3-Dimensional Style Transfer

Jui Shah
UMass Amherst
jbshah@umass.edu

Yamini Kashyap
UMass Amherst
ykashyap@umass.edu

Abstract

Neural style transfer is a widely studied task in the field Computer Vision. Numerous techniques have been explored and evaluated for style transfer from one 2D image to another. However, not many architectures have been developed for style transfer from 3D images to 3D images. In this project, we wish to implement a recently proposed architecture for 3D style transfer to the tasks of style transfer for 3D images. We try to replicate the method proposed in 3DSNet[6] to perform the style transfer across point clouds of wine bottle and jug. We use PointNet as our content encoder, style encoder and discriminator and use AtlasNet[4] as our decoder. We also perform an analysis on what effect the use of a shared style encoder and separate style encoders have on the end result. Additionally, we introduce a new loss to the 3DSNet and perform style transfer for jug and bottle.

1. Introduction

In the domain of computer vision and NLP, style-transfer has become a widely researched topic. Style transfer essentially is the preservation of some content of the original class while changing the style to that of a target class. In NLP, the content of the sentence is preserved while the style parts such as emotion, humor etc. are changed. In vision, the semantic content of original image is preserved while the style is changed to that of the target image. After the advent of the Generative Adversarial Networks (GAN)s [3] and auto-encoders [7], they began to be increasingly used for this task.

In 3D, style similarity is defined as similar shaped, salient, geometric elements [1]. We use one category of shapes to conduct our experiments of style transfer as shape is a fundamental element of 3D objects which can be identified easily upon visualization for human evaluation. In this paper, we choose our two classes as Wine Bottle(Class 0) and Jug(Class 1) to perform style transfer. Both of these classes share the same category of being water containers but both differ greatly in style. A bottle is an elongated cylinder while a jug is spherical in shape. When transferring style from a jug to a bottle, we expect the bottle(content) to open outwards in the shape of a voluminous sphere and when transferring style of bottle to a jug, we expect the jug to squeeze inwards in the shape of an elongated cylinder.

Applications of 3D style transfer lie majorly in the Virtual and Augmented Reality domain. People can visualize their own homes with desired furniture styles or try to visualize their own clothes in a different style to get the desired fashionable look. There is also a big use in virtual gaming where people can generate new objects of about anything from avatars, clothes, houses, tools, cars, rockets, office spaces, etc.

Conventionally, style transfer has been reduced to simpler tasks of learning shapewise deformation [5]. In this paper, we try to replicate the method proposed in 3DSNet[6] to perform the style transfer. We use PointNet as our content encoder, style encoder and discriminator and AtlasNet[4] as our decoder. We also perform an analysis on what effect the use of a shared style encoder and separate style encoders have on the end result. In the paper, a shared style decoder has been proposed along with adaptive normalization layers in the AtlasNet decoder. Due to computational constraints, we did not implement the adaptive normalization layers and wondered if using exclusive style decoders for each class will help achieve similar results.

In this paper -
1) We deal with Point Clouds and use a variety of reconstruction and adversarial losses to achieve style transfer.
2) We perform style transfer on the ShapeNet classes - wine bottle and jug.
3) We do not use the adaptive normalization layer in AtlasNet decoder and introduce separate style encoders instead.
4) We analyse the effect of using 2 style encoders instead of 1 by implementing each of them iteratively. 5) We also add L_1 content and style encoder reconstruction losses.
6) We use human evaluation to see the presence of style transfer. Since, ground truths are not available for the final result, it is left upto our imagination to evaluate the results. Keeping in mind, the acute lack of computational resources, we feel the basic style transfer inferred in our presented re-

sults is of value.

2. Related Work

Mattia Segu et. al.illustrated the encoder-decoder based approach called 3DSNet[6] for 3D style transfer. In their implementation, the final image gets a texture(style) heavily inspired by the style image and a shape(content) heavily inspired by the content image. Their implementation involved the style transfer across *chair* $\leftrightarrow$ *armchair*, *jet* $\leftrightarrow$ *fighter aircraft* classes of the ShapeNet dataset[2] and *horses* $\leftrightarrow$ *hippos* classes of SMAL dataset, on which they evaluated their methodology.

3. Experimental set-up

We implement a replica of 3DSNet framework. Section 3.1 describes the dataset we take to perform style transfer upon. Section 3.2 discusses the preprocessing steps. Section 3.3 describes our model architecture in depth and Section 3.4 discusses the training.

3.1. Dataset

ShapeNet dataset [2] is a publicly available dataset for 3D models and manually verified category and alignment annotations. ShapeNet consists of 55 common object categories with about 51,300 unique 3D models. We use the ShapeNet Point Clouds dataset as we deal with Point Clouds as input and output for our model. We use the taxonomy.json file to extract the wine bottle and jug data files. We have 119 number of point clouds for wine bottle and 30 number of point clouds for jug.

3.2. Data Preprocessing

We process the point clouds via BoundingBox normalization before passing the point cloud to our model. A BoundingBox is an imaginary cube that serves as a frame of reference for a particular region in a point cloud. By normalizing bounding box, we can refer to the same relative area of an image in a different point cloud so that it makes the style translation easier and efficient.

3.3. Model Architecture

We do two sets of experiments: With different style encoders for each class and with common style encoder for both the classes of 3D point cloud.

Figure 1 represents our proposed method (with different style encoders). Apart from the style encoder, we use two different content encoder, 2 decoders, 1 discriminator.

Content Encoder Input point cloud of each class is passed through a content encoder of that class. Content Encoders are basically CNN-based PointNets. The architecture details are defined in Table 1. The content encoder takes in a batch of 3D point clouds and maps it to a 1024D latent space vector, also knows as content vector.

Layers	Content Encoder	Style Encoder
Conv	(3, 64, 1)	(3, 64, 1)
Conv	(64, 128, 1)	(64, 128, 1)
Conv	(128, 1024, 1)	(128, 512, 1)
Lin	(1024, 1024)	(512,512)
Lin	(1024, 1024)	(512,512)

Table 1: Content and Style Encoder PointNet architecture

Style Encoder Similar to the content encoder, the style encoder is also a CNN-based PointNet with slightly different hidden layer sizes (Table 1). The style encoder takes in a batch of 3D point clouds and maps it to a 512D latent space vector, also knows as style vector.

Decoder We use the current state of the art decoder architecture - AtlasNet [4] with Mapping2Dto3D layers. It takes as input both - the content and the style feature vectors, to decode them into a final 3-channel style-transferred 2500D point cloud.

Discriminator A discriminator is introduced for adversarial training, to aid the training of abovementioned pipeline of the generator(content encoder, style encoder, decoder). It consists of a PointNet based encoder followed by an MLP layer. The encoder is shared for both the classes and has the same PointNet architecture as Content encoder (Table 1). The MLP is 1 layer FCN(1024 -¿ 1). The MLP layer is separate for both the classes.

3.4. Training

We train our model using a mixture of reconstruction and adversarial training losses.

3.4.1 3D reconstruction loss

The use of encoder and decoder architectures in our model pilots to the use of the reconstruction loss. This loss enures that output reconstructed by the decoder is similar to the original point clouds, correspondingly in content and style and we use unidirectional chamfer loss to calculate the loss between two point sets -

$$\mathcal{L}_{rec}^{x_i} = E_{\mathcal{T},\mathcal{Y}_{1,1}}[\sum_{p\in\mathcal{T}} \min_{q\in\mathcal{Y}_{1,1}} ||p-q||^2] \quad (1)$$

Here, $\mathcal{T}$ is set of points sampled from the target and $\mathcal{Y}_{1,1}$ is the set of generated vertices.

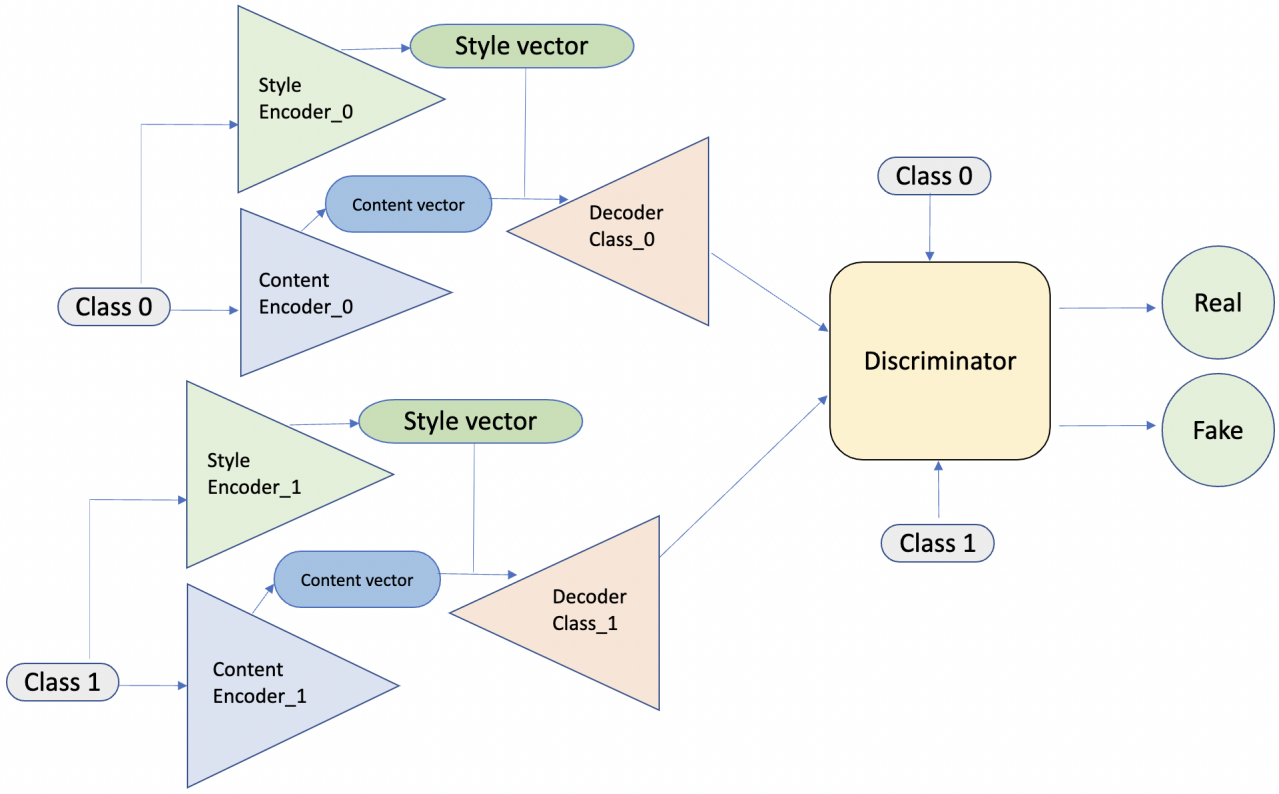


Figure 1: Proposed Model architecture

3.4.2 Adversarial loss

We use the standard adversarial loss for our 3D domain. Discriminator tries to distinguish between the generated outputs and the original ones. Here, i and j represent our two classes -

$$\mathcal{L}_{adv}^{x_j} = \mathbb{E}_{x_j}[\log D_j(x_j)] + \mathbb{E}_{c_i,s_j}[1 - \log D_j(G_j(c_i, s_j))] \tag{2}$$

3.4.3 3D Cycle reconstruction loss

Cycle loss is also added to the proposed method. We exploit the property of autoencoder and cycle loop and use reconstruction loss defined in 3.4.1 on the reconstructed output to our model, that is, x_1 is compared to $\phi_{2->1}(\phi_{1->2}(x_1))$.

3.4.4 Latent space loss

For both the style and content encoders, we add the L1 reconstruction loss where we check if the latent space vectors extracted by the encoders on the reconstructed outputs are similar to the latent space vectors of the original inputs.

$$\mathcal{L}_{se}^{x_i} = \lambda \sum_{i=1}^{n} |s(x_i) - s(\phi_{1to2}(x_i))| \tag{3}$$

$$\mathcal{L}_{ce}^{x_i} = \lambda \sum_{i=1}^{n} |c(x_i) - c(\phi_{1->2}(x_i))| \tag{4}$$

where c is the content encoder and s is the style encoder and $\phi_{1->2}$ represents our model combining content of class 1 with the style of class 2.

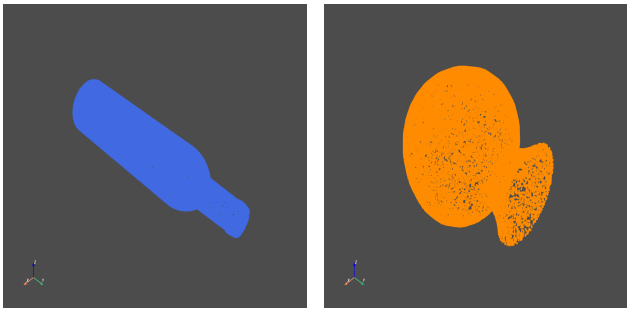
3.4.5 Total Loss

Our total loss is defined as followed -

$$\begin{aligned}\mathcal{L}_{tot} =& \lambda_{rec}(\mathcal{L}_{rec}^{x1}) + \mathcal{L}_{rec}^{x2}) \\ &+\lambda_{adv}(\mathcal{L}_{adv}^{x1}) + \mathcal{L}_{adv}^{x2})+ \\ &+\lambda_{cycle}(\mathcal{L}_{cycle}^{x1}) + \mathcal{L}_{cycle}^{x2})+ \\ &+\lambda_{se}(\mathcal{L}_{se}^{x1}) + \mathcal{L}_{se}^{x2})+ \\ &+\lambda_{ce}(\mathcal{L}_{ce}^{x1}) + \mathcal{L}_{ce}^{x2})\end{aligned} \tag{5}$$

3.5. Training Parameters

We train our model for 180 epochs and keep the learning rate of the generator to be $1e-3$ and that of discriminator to be $4e-3$. We use the LeakyReLU(0.2) as activation layer and BatchNorm as normalization layer after each Conv/Lin layer. We keep the batch size to be 2 and the number of points generated by our StyleAtlasNet decoder is set to 2500



(a) Wine Bottle (b) Jug

Figure 2: Original point clouds

(a) Wine Bottle (b) Jug

Figure 3: Different style encoders reconstructed outputs

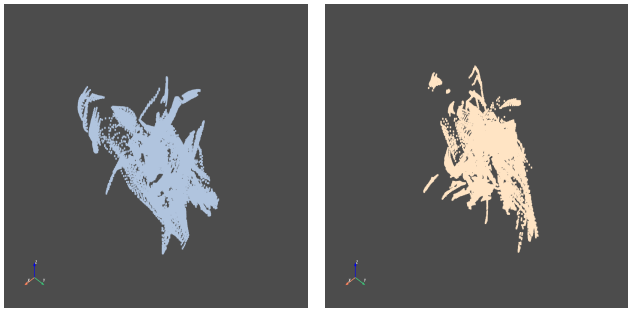
per iteration. At test time, the reconstruction is done by appending the outputs of 12 chunks of pairs sampled from content and style input clouds in an iterative manner.

4. Results

For the scope of this paper, we are only able to perform manual visual evaluation of our results.

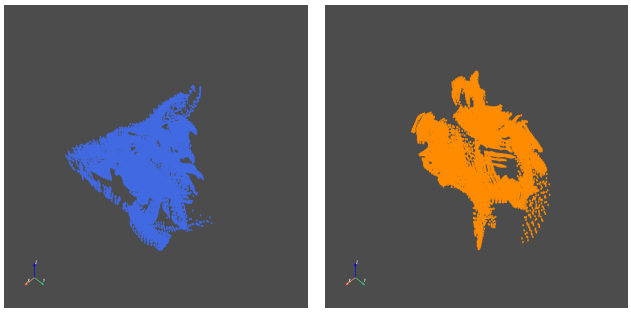
Figure 2 shows our original point cloud images for which we use our proposed method to perform style transfer. Figure 3 shows the reconstructed bottle and jug themselves when separate style encoders are used, that is, we pass the style and content vectors of the same class to their respective decoders to reconstruct the original point cloud. However, a perfect reconstruction does not occur. But, it should be noted that some elements of the original classes are preserved - the point cloud of the bottle is more compact and it looks almost like a cylinder while that of the jug is dispersed - widespread like a sphere.

When style transfer is performed with separate style encoders, the results are as shown in Figure 4. Upon comparing with the respective images in Figure 3 and 4 (3a-4a and 3b-4b), we can observe that shape of the bottle remains same to an extent but the points get more dispersed as if the bottle is getting swollen and trying to expand to be a sphere (jug). At the same time, the style changed jug(Figure 4b) is



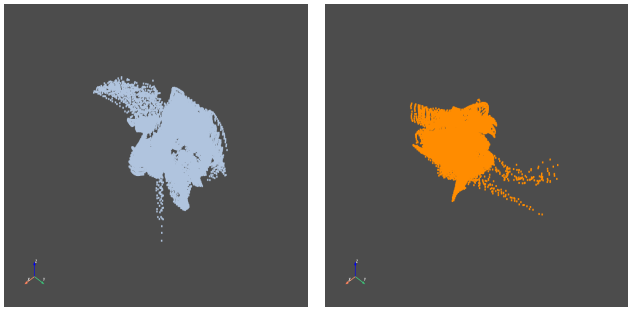
(a) Bottle to Jug (b) Jug to Bottle

Figure 4: Different encoders style transfer outputs



(a) Bottle (b) Jug

Figure 5: Common style encoder reconstructed outputs



(a) Bottle to Jug (b) Jug to Bottle

Figure 6: Common style encoder style transfer outputs

getting more and more compact like a bottle but the original shape (Figure 3b) remains same.

Figure 5 and 6 represent the output when only one style encoder is used. We can see in Figure 5 that the style of the bottle and that of jug is not preserved properly upon reconstruction as compared to Figure 3, that is, the bottle point cloud is not desirably compact and jug not exactly wide. Upon comparing Figure 5 and Figure 6, we see that after style transfer, some content remains the same for the original class but the style changes to that of the other class.

5. Conclusion

Based on our results, we conclude that the style transfer with 2 different style encoders performs better than style transfer with common style encoder. We do not achieve proper results due to lack of data and computational resources but we can still see an extent of style transfer between a jug and a bottle. A stark bottleneck we faced was to be not able to output all 30000 points via our decoder because of computational limitations. So, it certainly is not farfetched to say that with adequately hyperparameterized code and better computational resources, a really good style transfer can be achieved for 3D point clouds using our implementation as an inspiration. We have given our implementation on this GitHub repo - https://github.com/shahjui2000/674_Pro.

References

[1] Xu Cao, Weimin Wang, Katashi Nagao, and Ryosuke Nakamura. Psnet: A style transfer network for point cloud stylization on geometry and color. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 3337–3345, 2020. 1

[2] 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. 2

[3] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks, 2014. 1

[4] Thibault Groueix, Matthew Fisher, Vladimir G. Kim, Bryan C. Russell, and Mathieu Aubry. Atlasnet: A papier-mâché approach to learning 3d surface generation, 2018. 1, 2

[5] Thibault Groueix, Matthew Fisher, Vladimir G Kim, Bryan C Russell, and Mathieu Aubry. Unsupervised cycle-consistent deformation for shape matching. In *Computer Graphics Forum*, volume 38, pages 123–133. Wiley Online Library, 2019. 1

[6] Mattia Segu, Margarita Grinvald, Roland Siegwart, and Federico Tombari. 3dsnet: Unsupervised shape-to-shape 3d style transfer, 2020. 1, 2

[7] Athanasios Voulodimos, Nikolaos Doulamis, Anastasios Doulamis, and Eftychios Protopapadakis. Deep learning for computer vision: A brief review. *Computational intelligence and neuroscience*, 2018, 2018. 1