmidt, Marcel Campen, Janis Born, and Leif Kobbelt. Inter-surface maps via constant-curvature metrics. *ACM Transactions on Graphics (TOG)*, 39(4):119–1, 2020.

[43] Liang-Tsen Shen, Sheng-Jie Luo, Chun-Kai Huang, and Bing-Yu Chen. Sd models: Super-deformed character models. In *Computer Graphics Forum*, volume 31, pages 2067–2075. Wiley Online Library, 2012.

[44] Lu Sheng, Ziyi Lin, Jing Shao, and Xiaogang Wang. Avatar-net: Multi-scale zero-shot style transfer by feature decoration. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8242–8250, 2018.

[45] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[46] Olga Sorkine. Laplacian mesh processing. *Eurographics (STARs)*, 29, 2005.

[47] Olga Sorkine and Marc Alexa. As-rigid-as-possible surface modeling. In *Symposium on Geometry processing*, volume 4, pages 109–116, 2007.

[48] Oded Stein, Eitan Grinspun, and Keenan Crane. Developability of triangle meshes. *ACM Transactions on Graphics (TOG)*, 37(4):1–14, 2018.

[49] Robert W Sumner and Jovan Popović. Deformation transfer for triangle meshes. *ACM Transactions on graphics (TOG)*, 23(3):399–405, 2004.

[50] Robert W Sumner, Johannes Schmid, and Mark Pauly. Embedded deformation for shape manipulation. In *ACM SIGGRAPH 2007 papers*, pages 80–es. 2007.

[51] Towaki Takikawa, Joey Litalien, Kangxue Yin, Karsten Kreis, Charles Loop, Derek Nowrouzezahrai, Alec Jacobson, Morgan McGuire, and Sanja Fidler. Neural geometric level of detail: Real-time rendering with implicit 3d shapes. *arXiv preprint arXiv:2101.10994*, 2021.

[52] Christian Theobalt, Christian Rössl, Edilson de Aguiar, and Hans-Peter Seidel. Animation collage. In *Proceedings of the 2007 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 271–280. Citeseer, 2007.

[53] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 23–30, 2017.

[54] Tuanfeng Y Wang, Hao Su, Qixing Huang, Jingwei Huang, Leonidas J Guibas, and Niloy J Mitra. Unsupervised texture transfer from images to model collections. *ACM Trans. Graph.*, 35(6):177–1, 2016.

[55] Kai Xu, Honghua Li, Hao Zhang, Daniel Cohen-Or, Yueshan Xiong, and Zhiquan Cheng. Style-content separation by anisotropic part scales. *ACM Transactions on Graphics (Proc. of SIGGRAPH Asia 2010)*, 29(5):184:1–184:10, 2010.

[56] Qiangeng Xu, Weiyue Wang, Duygu Ceylan, Radomir Mech, and Ulrich Neumann. Disn: Deep implicit surface network for high-quality single-view 3d reconstruction. In *Advances in Neural Information Processing Systems 32*, pages 492–502. Curran Associates, Inc., 2019.

[57] Li Yi, Lin Shao, Manolis Savva, Haibin Huang, Yang Zhou, Qirui Wang, Benjamin Graham, Martin Engelcke, Roman Klokov, Victor Lempitsky, et al. Large-scale 3d shape reconstruction and segmentation from shapenet core55. *arXiv preprint arXiv:1710.06104*, 2017.

[58] Wang Yifan, Noam Aigerman, Vladimir G. Kim, Siddhartha Chaudhuri, and Olga Sorkine-Hornung. Neural cages for detail-preserving 3d deformations. In *CVPR*, 2020.

[59] Silvia Zuffi, Angjoo Kanazawa, David Jacobs, and Michael J. Black. 3D menagerie: Modeling the 3D shape and pose of animals. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, July 2017.