PSNet: A Style Transfer Network for Point Cloud Stylization on Geometry and Color

Xu Cao* Weimin Wang† Katashi Nagao* Ryosuke Nakamura†
* Graduate School of Informatics, Nagoya University, Japan
† National Institute of Advanced Industrial Science and Technology (AIST), Japan
sou@nagao.nuie.nagoya-u.ac.jp, {weimin.wang, r.nakamura}@aist.go.jp, nagao@nuie.nagoya-u.ac.jp

Abstract

We propose a neural style transfer method for colored point clouds which allows stylizing the geometry and/or color property of a point cloud from another. The stylization is achieved by manipulating the content representations and Gram-based style representations extracted from a pre-trained PointNet-based classification network for colored point clouds. As Gram-based style representation is invariant to the number or the order of points, the style can also be an image in the case of stylizing the color property of a point cloud by merely treating the image as a set of pixels. Experimental results and analysis demonstrate the capability of the proposed method for stylizing a point cloud either from another point cloud or an image.

1. Introduction

Point cloud is an essential 3D representation of the physical world that can be obtained by range sensors like LiDAR or RGB-D camera. Point cloud is used in many real-world applications, e.g. autonomous driving and virtual reality experience. With the increased need of point cloud content, it would be pleasing to if we can edit existing point clouds to enrich point cloud contents. For example, in VR chat, we could change the theme of the room; in 3D animation character design, we could create a series of characters based on a few different characters. However, this task is challenging in that a point cloud usually contains numerous points scattered in 3D space, which is not easy to edit.

In image editing, neural style transfer [7] is a promising method that stylizes a source image by just providing a target image along with a pre-trained ConvNet [19]. The network implicitly encodes the style of the target image and transfers it onto the source image. This method is advantageous in that the source image is modified as a whole; thus no region of interest needs to be designated. Inspired by the idea, we aim to edit a whole point cloud by stylizing it from a target. To this end, a pre-trained neural network that can extract representation for point cloud is required.

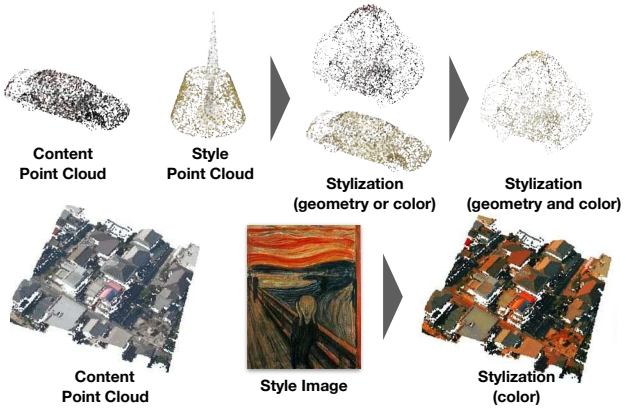


Figure 1: PSNet independently stylizes the geometry or/and color property of a point cloud from a style point cloud. The style can also be an image in the case of stylizing only color property.

Recent years, neural networks that can directly consume point cloud for different tasks, such as classification [5, 16], semantic segmentation [18], and instance segmentation [22], have been studied. PointNet and PointNet-based networks apply a shared MLP to a set of points to expand the dimension of each point and use pooling to aggregate a global feature vector for the point cloud. Therefore, the intermediate output of these networks has a different meaning than that from ConvNets. How we can utilize the intermediate output of PointNet-like networks for stylizing point clouds remains uncertain.

In this study, we propose PSNet, a PointNet-based stylization neural network allowing geometry and/or color property transfer from a point cloud to another. Compared with PointNet, we apply two shared MLPs to extract representation for the geometry and color property of a point

cloud, respectively. We treat the intermediate outputs of the input point cloud as the content representation and implicitly encode the style of the point cloud with Gram-matrix of these intermediate outputs. By matching with the style representation of the geometry and/or the color property of a style point cloud, the geometry and/or the color of the content point cloud is stylized. This method enables editing a point cloud by just providing a target style point cloud. In addition, PSNet can be adopted to stylize the color property of a point cloud from an image by treating the image as a set of pixels (points), as the Gram-based style representation is invariant to the number or the order of input points.

Our contributions can be summarized as follows:

1. We demonstrate that feature representations extracted from a PointNet-based network can be utilized to stylize a colored point cloud.

2. We propose PSNet, a network that can stylize the geometry and/or the color property of a point cloud from another point cloud or even an image; the number of points in the two point clouds need not be the same.

2. Related Work

Neural style transfer between images. Neural style transfer aims at seeking a stylized image that preserves the content of a content image with its style from a style image by utilizing feature representations extracted from neural networks. Gatys et al. [7] propose a style transfer method by iteratively updating an image such that its content representation and style representation extracted from VGG [19] simultaneously match with that of a content image and a style image. This online optimization algorithm suffers from efficiency issue. To address the issue, a bunch of follow-up work additionally trains a neural network which can generate the stylized image with only one feed-forward pass. Depending on how many types of styles can one network generate, this type of method can be classified into three subclasses: per-style-per-model [11, 21], multiple-style-per-model [14, 25] and arbitrary-style-per-model [8].

Gatys et al. [7] utilize the Gram matrix of feature maps to represent the style of an image, and this almost becomes an unsuspicious standard. Li et al. [15] point out the matching of Gram-based style representation is equivalent to minimizing Maximum Mean Discrepancy (MMD) between feature maps of style images and stylized images, and demonstrate other distribution alignment options for style transfer.

Point set representation learning. Learning good representations directly from point clouds in an end-to-end fashion is a challenging task due to its irregular data structure. PointNet [5] is a pioneering work in addressing this problem. To address the variant order issue of an point cloud, PointNet first applies a shared multi-layer perceptron (MLP) to expand each 3D point to higher dimensional feature space, and then use a symmetric function, e.g. max-pooling, to aggregate information along feature axis. This aggregation results in a global feature representing the overall shape of the point cloud. PointNet++ [16] further improved PointNet by recursively applying PointNet to points in local regions to capture local structures at different scales. Similarly, PointCNN [13] designs an X-conv operation to overcome irregularity, and PointSIFT [10] designs an orientation encoding unit to extract multi-scale representations from point clouds. SpiderCNN [23] encodes the spatial difference between a point and its neighbours into filter functions and extend transitional convolutional operation to irregular point sets.

3. Method

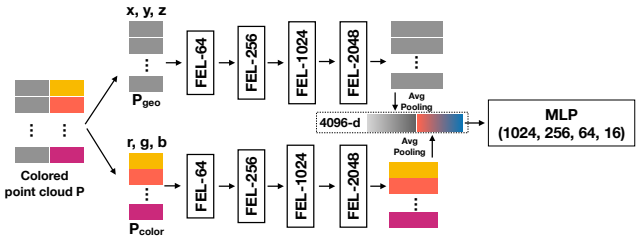
In this section, we first describe the architecture of PSNet for colored point cloud classification; then we explain how to utilize the feature representations extracted from PSNet for style transfer. Finally, we explain the extension of the same method to transfer the style of images onto the color expression of point clouds.

3.1. Classification network for colored point clouds

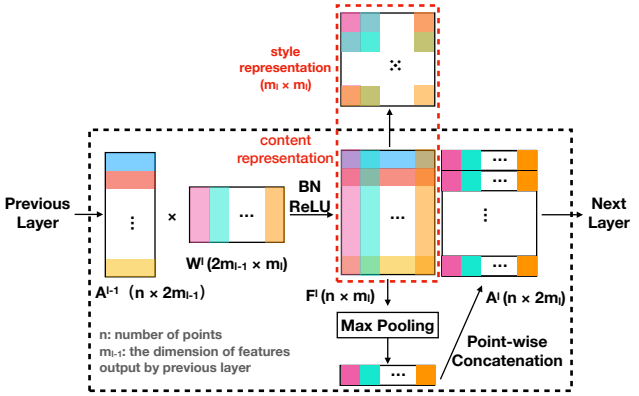
We adopt a PointNet [5] based neural network to extract feature representations for point clouds. To extract a global feature for a point cloud that is invariant to the order of points, PointNet applies shared MLP to all points to extend their dimensions and utilizes a symmetric function (max pooling) to aggregate a global feature vector.

We make two modifications to PointNet for colored point clouds classification. First, instead of applying one shared MLP, we apply two separate MLPs to the geometry and color property of the point cloud, respectively. Without loss of generality, we denote a point cloud $\mathbf{P}$ as a $N \times 6$ matrix, in which each row is its XYZ coordinates and RGB colors, and the identity of $\mathbf{P}$ is invariant to the order of its rows. We split $\mathbf{P}$ into $\mathbf{P}_{\text{geo}}$ and $\mathbf{P}_{\text{color}}$, two $N \times 3$ matrices containing only XYZ coordinates and RGB colors, respectively. Then two global feature vectors independently extracted for $\mathbf{P}_{\text{geo}}$ and $\mathbf{P}_{\text{color}}$ are concatenated into one 4096-d vector, as shown in Fig. 2(a).

Second, as shown in the black dashed box in Fig. 2(b), for the output of each shared MLP, we extract a global feature vector and concatenate it to each feature. This operation results in features containing both its local information and global information. We term the modified layer as feature encoding layer (FEL). The number after FEL in Fig. 2(a) denotes the dimension of each feature after being passed that FEL. Suppose $\mathbf{A}^{l-1}$ is the input of the l^{th} FEL, then $\mathbf{A}^l$ is obtained by $\text{FEL}(\mathbf{A}^{l-1}; W^l)$, and $\mathbf{A}^0$ is $\mathbf{P}_{\text{geo}}$ or $\mathbf{P}_{\text{color}}$. In our implementation, l is up to 4.



(a)



(b)

Figure 2: Network architecture for feature extraction. (a) The geometry and color property of a point cloud are separately processed by a stack of FELs; then two global features are extracted and concatenated into one 4096-d vector, which was further processed by a MLP. (b) Feature Encoding Layer (FEL). The numbers within brackets denote the dimensions of matrices. We first compute F^l by multiplying each row of the input A^{l-1} with the weight matrix W^l. Then, the maximum vector of F^l is calculated and concatenated to each row of F^l. This concatenation results in A^l, the output of this layer and the input to the next layer. F^l is taken as content representation while the Gram matrix of F^l is taken as style representation.

3.2. Style transfer between point clouds

To perform style transfer between point clouds, we follow the idea of neural style transfer [7]. Given a pre-trained feature extractor, a content point cloud $\mathbf{C} \in \mathbb{R}^{N_c \times 6}$ and a style point cloud $\mathbf{S} \in \mathbb{R}^{N_s \times 6}$, we try to seek a stylized point cloud $\mathbf{P} \in \mathbb{R}^{N_p \times 6}$ that minimizes the following loss function:

$$\begin{aligned}\mathbf{P}^*_{\text{geo}} = \arg\min_{\mathbf{P}_{\text{geo}}} & L_{\text{geo\_content}}(\mathbf{P}_{\text{geo}}, \mathbf{C}_{\text{geo}}) \\ & + \alpha_{\text{geo}} L_{\text{geo\_style}}(\mathbf{P}_{\text{geo}}, \mathbf{S}_{\text{geo}})\end{aligned} \tag{1}$$

$$\begin{aligned}\mathbf{P}^*_{\text{color}} = \arg\min_{\mathbf{P}_{\text{color}}} & L_{\text{color\_content}}(\mathbf{P}_{\text{color}}, \mathbf{C}_{\text{color}}) \\ & + \alpha_{\text{color}} L_{\text{color\_style}}(\mathbf{P}_{\text{color}}, \mathbf{S}_{\text{color}}),\end{aligned} \tag{2}$$

where $L_{\text{\_content}}$ measure the difference between the content representation of $\mathbf{P}$ and $\mathbf{C}$, and $L_{\text{\_style}}$ compares the style representation of $\mathbf{P}$ and $\mathbf{S}$. α_{geo} or α_{color} balance the content component and style component in the stylized point cloud.

Let $F^l(\cdot)$ denote the feature response directly after the activation function in the l^{th} FEL. The number of row in $F^l(\cdot)$ is decided by that of the input point cloud, and the dimension of each row is m_l. This feature response is considered as the content representation of the point cloud, as shown in the red dashed box in Fig. 2(b). We define the content loss for geometry and color independently as follows:

$$\begin{aligned}L_{\text{geo\_content}}(\mathbf{P}_{\text{geo}}, \mathbf{C}_{\text{geo}}) = \\ \sum_{l \in \{l_c\}} \|F^l(\mathbf{P}_{\text{geo}}) - F^l(\mathbf{C}_{\text{geo}})\|_2^2\end{aligned} \tag{3}$$

$$\begin{aligned}L_{\text{color\_content}}(\mathbf{P}_{\text{color}}, \mathbf{C}_{\text{color}}) = \\ \sum_{l \in \{l_c\}} \|F^l(\mathbf{P}_{\text{color}}) - F^l(\mathbf{C}_{\text{color}})\|_2^2,\end{aligned} \tag{4}$$

where $\{l_c\}$ denotes the set of FEL layers from which we extract feature representations for computing content loss.

The style representation in l^{th} FEL is the Gram matrix of $F^l(\cdot)$, as shown in the red dashed box in Fig. 2(b). Next, we explain how this works for feature representation extracted by FELs. Let $F^l_{:j}(\cdot)$ denote the j^{th} column in $F^l(\cdot)$. Each element in this column is the inner product between the j-th column in the weight matrix W^l and its corresponding row in $F^l(\cdot)$. Thus, the j^{th} column in W^l can be viewed as a filter, and $F^l_{:j}$ is the response to that filter. In other words, the j^{th} column in W^l and $F^l_{:j}$ is analogous to a convolution kernel and the feature map to that kernel in convolutional neural networks.

Taking the analogy into consideration, we compute $G_{ij}(F^l(\cdot))$, the ij^{th} element of the Gram matrix for $F^l(\cdot)$ as the inner product between the i^{th} column and j^{th} column of $F^l(\cdot)$:

$$G_{ij}(F(\cdot)) = \mathbf{f}_i^\top \mathbf{f}_j.$$

We denote $G(F^l(\cdot)) \in \mathbb{R}^{m_l \times m_l}$ as the whole Gram matrix, of which the dimension is solely decided by the number of columns in the weight matrix W^l.

The style loss for geometry and color is defined independently as follows:

$$\begin{aligned}L_{\text{geo\_style}}(\mathbf{P}_{\text{geo}}, \mathbf{S}_{\text{geo}}) = \\ \sum_{l \in \{l_s\}} \|G(F^l(\mathbf{P}_{\text{geo}})) - G(F^l(\mathbf{S}_{\text{geo}}))\|_2^2\end{aligned} \tag{5}$$

$$\begin{aligned}L_{\text{color\_style}}(\mathbf{P}_{\text{color}}, \mathbf{S}_{\text{color}}) = \\ \sum_{l \in \{l_s\}} \|G(F^l(\mathbf{P}_{\text{color}})) - G(F^l(\mathbf{S}_{\text{color}}))\|_2^2,\end{aligned} \tag{6}$$

where $\{l_s\}$ denotes the set of FEL layers from which we extract feature representations to compute the style loss.

To seek $\mathbf{P}^*_{\text{geo}}$ or $\mathbf{P}^*_{\text{color}}$, we adopt a gradient-based optimization method, e.g. Adam optimizer [12], to iteratively update $\mathbf{P}_{\text{geo}}$ or $\mathbf{P}_{\text{color}}$ until its convergence.

3.3. Style transfer from images onto point clouds

To transfer the style of an image to the color property of a content point cloud $\mathbf{C}_{\text{color}}$, we merely treat an image as a set of pixel values $\mathbf{S}_{\text{color}}$ and follow Eq. (2) to seek the stylized point cloud $\mathbf{P}_{\text{color}}$. This method works due to two advantageous properties of Gram matrix. First, the dimension of Gram matrix is solely decided by the number of columns in the weight matrix and irrelevant to the number of points. That is, the difference in the number of points in $\mathbf{S}$ and $\mathbf{P}$ doesn't disable the computation of Eq. (6). Second, Gram matrix $G(F^l(\mathbf{S}_{\text{color}}))$ is invariant to the order of rows in $\mathbf{S}_{\text{color}}$. According to the two properties, by simply reshaping an image of dimension $H \times W \times 3$ into dimension $(H * W) \times 3$, the same method discussed in Sec.3.2 can be used to transfer the style of an image onto the color property of a point cloud.

Although this reshaping operation discards the spatial structure information of the image, experimental results show that the color distribution is captured and well transferred onto $\mathbf{C}_{\text{color}}$.

We summarize the transferable properties between content and style point in Table **??** to clarify the flexibility of our method.

4. Experiments

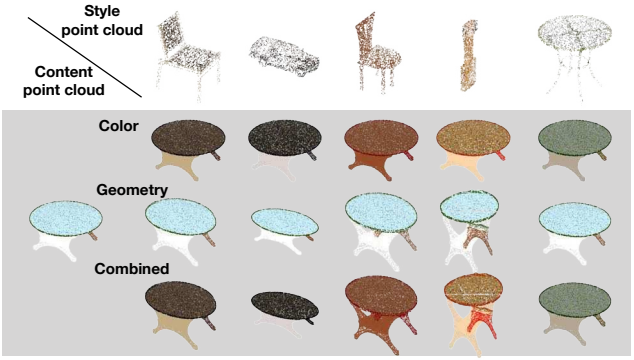
In Sec.4.1, we give training details of our proposed model for extracting representations of colored point clouds. In Sec.4.2, we demonstrate qualitative results of the style transfer method. In Sec.4.3, we analyze different factors affecting style transfer results. In Sec.4.4, we perform ablation study on our model design.

4.1. Model setup for feature extraction

Dataset. To train a model that can encode both geometry and color of a point cloud, we use DensePoint dataset [3]. DensePoint is point cloud dataset built on top of ShapeNet [4] and ShapeNetPart Dataset [24]. There are 10454 colored point clouds of single objects across 16 categories; the dataset is split into 7612/1112/1730 training/validation/test subsets according to ShapeNet's setting.

Training settings. DensePoint suffers from class imbalance problem. For example, there are over 3000 instances for table class while there are only less than 100 instances for several other classes. To remedy the problem, we use a combination of undersampling and oversampling techniques [2]: we undersample or oversample 320 instances for classes in which instances are over or under 320, respectively. For the value of XYZ coordinate and RGB colors in point clouds, we normalize them into the range between -1 and 1.

We train our model for 50 epochs with batch size 32. We use Adam optimizer [12] with initial learning rate 0.01, β_1 0.9 and β_2 0.999. Batch normalization [9] is applied before activation functions in all layers except the last layer. Leaky ReLU with a fixed leakiness parameter 0.2 is used as activation functions. Dropout [20] with keep ratio 0.7 is applied on last three fully-connected layers.



(a)

(b)

Figure 3: Style transfer results between single object point clouds. (a) Either the geometry, the color property or both is stylized, and the points number of style point clouds is arbitrary. (b) More results where both the geometry and color property are stylized.

4.2. Qualitative results of neural style transfer

With the pre-trained network, we utilize it to extract content or style representations for point cloud style transfer. In this section, we demonstrate qualitative results of the proposed style transfer method.

4.2.1 Style transfer between single object point clouds

Given a content point $\mathbf{C}$ and a style point cloud $\mathbf{S}$, three factors affect the resulted point cloud $\mathbf{P}$: the initialization

strategy of $\mathbf{P}$, the value of α and the layers used for extracting representations. In this experiment, we initialize $\mathbf{P}$ as $\mathbf{C}$, set $\alpha_{\text{geo}} = 1, \alpha_{\text{color}} = 100$, and extract content and style representations both from {FEL-64}. We adopt Adam optimizer with learning rate 0.01 to update $\mathbf{P}$ and empirically set the update steps as 4000. The results are shown in Fig. 3. In Fig. 3(a), we demonstrate two advantageous point of our method. First, the geometry or color property of a point cloud is independently stylized. Second, there is no constraint on the number of points in $\mathbf{C}$ and $\mathbf{S}$. In this experiment, $\mathbf{S}$ contains 4096 points while $\mathbf{C}$ contains 40k points. We give more qualitative results in Fig. 3(b). We find that in the case of geometry transfer, the overall shape style of $\mathbf{S}$ is transferred; in the case of color transfer, the content color pattern is preserved while the overall color distribution is shifted towards that of style point clouds.

4.2.2 Stylizing color property from images

In this experiment, we stylize the color expression of content point clouds $\mathbf{C}_{\text{color}}$ either of single objects or indoor scenes from images. The style image is reshaped to $(H * W) \times 3$ and being treated as $\mathbf{S}_{\text{color}}$. In both cases, content representation is extracted from {FEL-64} and style representations are extracted from {FEL-1024, FEL-2048}. α_{color} is 100, and $\mathbf{P}_{\text{color}}$ is initialized as $\mathbf{C}_{\text{color}}$. Qualitative results on point clouds of single objects are given in Fig.4(a). The point cloud of indoor scenes is from S3DIS dataset [1]. Adam optimizer with learning rate 0.001 is utilized to update $\mathbf{P}_{\text{color}}$ for 30000 steps. Quantitative results are given in Fig. 4(b).

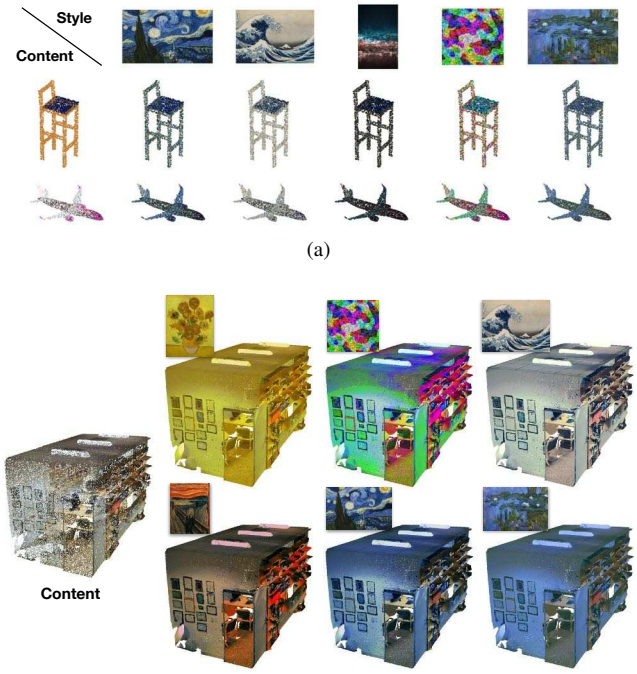
4.2.3 Style transfer loss

To confirm whether the iterative update process eventually converge, we visualize the style transfer loss in Fig. 5. Figure 5(a) illustrates an example of the change in the loss during the stylization between single object point clouds in Fig. 3(a). We can find that the style transfer loss converge fast to stability within just 200 iterations. Since $\mathbf{P}$ is initialized as $\mathbf{C}$, the content loss is 0 at the beginning and increases to stability.

The loss of transferring from images onto color expression of point clouds is given in Fig. 5(b). In this case, the fact that indoor scene point clouds comprise much more points detains the update process. This difficulty is the reason why we decrease the initial learning rate to 0.001 and increase the iteration steps to 30000 in Sec.4.2.2.

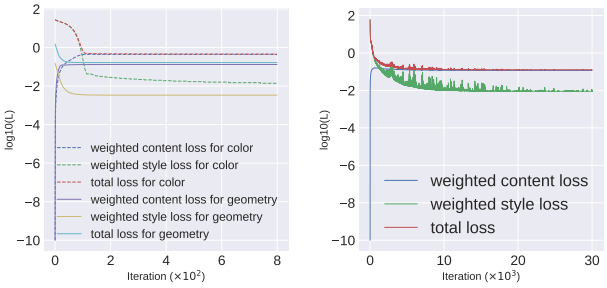
4.3. Analysis of style transfer

According to Eq. (1) or Eq. (2), given a pre-trained network for feature extraction, four factors are likely to affect the stylized point cloud: the weight of content loss and style



(a) (b)

Figure 4: Style transfer results from images onto point clouds. (a) Single object point clouds with 4096 points. (b) Indoor scene point clouds over 800k points. Both results demonstrate that the color pattern of point clouds (e.g. darker seat of the chair) are preserved while the color distribution is more like that in the style image.



(a) (b)

Figure 5: Change of style transfer loss. (a) Style transfer on the geometry and color property between single objects. (b) Style transfer on the color property of indoor scene point clouds. In both cases, the total losses converge fast to stability.

loss, the layer used to extract feature representation, the initialization strategy of $\mathbf{P}$ and the choice of the optimizer to

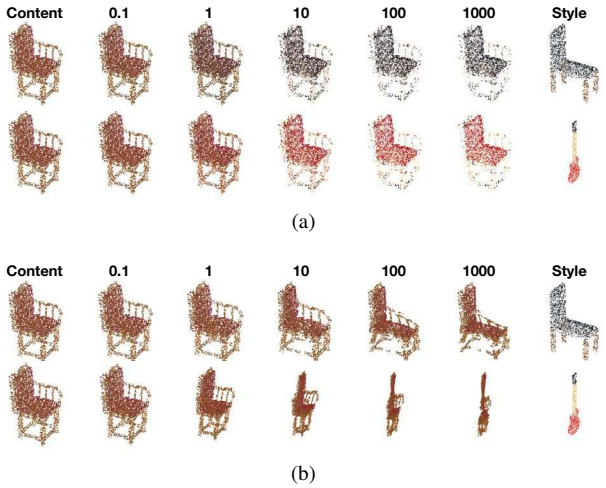


Figure 6: Style transfer results with different α. (a) Transfer on color property with different α_{color}. (b) Transfer on geometry property with different α_{geo}.

seek $\mathbf{P}^*_{\text{geo}}$ or $\mathbf{P}^*_{\text{color}}$. In this section, we investigate their effects on the stylized point clouds.

4.3.1 Content-style trade-off

Intuitively, different ratios between the content loss and style loss may decide how much the stylized point cloud look like the content or the style point cloud. We extract the content representation and style representation both from {FEL-64}; $\mathbf{P}$ is initialized as $\mathbf{C}$. In the first experiment, we fix $\mathbf{P}_{\text{geo}}$ as $\mathbf{C}_{\text{geo}}$, and $\mathbf{P}_{\text{color}}$ is to be stylized. We vary α_{color} among 0.1, 1, 10, 100 and 1000. The results are shown in Fig. 6(a).

In the second experiment, we fix $\mathbf{P}_{\text{color}}$ as $\mathbf{C}_{\text{color}}$, and $\mathbf{P}_{\text{geo}}$ is to be stylized. α_{geo} is varied among 0.1, 1, 10, 100 and 1000. The results are given in Fig. 6(b). From the results in both experiments, we can see that as α_{geo} increases, the style of $\mathbf{S}$ is more evidently transferred onto $\mathbf{P}$ while its content is preserved. In the case of color property, the color pattern of the content chair point cloud (darker seat) is preserved in all cases while its overall color distributions become more similar to that of $\mathbf{S}_{\text{color}}$.

4.3.2 Content/style representation from different layers

In this experiment, we aim at inspecting the effect of target feature extracted from different layers on the style transfer results. Explicitly, we compute style loss and content loss based on features extracted from only one FEL but enumerate all their combinations. There are 4 FELs from which we can extract content or style representation, resulting in

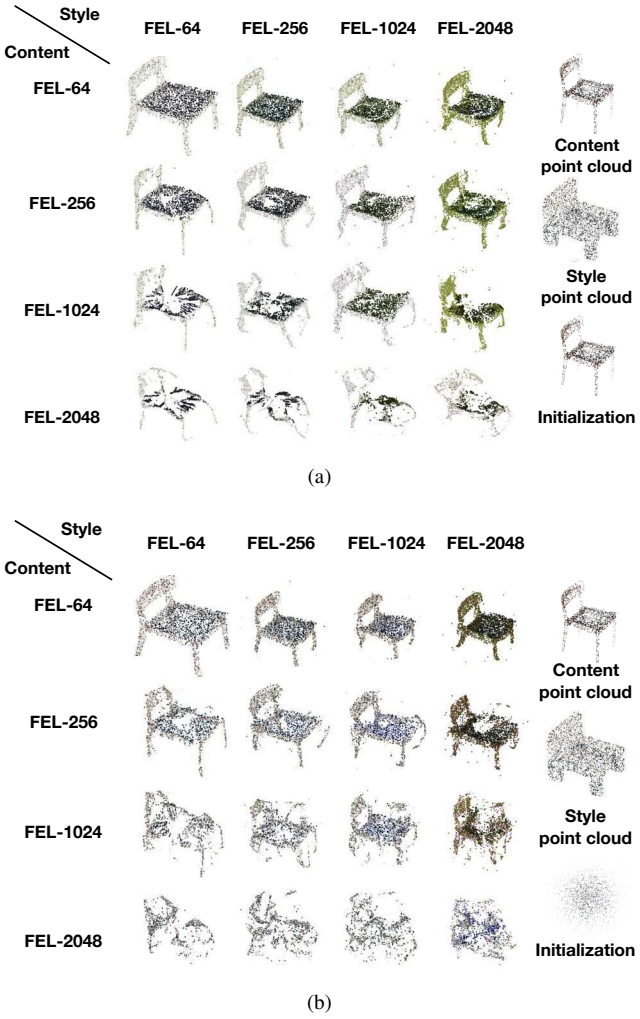


Figure 7: Style transfer results with content/style representation extracted from different layers and different initialization strategy. (a) Initialize $\mathbf{P}$ as content point cloud. (b) Randomly initialize $\mathbf{P}$. Initializing $\mathbf{P}$ as $\mathbf{C}$ helps maintaining the shape.

16 combinations. We set α_{color} and α_{geo} as 100 and 10, respectively. The results are shown in Fig. 7(a). The results in a row are computed from the same content representation while those in the same column are computed from the same style representation; the column or the row name denotes the layer name. As we can see, when content representations are extracted from low layers, the generated point cloud preserve a distinct shape; when content representations extracted from high layers, the overall shape of generated point clouds become vague.

In the case of style representation, when they are extracted from low layers, the point cloud is more stylized; when they are extracted from high layers, the stylized point

clouds again become more similar to the content point cloud, and noise-like points appear.

4.3.3 Initialization strategy

As the stylization is an iterative update process on $\mathbf{P}$, different initialization strategy is likely to lead to different $\mathbf{P}^{\star}$. So far we have initialized $\mathbf{P}$ as $\mathbf{C}$, we compare the initialization strategy with an alternative initialization method: every element of $\mathbf{P}$ is sampled from $\mathcal{N}(0, 0.5)$. We conduct the same experiment as in 4.3.2 except for the different initialization strategy, as shown in Fig. 7(b).

Compare to Fig. 7(a), we find that when feature representations are extracted from low layers, different initialization strategies result in almost the same stylized point cloud, however, initializing $\mathbf{P}$ as $\mathbf{C}$ helps $\mathbf{P}$ maintain a distinct shape. When $\mathbf{P}$ is randomly initialized, and feature representations are extracted from high layers, the result becomes blurred and unrecognizable, but its counterpart in Fig. 7(a) still have a relatively clear shape.

4.3.4 Choice of optimizer

Different optimizers with different learning rate affect the time for $\mathbf{P}$ to converge and may result in different stylized point clouds. In this experiment, we utilized different optimizers from the combination of the optimizer in {SGD, Momentum [17], Adagrad [6], RMSprop, Adam [12]} and the learning rate in $\{0.01, 0.1, 1\}$. We filter out those does not converge after 30k iterations either because of slow learning rate or fluctuation, and illustrate the style transfer loss curve of the remaining in Fig. 8(a) and (b). It is evident that whatever the speed to converge, the final content loss of geometry is almost the same.

We give a qualitative comparison of the stylized point cloud by different optimizers in Fig. 8(c). We find that fast convergence does not mean a better quality of the stylized point clouds. As shown in the red circle of Fig. 8(c), cracks appear in the point clouds updated by fast-converged optimizers, e.g. Adam. In contrast, the problem is alleviated or disappears in the point clouds updated by slow-converged optimizers, e.g. SGD. We consider the reason of the appearance of these cracks is likely to be that fast update encourages the optimizer to merely translate different parts of a point cloud along different directions to minimize the style loss.

4.4. Ablation study on model architecture

As described in Sec.3.1, we add two modifications to PointNet: the separation of different kind of properties of point clouds (late-fusion) and the replacement of shared fully connected layer with FEL. In this section, we evaluate the effectiveness of our modifications for point cloud style transfer.

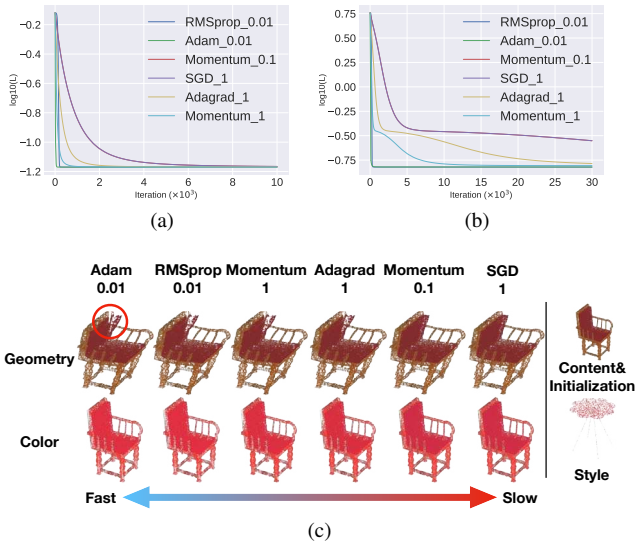


Figure 8: Stylize point clouds with different optimizers. (a) Style loss curve of geometry. (b) Style loss curve of color. (c) Qualitative comparison of stylized point clouds updated by different optimizers. There is no apparent difference in overall shape or color distribution. However, cracks appear in point clouds updated by fast-converged optimizers.

model type	accuracy	multiclass AUC
early-fusion + FEL	90.54	98.49
late-fusion	87.50	96.56
late-fusion + FEL	**90.80**	**98.85**

Table 1: Classification performance of different model choices on DensePoint test set [3].

Comparison on classification performance. Although the primary goal of modifying PointNet is to extract better representations for style transfer, we are still interested in whether the two modifications improve the model performance on colored point clouds classification task. We compare our final model with two alternative models: the model directly consuming colored point cloud without splitting it (early-fusion) and the model without using FEL (the same as PointNet except for late-fusion). We report the performance of these models in Table 1 based on two evaluation metrics: accuracy and multiclass Area Under the ROC Curve (AUC). According to Table 1, our final model achieves the best performance, which validates our design.

Late fusion vs. early fusion. We design a two-route network to independently process the geometry and color property of point clouds and concatenate extracted two global

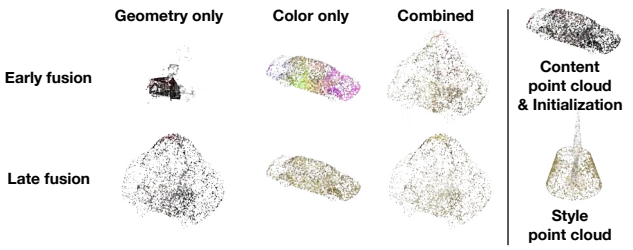


Figure 9: Style transfer results comparison on different model architecture designs.

feature vectors later (late fusion). An alternative network design is a one-route network directly consuming colored point clouds which are not split into two parts (early fusion). In the latter case, we cannot extract feature representations separately for geometry or color property; Eq. (3) and Eq. (4) now read

$$L_{\text{content}}(\mathbf{P}, \mathbf{C}) = \sum_{l \in \{l_c\}} \|F^l(\mathbf{P}) - F^l(\mathbf{C})\|_2^2.$$

Equation (5) and Eq. (6) now becomes

$$L_{\text{style}}(\mathbf{P}, \mathbf{S}) = \sum_{l \in \{l_s\}} \|G(F^l(\mathbf{P})) - G(F^l(\mathbf{S}))\|_2^2$$

We can still opt to update only the geometry or color property and fix another part during the update process:

$$\mathbf{P}^*_{\text{geo}} = \arg\min_{\mathbf{P}_{\text{geo}}} L_{\text{content}}(\mathbf{P}, \mathbf{C}) + \alpha L_{\text{style}}(\mathbf{P}, \mathbf{S}),$$

where $\mathbf{P}_{\text{geo}}$ can be replaced by $\mathbf{P}_{\text{color}}$ or $\mathbf{P}$ depending on whether we want to modify the geometry property, the color property or both of the point cloud $\mathbf{P}$.

We conduct a comparison experiment between the two models to inspect their effects on the stylization results, as shown in Fig. 9. In both cases, content/style representations are extracted from the first layer, and $\mathbf{P}$ is initialized as $\mathbf{C}$. In the case of early fusion network, $\alpha = 100$;in the case of late fusion network, $\alpha_{\text{geo}} = 100, \alpha_{\text{color}} = 1000$.

As we can see in Fig.9, in the case of early fusion network, updating the geometry or the color property leads to absurd results. This is not surprising since the extracted representation depends on both the geometry and color property of $\mathbf{P}$. When the geometry and the color property are jointly updated, the result is close to that from late fusion network, but still not appealing. In contrast, by utilizing a late fusion network, either the geometry or the color property can be stylized, resulting in appealing stylized point clouds. The comparison demonstrates the effectiveness and flexibility of our proposed late fusion design.

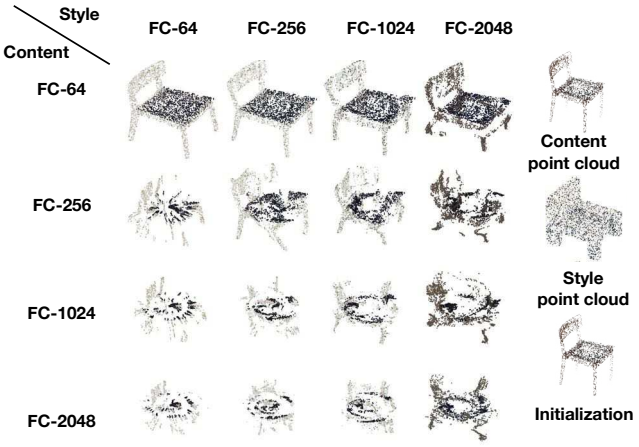


Figure 10: Style transfer results without FEL.

FEL vs. shared FC. When the point-wise concatenation operation in Fig. 2(b) is not performed, a feature encoding layer degrades to a naive shared fully connected (shared FC) layer as in PointNet. We can still extract feature representations in this design and perform style transfer in the same way as described in Sec.3.2. The same experiment as in Sec.4.3.2 is conducted, as shown in Fig. 10. We can find that when content representations are extracted from the first layer, the quality of stylized point clouds is comparable to that in Fig. 7(a). However, when the content representation is extracted from higher layers, the quality dropped drastically wherever the style representation is from. This comparison validates the robustness of our design of replacing shared FCs in a naive PointNet with FELs.

5. Conclusion

We proposed PSNet, a style transfer network for colored point clouds. PSNet stylize the geometry and/or color property of a point cloud from another. The style can be an image when stylizing the color property of a point cloud. We investigated the effect of different hyperparameters in our method on the stylized point cloud. As the method can be applied to point clouds containing any number of points, it opens a door for stylizing large-scale indoor scene or terrain point clouds for virtual reality applications in the future.

6. Acknowledgements

This paper is based on results obtained from a project commissioned by the New Energy and Industrial Technology Development Organization (NEDO) and an internal grant of National Institute of Advanced Industrial Science and Technology (AIST).

References

[1] I. Armeni, O. Sener, A. R. Zamir, H. Jiang, I. Brilakis, M. Fischer, and S. Savarese. 3d semantic parsing of large-scale indoor spaces. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[2] M. Buda, A. Maki, and M. A. Mazurowski. A systematic study of the class imbalance problem in convolutional neural networks. *Neural Networks*, pages 249–259, 2018.

[3] X. Cao and K. Nagao. Point cloud colorization based on densely annotated 3d shape dataset. In *MultiMedia Modeling (MMM)*, pages 436–446, 2019.

[4] A. X. Chang, T. Funkhouser, L. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su, J. Xiao, L. Yi, and F. Yu. ShapeNet: An Information-Rich 3D Model Repository. Technical report, 2015.

[5] R. Q. Charles, H. Su, M. Kaichun, and L. J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 77–85, 2017.

[6] J. Duchi, E. Hazan, and Y. Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(Jul):2121–2159, 2011.

[7] L. A. Gatys, A. S. Ecker, and M. Bethge. Image style transfer using convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2414–2423, 2016.

[8] X. Huang and S. Belongie. Arbitrary style transfer in real-time with adaptive instance normalization. In *Proceedings of the International Conference on Computer Vision (ICCV)*, pages 1510–1519. IEEE, 2017.

[9] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[10] M. Jiang, Y. Wu, and C. Lu. Pointsift: A sift-like network module for 3d point cloud semantic segmentation. *arXiv preprint arXiv:1807.00652*, 2018.

[11] J. Johnson, A. Alahi, and L. Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 694–711, 2016.

[12] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014.

[13] Y. Li, R. Bu, M. Sun, W. Wu, X. Di, and B. Chen. Pointcnn: Convolution on x-transformed points. In *Advances in Neural Information Processing Systems (NIPS)*, pages 826–836, 2018.

[14] Y. Li, C. Fang, J. Yang, Z. Wang, X. Lu, and M.-H. Yang. Diversified texture synthesis with feed-forward networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[15] Y. Li, N. Wang, J. Liu, and X. Hou. Demystifying neural style transfer. *arXiv preprint arXiv:1701.01036*, 2017.

[16] C. R. Qi, L. Yi, H. Su, and L. J. Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *Advances in Neural Information Processing Systems (NIPS)*, pages 5099–5108, 2017.

[17] N. Qian. On the momentum term in gradient descent learning algorithms. *Neural networks*, 12(1):145–151, 1999.

[18] D. Rethage, J. Wald, J. Sturm, N. Navab, and F. Tombari. Fully-convolutional point networks for large-scale point clouds. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018.

[19] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[20] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, pages 1929–1958, 2014.

[21] D. Ulyanov, V. Lebedev, A. Vedaldi, and V. S. Lempitsky. Texture networks: Feed-forward synthesis of textures and stylized images. In *ICML*, pages 1349–1357, 2016.

[22] W. Wang, R. Yu, Q. Huang, and U. Neumann. Sgpn: Similarity group proposal network for 3d point cloud instance segmentation. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2569–2578, 2018.

[23] Y. Xu, T. Fan, M. Xu, L. Zeng, and Y. Qiao. Spidercnn: Deep learning on point sets with parameterized convolutional filters. In *Proceedings of the European Conference on Computer Vision (ECCV)*, September 2018.

[24] L. Yi, V. G. Kim, D. Ceylan, I.-C. Shen, M. Yan, H. Su, C. Lu, Q. Huang, A. Sheffer, and L. Guibas. A scalable active framework for region annotation in 3d shape collections. *SIGGRAPH Asia*, 2016.

[25] H. Zhang and K. Dana. Multi-style generative network for real-time transfer. *arXiv preprint arXiv:1703.06953*, 2017.