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

A. Appendices

A.1. Qualitative Results and Failure Cases

In the attached video "supp.mp4", we provide qualitative comparison rendered in a rotating camera view. We also provide more example failure cases of our method in the video(1:38). These cases include distortions caused by wrong segmentation(1:40), dramatic difference in parts(1:46), lack of regularization for man-made object(1:57).

To assist the readers in assessing the results shown in Fig 5 of the paper, we further render the examples in different camera view and provide them in Figure 1.

Visualization. In Figure 2 3 4, we provide additional visualization of part segmentation and ellipsoid prediction.

A.2. User Study

We conducted a user study through Amazon Mechanical Turk where users were asked to compare our results against using Neural Cage [10] and Style Transfer [5] combined. We generated 2000 videos of the source and target models for both the *Animals* dataset and the *People* dataset. For both cases we labeled the original models *A* and *B*, and the two stylized models, the hybrid animals or people, *C* and *D*. Our study setup is randomized such that in half the videos our model is hybrid *C* and in the other half it is hybrid *D* to overcome what is known as left-side selection bias. We also ran an additional study on *People* to test for model quality and identity preservation. The Animals study form that we presented to our users is shown in Figure 5.

Animals Dataset. When asked whether 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.

People Dataset. While slightly more users thought our generated results had colors, patterns, and proportions closer to the target shape, the vast majority of the results on RenderPeople were a close tie. We hypothesize that this is because we are all so attuned to the appearance of humans that it is likely a specialized technique for human stylization would be required to perform any better.

A.3. Qualitative comparison with 2D-to-3D style transfer

Directly transferring image style to 3D shapes cannot yield our desired 3D object-to-object style transfer results. To show that, we run a simple experiment, shown in Figure 7, where we use [3] to transfer the style of a single-view image of the target object(top row) to the source shape(first column). Note that the source shapes do not have texture in this experiment, as it is not supported by [3].

A.4. Implementation Details

Part segmentation. We segment animal shapes into 11 semantic parts, cars into 7 semantic parts, and people into 6 semantic parts. To train BAE-NET [1] for semi-supervised segmentation, we manually labeled 32 animal shapes, 8 car shapes, and 4 human shapes, respectively. The same hyper-parameters and configuration to the original one-shot training setting reported in the paper of BAE-NET was used in our training. Specifically, we first pre-train the BAE-NET on the small set of labeled examples for 2000 iterations to initialize the weights. After that, the network is further trained alternately using unsupervised self-reconstruction loss on unlabeled examples and supervised MSE loss on labeled examples for 200k iterations.

We found that BAE-NET does not perform well on small parts such as animal ears and car wheels. To solve this challenge, we further provide weak supervision to BAE-NET for the small parts. For example, for cars, we can

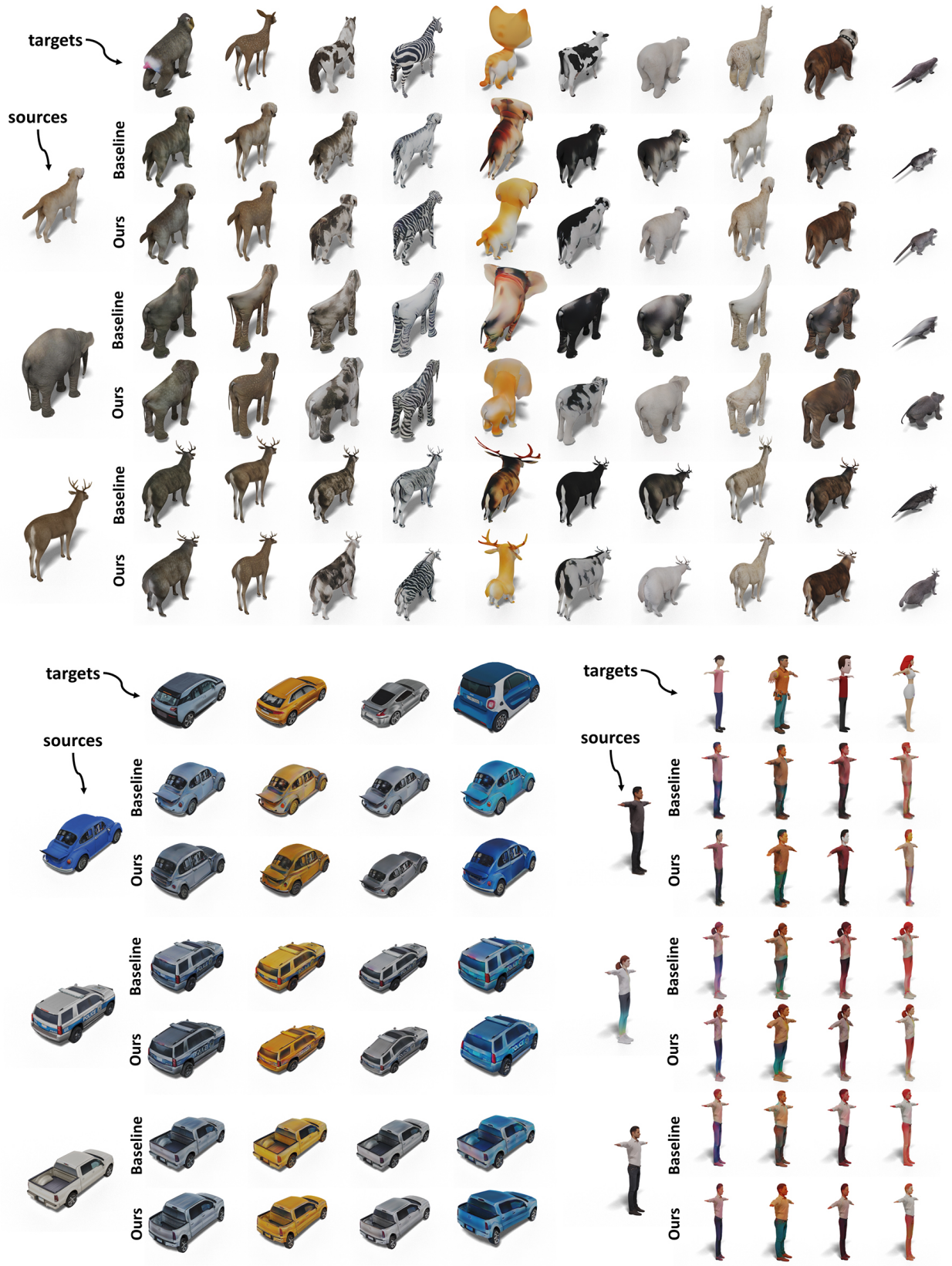

Figure 1: **Qualitative comparison:** Our method v.s. NeuralCage [10] + Linear Image Style Transfer [5].

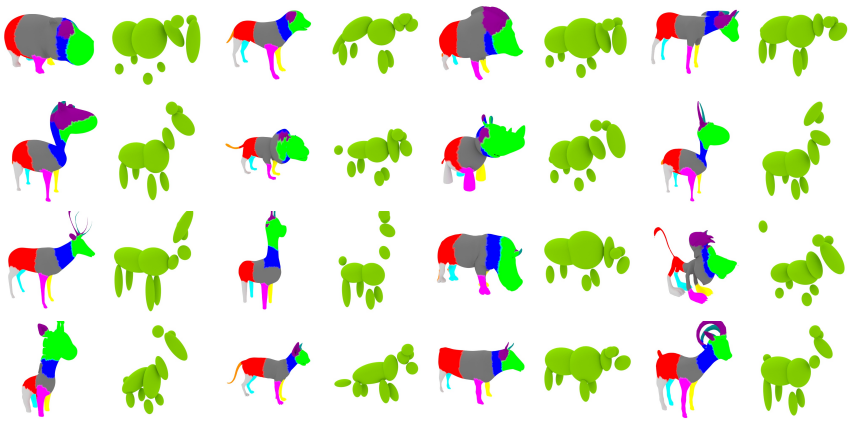

Figure 2: Visualization of part segmentation and predicted ellipsoids for animals.

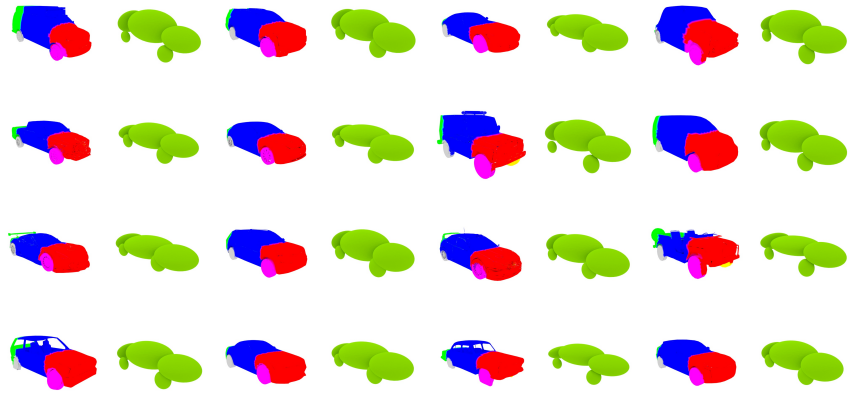

Figure 3: Visualization of part segmentation and predicted ellipsoids for cars.

tell whether a point in the point cloud of our ShapeNet training shape is a wheel point or not by looking at its original ShapeNet part annotation. We add a further loss for weak supervision defined on the wheel labels. Let us denote the probability of a point being wheel as $p = p_{LeftFront} + p_{LeftBack} + p_{RightFront} + p_{RightBack}$, where $p_{LeftFront} + p_{LeftBack} + p_{RightFront} + p_{RightBack}$ are the four-wheel probabilities predicted by BAE-NET. The weak supervision is then defined as $MSE(p, l)$, where l is the ground-truth probability from ShapeNet annotations.

We do the same weak supervision for animal ears. However, since we do not have an animal dataset with ear annotation. We render all our animal shapes into multi-view images, and train a semi-supervised DatasetGAN [11] to predict ear labels for our rendered images. Then we project the ear label back to the 3D shape via max-voting. Since the segmentation is sometimes noisy, we manually screen the ear labels, and only reserve 985 training shapes with high-quality labels. A similar MSE loss is defined on the animal ears of the 985 training shapes for providing weak supervision for BAE-NET.

Geometric Network architecture. We show in Figure 6 the detailed architectures of the PVCNN [7] and MLP used in our geometric style transfer network. As shown in the figure, we have 4 PVConv layers in the PVCNN encoder. The PVConv starts from voxelization resolution 32, and decreases it to 16, 8 and 1, while the feature getting aggregated. The MLP has 4 fully-connected layers. We connect the latent feature vectors to all the 4 FC layers in the network. The output length of the last MLP layer is $N \times m$, where N is number of semantic parts, m is the number of parameters needed for representing the ellipsoid and affine transformations of each part.

In our implementation, each ellipsoid needs 9 parameters representing R, S, T that generate it from a unit sphere, and each affine transformation needs an additional 9 parameters as we have R, S, T for them as well. Thus, the animal category requires $11 \times (9 + 9) = 198$ parameters as the output of the last FC layer, and the people category requires $6 \times (9 + 9) = 108$ parameters. However, for the car category, we reduce the number of parameters by letting the four wheels share the same 3D scaling for their ellipsoids,

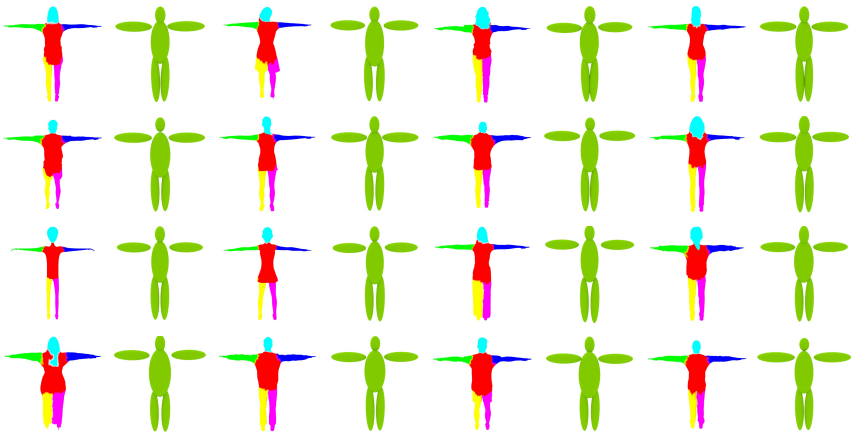

Figure 4: Visualization of part segmentation and predicted ellipsoids for people.

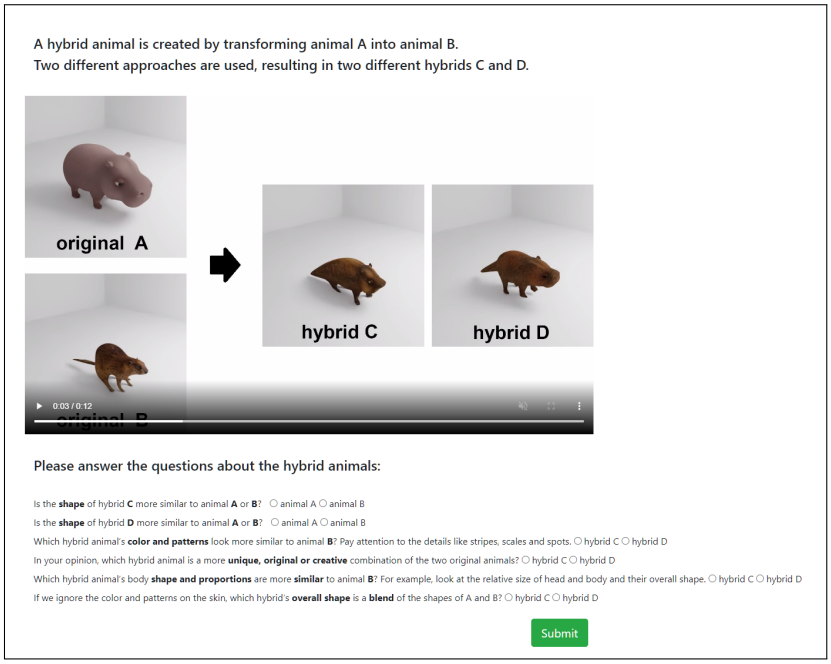

Figure 5: The survey form that we presented to Amazon Mechanical Turk users. The 3D objects are displayed rotating in 3D within a continuously looping video.

and share the same uniform scaling in their affine transformations. The rotation transformation is also disabled for car. Thus, car needs $3 * 9 + 4 * 6 + 3 = 54$ parameters for representing its ellipsoids, and needs $3 * 6 + 4 * 3 + 1 = 31$ parameters for affine transformations.

Our shapes are normalized such that its the maximal dimension is 1.0. The default range for the ellipsoid translation in our network is $[-0.5, 0.5]$, range for Euler angles is $[-0.5\pi, 0.5\pi]$, and range of ellipsoid radii is $[0.1, 0.7]$. The default range for the translation in affine transformation is $[-0.5, 0.5]$, range for Euler angles is $[-0.25\pi, 0.25\pi]$, and range for scaling is is $[0.5, 1.5]$. We find if the range of radii for ellipsoid is set to be small, the affine transformation field can be discontinuous because of potentially thin Gaussians.

Joint Geometric and Texture Style transfer Unlike [5], we use $\{relu1_1, relu1_2, relu1_3, relu1_4, relu1_5\}$ layers of VGG for style loss, and $\{relu1_5\}$ for content loss. To ensure that the camera views cover the surface the object, we initialize the camera with six orthogonal views, and randomly rotate them with the same Euler angles in each iteration of optimization. The texture image resolution is 512×512. The rendering resolution is 256×256. To mask out the VGG features of background pixels. We start from the binary mask in resolution 256×256 produced by the renderer, and compute masks in lower resolutions for deeper feature maps by max pooling.

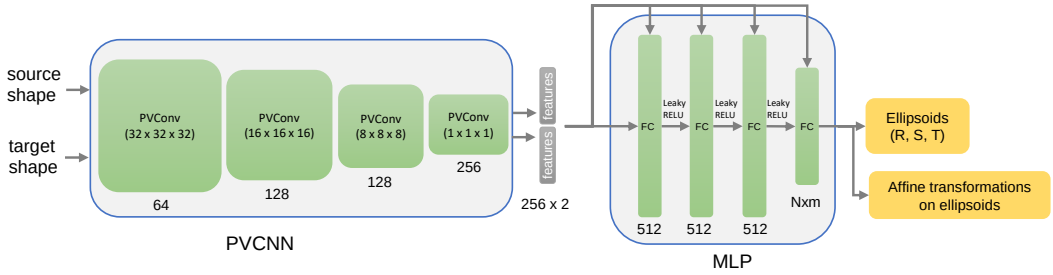

Figure 6: Detailed architectures of the PVCNN and MLP used in our geometric style network. The number below each module/layer represents the output feature dimension.

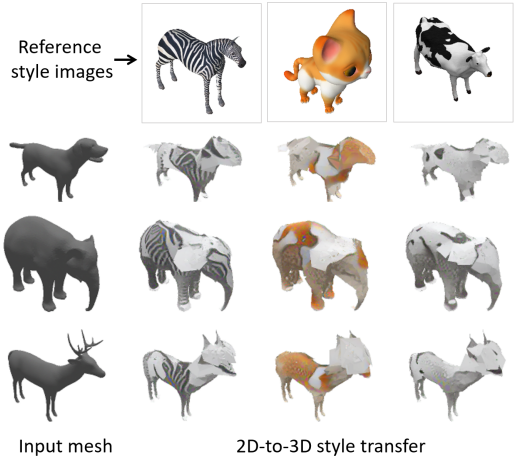

Figure 7: 2D-to-3D style transfer where single-view image of the target object is used as the style image. We use [3] as the method for this experiment.

The teaser figure. The above details are consistently used in the experiment section of the paper. However, since the teaser figure was rendered before we finalized the default setting for the experiment section, it uses a different setting which is not as good as the experiment setting. More specifically, it uses a semi-supervised BAE-NET without the hierarchical weak supervision described above for segmenting shapes. It does not use the geometric symmetry loss described in the paper. And the range of ellipsoid radii was $[0.02, 0.7]$. We found that thin ellipsoids and Gaussians can cause distortions in the results(though it produces better ellipsoids fitting), thus we increased the minimal ellipsoid radius to 0.1 in the experiment section.

A.5. Data Augmentation Experiment

In this section, we first describe the implementation details and show how we train the single image reconstruction network. Then, we present additional experiments on Single Image 3D Reconstruction on Cars.

A.5.1 Training Details

Network Architecture: We adopt the network architecture from DISN [9] for Single Image 3D Reconstruction. We briefly summarize the architecture here and refer the readers to the original paper [9] for more details. The network has one VGG16 [8] backbone which encodes the image to one global feature that represents the global context of the image, and five feature maps in different scales. For each location in 3D space, we project the 3D location into the image plane using the intrinsics and extrinsics of the camera, and grab the local feature through bilinear sampling in the multi-scale feature maps. The network then utilizes two branches to predict the occupancy for each 3D location. The first branch is taking input of global feature, concatenated with 3D coordinates, while the second branch is taking input of the sampled local features with 3D coordinates. The network simply adds two prediction to combine the two branches. In addition to occupancy prediction in the original DISN network [9], we also predict the RGB color for each 3D location.

To determine the ground truth occupancy, we convert the ground truth mesh to a watertight mesh using the pipeline from Kaolin [2], and calculate the winding number for each 3D location. To determine the ground truth RGB color for one 3D location, we find the closest point in the surface of the ground truth mesh, and use the color of the closest point as the ground truth color. The network is trained with Binary Cross Entropy on the occupancy prediction, and the Mean Squared Loss on the color prediction.

Training Settings: The network was trained using Adam optimizer [4] with 1e-4 learning rate. We implement the network using Pytorch and train on one NVIDIA-V100 GPU, the batch size is set to 8, and the whole training process takes 2 days to converge. Note that, both the baselines and our methods all use the same set of hyper-parameters and network configurations, the only difference is in the training dataset.

A.5.2 Additional Experiment Results

We first provide more quantative results on single image 3D reconstruction on Animals in Table 3. The observations we have in the main paper is consistent across other metrics.

Figure 8: Qualitative results on **Single Image 3D reconstruction** using DISN as the 3D reconstruction method and various 3D data augmentation strategies.

We also provide quantitative results on single image 3D reconstruction on Cars in Table 1 and 2. Since most of the cars have similar shapes, we can only categorize the test into Seen Shapes and similar shapes based on the closest chamfer distance to the training set. The qualitative results can be found in Figure 8.

References

[1] 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. 1

[2] 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

Augmentation Method	Chamfer ↓		Chamfer L1 ↓		F-score ↑	
	Seen Shapes	Similar Shapes	Seen Shapes	Similar Shapes	Seen Shapes	Similar Shapes
No Data Augmentation	0.014	0.023	0.040	0.063	0.458	0.283
Random Affine	0.020	0.025	0.055	0.071	0.348	0.268
Neural Cage	0.014	0.022	0.040	0.060	0.485	0.299
Random COCO	0.012	0.017	0.033	0.047	0.616	0.408
Style Transfer	0.012	0.021	0.035	0.057	0.551	0.315
Neural cage + style transfer	0.012	0.019	0.035	0.052	0.573	0.357
Our Geo	0.014	0.021	0.039	0.058	0.512	0.311
Our Tex	0.011	0.018	0.032	0.050	0.618	0.374
Our(w/o finetune) Tex + Geo	0.012	0.018	0.033	0.050	0.622	0.397
Out Tex + Geo	0.012	0.018	0.033	0.049	0.619	0.403

Table 1: Quantitative results on the downstream task of **Single Image 3D reconstruction** using DISN [9] as the 3D reconstruction method and 3DSTYLENET compared with baselines as a 3D data augmentation strategy on Cars.

Augmentation Method	3D IoU ↑		Mean Hausford ↓		Max Hausdorff ↓	
	Seen Shapes	Similar Shapes	Seen Shapes	Similar Shapes	Seen Shapes	Similar Shapes
No Data Augmentation	0.806	0.682	0.012	0.022	0.064	0.096
Random Affine	0.717	0.644	0.018	0.024	0.092	0.107
Neural Cage	0.815	0.699	0.012	0.020	0.064	0.091
Random COCO	0.869	0.797	0.009	0.015	0.056	0.074
Style Transfer	0.849	0.739	0.010	0.019	0.060	0.087
Neural cage + style transfer	0.852	0.753	0.010	0.017	0.059	0.083
Our Geo	0.834	0.732	0.011	0.019	0.062	0.087
Our Tex	0.869	0.774	0.008	0.016	0.058	0.082
Our(w/o finetune) Tex + Geo	0.856	0.772	0.009	0.016	0.054	0.076
Our Tex + Geo	0.863	0.785	0.009	0.015	0.055	0.075

Table 2: Quantitative results on the downstream task of **Single Image 3D reconstruction** using DISN [9] as the 3D reconstruction method and 3DSTYLENET compared with baselines as a 3D data augmentation strategy on Cars.

Augmentation Method	3D IoU ↑			Mean Hausdorf ↓			Max Hausdorff ↓		
	Seen Shapes	Similar Shapes	Unseen shapes	Seen Shapes	Similar Shapes	Unseen shapes	Seen Shapes	Similar Shapes	Unseen shapes
No Data Augmentation	0.641	0.519	0.317	0.024	0.034	0.064	0.129	0.161	0.218
Random Affine	0.525	0.493	0.472	0.040	0.043	0.052	0.187	0.193	0.211
Neural Cage	0.744	0.683	0.530	0.015	0.020	0.042	0.109	0.124	0.189
Random COCO	0.806	0.681	0.480	0.012	0.019	0.043	0.104	0.122	0.189
Style Transfer	0.791	0.651	0.444	0.012	0.022	0.046	0.100	0.129	0.181
Neural cage + style transfer	0.747	0.707	0.552	0.017	0.020	0.038	0.122	0.128	0.176
Our Geo	0.748	0.699	0.559	0.016	0.020	0.038	0.115	0.122	0.174
Our Tex	0.825	0.632	0.403	0.009	0.022	0.050	0.080	0.124	0.192
Our(w/o finetune) Tex + Geo	0.747	0.698	0.569	0.017	0.020	0.037	0.118	0.131	0.165
Our Tex + Geo	0.773	0.723	0.587	0.014	0.017	0.036	0.105	0.114	0.171

Table 3: Quantitative results on the downstream task of **Single Image 3D reconstruction** using DISN [9] as the 3D reconstruction method and 3DSTYLENET compared with baselines as a 3D data augmentation strategy. The results in this table are for Animals.

learning research. In *arXiv:1911.05063*, 2019. 5

[3] Hiroharu Kato, Yoshitaka Ushiku, and Tatsuya Harada. Neural 3d mesh renderer. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 1, 5

[4] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 5

[5] 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. 1, 2, 4, 6

[6] 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. 6

[7] 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. 3

[8] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 5

[9] 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. 5, 7

[10] Wang Yifan, Noam Aigerman, Vladimir G. Kim, Siddhartha Chaudhuri, and Olga Sorkine-Hornung. Neural cages for detail-preserving 3d deformations. In *CVPR*, 2020. 1, 2, 6

[11] Yuxuan Zhang, Huan Ling, Jun Gao, Kangxue Yin, Jean-Francois Lafleche, Adela Barriuso, Antonio Torralba, and Sanja Fidler. Datasetgan: Efficient labeled data factory with minimal human effort. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10145–10155, 2021. 3