Learning to Stylize Novel Views

Hsin-Ping Huang[1], Hung-Yu Tseng[1], Saurabh Saini[2], Maneesh Singh[2], Ming-Hsuan Yang[1,3,4]
[1]UC Merced [2]Verisk Analytics [3]Google Research [4]Yonsei University

https://hhsinping.github.io/3d_scene_stylization

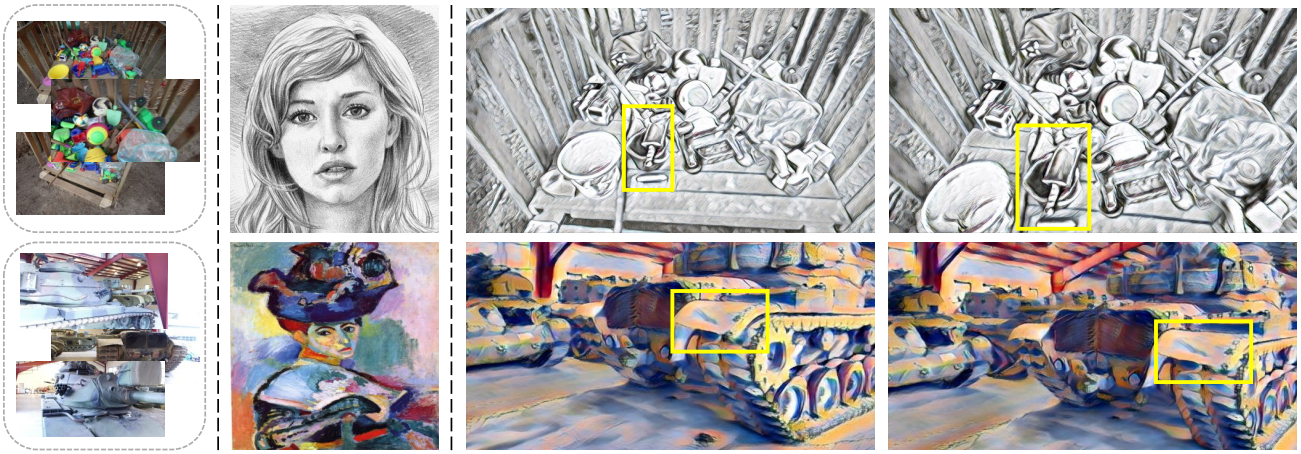

Figure 1. **3D scene stylization**. Given a set of images of a 3D scene (*left*) as well as a reference image of the desired style (*middle*), our method is able to modify the style of the 3D scene, and synthesize images of arbitrary novel views (*right*). The novel view synthesis results 1) contain the desired style and 2) are consistent across various novel views, *e.g.* the texture in the yellow boxes.

Abstract

We tackle a 3D scene stylization problem — generating stylized images of a scene from arbitrary novel views given a set of images of the same scene and a reference image of the desired style as inputs. Direct solution of combining novel view synthesis and stylization approaches lead to results that are blurry or not consistent across different views. We propose a point cloud-based method for consistent 3D scene stylization. First, we construct the point cloud by back-projecting the image features to the 3D space. Second, we develop point cloud aggregation modules to gather the style information of the 3D scene, and then modulate the features in the point cloud with a linear transformation matrix. Finally, we project the transformed features to 2D space to obtain the novel views. Experimental results on two diverse datasets of real-world scenes validate that our method generates consistent stylized novel view synthesis results against other alternative approaches.

1. Introduction

Visual content creation in 3D space has recently attracted increasing attention. Driven by the success of 3D scene representation approaches [38, 46, 64], recent methods make significant progress on various content creation tasks for 3D scenes, such as semantic view synthesis [16, 19] and scene extrapolation [34]. In this work, we focus on the *3D scene stylization* problem. As shown in Figure 1, given a set of images of a target scene and a reference image of the desired style, our goal is to render stylized images of the scene from arbitrary novel views. 3D scene stylization enables a variety of interesting virtual reality (VR) and augmented reality (AR) applications, *e.g.* augment the street scene at user locations to the *Cafe Terrace at Night* style by van Gogh.

Learning to modify the style of an existing 3D scene is challenging for two reasons. First, the synthesized novel views (*i.e.* 2D images) of the stylized 3D scene must contain the desired style provided by the reference image. Second, since our goal is to stylize the *holistic* 3D scene, the generated novel views need to be consistent across different viewpoints for the same scene, such as the texture in the yellow boxes shown in Figure 1.

To handle these challenges, one plausible solution is to combine existing novel view synthesis [46, 64] and image stylization approaches [30, 53]. However, such straightforward approaches lead to problematic results since image stylization schemes are not designed to consider the consistency issue across different views for the same scene.

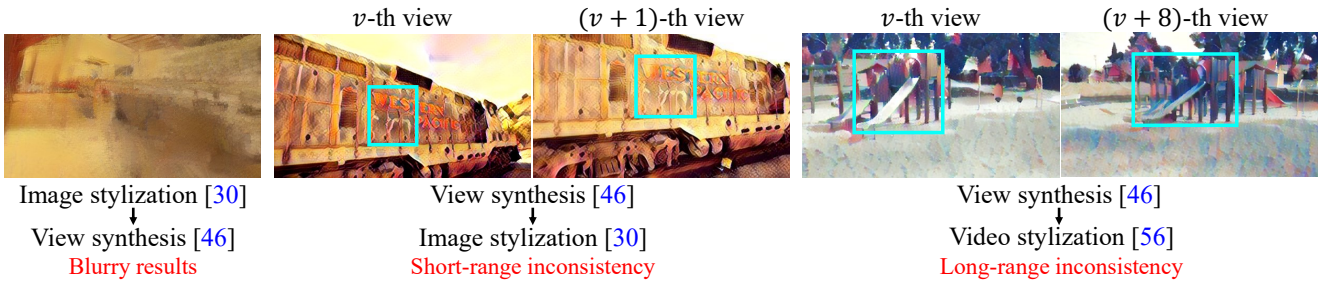

Figure 2. **Motivation.** While the existing methods can be used for the 3D scene stylization task, these methods either produce blurry (image stylization → novel view synthesis), short-range inconsistent (novel view synthesis → image stylization), or long-range inconsistent (novel view synthesis → video stylization) results.

We present the examples in Figure 2 where the results may be blurry if the input images of the target scene are stylized before conducting novel view synthesis. On the other hand, if we apply image stylization after novel view synthesis, the results are not consistent across different views. Another possible solution is to treat a series of novel view synthesis results as a *video*, and use the video stylization frameworks [9, 11, 56] to obtain temporally consistent results. However, as shown in Figure 2, these approaches are not able to enforce long-range consistency (*i.e*. between two far-away views) as the video stylization schemes only guarantee the short-term consistency.

In this paper, we propose a point cloud-based method for consistent 3D scene stylization. To synthesize novel views that 1) match *arbitrary* style images and 2) render images with consistent appearance across different views, the core idea is to operate on the 3D scene representation, *i.e*. point cloud, of the target scene. Given a set of input images of the target scene, we first construct the point cloud by back-projecting the image features to the 3D space according to the pre-computed 3D proxy geometry. To transfer the style of the holistic 3D scene, we develop a point cloud transformation module. Specifically, we use a series of point cloud aggregation modules to gather the style information of the 3D scene. We then modulate the features in the point cloud with a linear transformation matrix [30] computed according to the style information of the point cloud and reference image. Finally, we project the transformed features from the point cloud to the 2D space to obtain the novel view synthesis results. Since our method synthesizes novel view images from the same stylized point cloud, the rendered results not only demonstrate the desired style, but also are consistent across different viewpoints.

We evaluate the proposed 3D scene stylization method through extensive qualitative and quantitative studies. The experiments are conducted on two diverse datasets of real-world scenes: Tanks and Temples [25] and FVS [45]. We conduct a user preference study to evaluate the stylization quality, *i.e*. whether the novel view synthesis results match the style of the reference image. In addition, we use the Learned Perceptual Image Patch Similarity (LPIPS) [65] metric to measure the consistency of the results synthesized across different novel views.

We make the following contributions in this paper:

- We propose a point cloud-based framework for the 3D scene stylization task.
- We design a point cloud transformation module that learns to transfer the style from an arbitrary 2D reference image to the point cloud of a 3D scene.
- We validate that our method produces high-quality and consistent stylized novel view synthesis results on the Tanks and Temples as well as FVS datasets.

2. Related Work

Novel View Synthesis. Given a set of images for a scene, novel view synthesis aims to generate high-quality images at arbitrary viewpoints. It can be categorized by the number of input images that cover the scene. One line of work takes as input a single image or stereo images. These methods use multi-plane images [52, 54, 58, 67], layer depth image [27, 50], or point cloud [40, 57] representations to synthesize images at novel views near the input views, *e.g*. 3D photo. To enable the image synthesis at *arbitrary* novel views, several recent frameworks take hundreds of input images of a scene as the input. These frameworks leverage different 3D representations to accomplish the task. Image-based rendering approaches [45, 46] compute 3D proxy geometry of the scene, and generate images by warping the input frames to the desired novel views. Neural radiance field schemes [35, 38, 63, 64] use multi-layer perceptrons to implicitly encode the scene for novel view synthesis. Point cloud-based methods [37, 1] solve different optimization problems to construct the point cloud for a specific 3D scene. Different from these frameworks, our goal is to generate *stylized* novel view images of the 3D scene. As shown in Figure 2, while existing algorithms can be used for the 3D scene stylization task, they fail to generate high-quality novel view synthesis results with the desired style.

Image and Video Stylization. Image stylization [12] aims to transfer the style of a reference image to the single input image. Existing methods [5, 22, 32, 49, 55] are designed based on feed-forward networks for transferring a set of *pre-defined* styles. For *arbitrary* image style transfer,

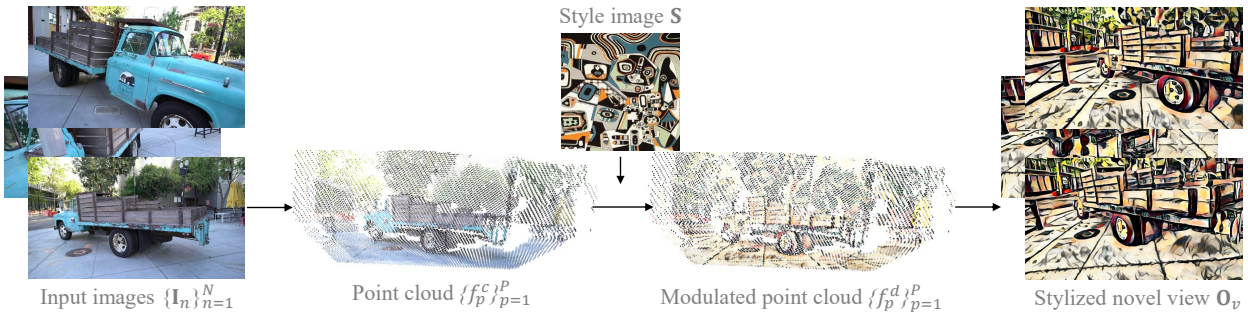

Figure 3. **Algorithmic overview.** The proposed method consists of three steps: 1) constructing the 3D point cloud from the set of input images $\{\mathbf{I}_n\}_{n=1}^N$, 2) transforming the point cloud according to the reference image $\mathbf{S}$ with the desired style, and 3) synthesizing the stylized image $\mathbf{O}_v$ at arbitrary novel view v. The coloring of the point clouds is for visualization purposes only. In our approach, the point clouds store the features rather than RGB values.

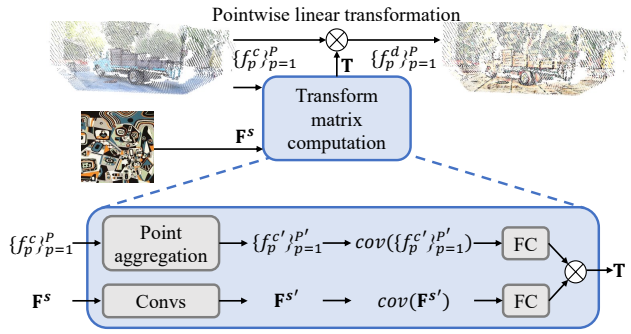

Figure 4. **Point cloud transformation.** We model the 3D scene stylization process as the linear transformation between the constructed and stylized point clouds. Specifically, the constructed point cloud is modulated using the predicted linear transformation matrix $\mathbf{T}$, as described in (1). We use a series of point cloud aggregation modules to gather the point cloud information, and the convolution layers to process the reference image feature $\mathbf{F}^s$ to compute the matrix $\mathbf{T}$.

Huang and Belongie [20] use first-order statistics to encode the style information, and transform the image style via the AdaIN normalization layers. The WCT [33] approach uses whitening and coloring transformation to match the second-order statistics of the input image to those of the reference image. In addition, the LST [30] scheme leverages the convolutional neural networks to reduce the computational cost of solving the transformation matrix in the WCT method for real-time universal style transfer. Most recently, the TPFR method [53] proposes a regularization layer to facilitate the generalization of image stylization models.

Video stylization aims to transfer the style of a reference image to a sequence of video frames. To address the temporal flickering issue produced by the image stylization approaches, numerous approaches [4, 7, 10, 15, 18] incorporate optical flow modules to train feed-forward networks for transferring a particular style to the videos. Several recent frameworks [9, 11, 56] enable the video style transfer to *arbitrary* styles. Although significant advances have been made, existing methods are designed specifically for transferring the style of 2D images or video sequences. As shown in Figure 2, simply applying these schemes for the 3D scene stylization task leads to problematic results, such as blurry or short/long-range inconsistent images across different novel views.

Several efforts have been made to perform the stylization in 3D space. However, these approaches are only applicable to single objects [23], narrow-baseline stereo images [6, 13], or light field images [17]. In contrast, our method stylizes complex 3D scenes, and produces consistent results at arbitrary viewpoints.

Deep neural networks for point clouds. Various deep neural network (DNN)-based models [24, 28, 29, 31, 41, 42, 59, 61, 66] that take point clouds as input are widely studied for vision recognition tasks including 3D semantic segmentation [2], 3D shape classification or normal estimation [60], and 3D object part segmentation [62]. Recently, Mallya *et al*. [36] proposes a point cloud colorization approach for the video-to-video synthesis task. In this work, we propose a DNN-based point cloud transformation model for the 3D scene stylization task. We note that the PSNet [3] model aims to transfer the style of the point cloud. Nevertheless, there are two issues for the PSNet method to be applied to the 3D scene stylization task. First, it does not support synthesizing high-quality stylized images at novel views, which makes the PSNet framework limited for real-world (*e.g*. AR) applications. Second, since the PSNet scheme requires the optimization process for each specific scene, it is time-consuming, and fails to handle large-scale scenes in the real-world with more than 60M points, such as those in the Tanks and Temples dataset [25]. In contrast, we propose a feed-forward point cloud model that is efficient, capable of handling large-scale 3D scenes, and generating images with arbitrary styles at various novel views.

3. Methodology

We present the overview of the proposed 3D scene stylization framework in Figure 3. Given a set of N input images $\{\mathbf{I}_n\}_{n=1}^N$ of a static scene, and a reference image $\mathbf{S}$ with the desired style, our goal is to synthesize the image $\mathbf{O}_v$ at the novel view v with the camera pose $(\mathbf{R}_v, t_v)$ and intrinsic $\mathbf{K}_v$. Specifically, the generated novel view image $\mathbf{O}_v$ needs to 1) match the style of the reference image $\mathbf{S}$ and 2) be consistent for different viewpoints v. To handle such (especially the consistency) requirements, our core idea is to 1) construct a single 3D representation, *i.e.* point cloud, for the holistic scene, and 2) transform the representation to produce not only stylized but also consistent novel view synthesis results. The proposed approach consists of three steps: point cloud creation, point cloud transformation, and novel view synthesis, described in the following sections.

3.1. Point Cloud Construction

Pre-processing. Our method leverages camera pose and proxy geometry to construct the 3D point cloud. Given the input images $\{\mathbf{I}_n\}_{n=1}^N$, we first use a structure-from-motion algorithm [47] to estimate the camera poses $\{\mathbf{R}_n, t_n\}_{n=1}^N$ and intrinsic parameters $\{\mathbf{K_n}\}_{n=1}^N$. For each image $\mathbf{I}_n$, we use the COLMAP [47, 48] and Delaunay-based reconstruction [21, 26] schemes to obtain the depth map $\mathbf{D}_n$ that can appropriately back-project the points from the image plane to the 3D space.

Feature extraction and back-projection. Since our goal is to transform the point cloud representation for the 3D scene stylization purpose, we need the point cloud representation to encode the style information. Therefore, we use the VGG-19 model [51] pre-trained on the ImageNet [8] dataset to extract the relu3_1 feature maps $\{\mathbf{F}_n^c\}_{n=1}^N$ of the input images $\{\mathbf{I}_n\}_{n=1}^N$. The width and height of each feature map is H and W. According to the depth map $\{\mathbf{D}_n\}_{n=1}^N$, we back-project all the points in each feature map to build the 3D point cloud $\{f_p^c\}_{p=1}^P$, where $P = NHW$ is the total number of points in the constructed point cloud.

3.2. Point Cloud Transformation

We model the 3D scene stylization process as a linear transformation [30] between the constructed and stylized point clouds. Intuitively, the goal is to match the covariance statistics of the stylized point clouds and those of the reference image $\mathbf{S}$. To achieve this, we use the pre-trained VGG-19 network to extract the relu3_1 feature map from the reference image $\mathbf{S}$ as the style feature map $\mathbf{F}^s$. Given the constructed point cloud $\{f_p^c\}_{p=1}^P$, we use a predicted linear transformation matrix $\mathbf{T}$ to compute the modulated point cloud $\{f_p^d\}_{p=1}^P$, namely

$$f_p^d = \mathbf{T}(f_p^c - \bar{f}^c) + \bar{f}^s \quad \forall p \in [1, \cdots, P], \qquad (1)$$

where $\bar{f}^c$ is the mean of the features in the point cloud $\{f_p^c\}_{p=1}^P$, and $\bar{f}^s$ is the mean of the style feature map $\mathbf{F}^s$.

Linear transformation matrix T. The transformation matrix $\mathbf{T}$ is computed from the style feature map $\mathbf{F}^s$ and constructed point cloud $\{f_p^c\}_{p=1}^P$. As shown in Figure 4, we adopt the strategy similar to the LST [30] method that uses the convolution layers, covariance computation, and fully-connected layers to compute the matrix $\mathbf{T}^s$ from the style feature map $\mathbf{F}^s$. On the other hand, we develop a series of point cloud aggregation modules to process the point cloud $\{f_p^c\}_{p=1}^P$, and use the covariance computation followed by the fully-connected layers to calculate the matrix $\mathbf{T}^c$. Finally, we obtain the transformation matrix $\mathbf{T} = \mathbf{T}^s\mathbf{T}^c$.

Point cloud aggregation. It is challenging to gather the information contained in the constructed point cloud $\{f_p^c\}_{p=1}^P$ due to the sparsity and non-uniformity. We note that the constructed point cloud is non-uniform if the input images cover a particular region of the 3D scene. In this work, we leverage the set abstraction [43] concept to aggregate the point cloud. The input $\{f_p^c\}_{p=1}^P$ to a point cloud aggregation module is a set of P points with feature dimension c, and the output $\{f_p^{c'}\}_{p=1}^{P'}$ is a set of P' points with dimension c'. We first sample a subset of P' points $\{f_p^c\}_{p=1}^{P'}$ using the iterative farthest point sampling algorithm [14, 39]. Viewing the sampled points as the centroids in the 3D space, we use a radius parameter r to find the nearby points to form a point group. By using the MLP layers and the max pooling operator to map each point group to a vector, we obtain the aggregated point cloud $\{f_p^{c'}\}_{p=1}^{P'}$. The output $\{f_p^{c'}\}_{p=1}^{P'}$ is then used as the input for the next module. We use three point cloud aggregation modules sequentially in our pipeline.

3.3. Novel View Synthesis and Model Training

We aim to synthesize stylized image $\mathbf{O}_v$ at an arbitrary novel view v. Given the target camera pose $(\mathbf{R}_v, t_v)$ and intrinsic $\mathbf{K}_v$, we use Pytorch3D [44, 57] to render the transformed 2D feature map $\mathbf{F}_v^d$. We then use a decoder network to generate the stylized novel view image $\mathbf{O}_v$ from the 2D feature map $\mathbf{F}_v^d$.

Model training. We keep the pre-trained VGG-19 feature extractor fixed during the whole training phase. We first train the decoder network to perform the non-stylized novel view synthesis. Since the ground-truth (non-stylized) novel view image is available in the training sets, we use the $\ell 1$ reconstruction loss to optimize the decoder network. We then keep the decoder network fixed, and train the proposed point cloud transformation module with the following loss functions:

- **Content loss** $\mathcal{L}_c$ ensures the preservation of the content information by measuring the distance between the pre-trained VGG-19 features of the generated stylized image $\mathbf{O}_v$ and the ground-truth (non-stylized) image $\mathbf{I}_v$.
- **Style loss** $\mathcal{L}_s$ encourages the synthesized image $\mathbf{O}_v$

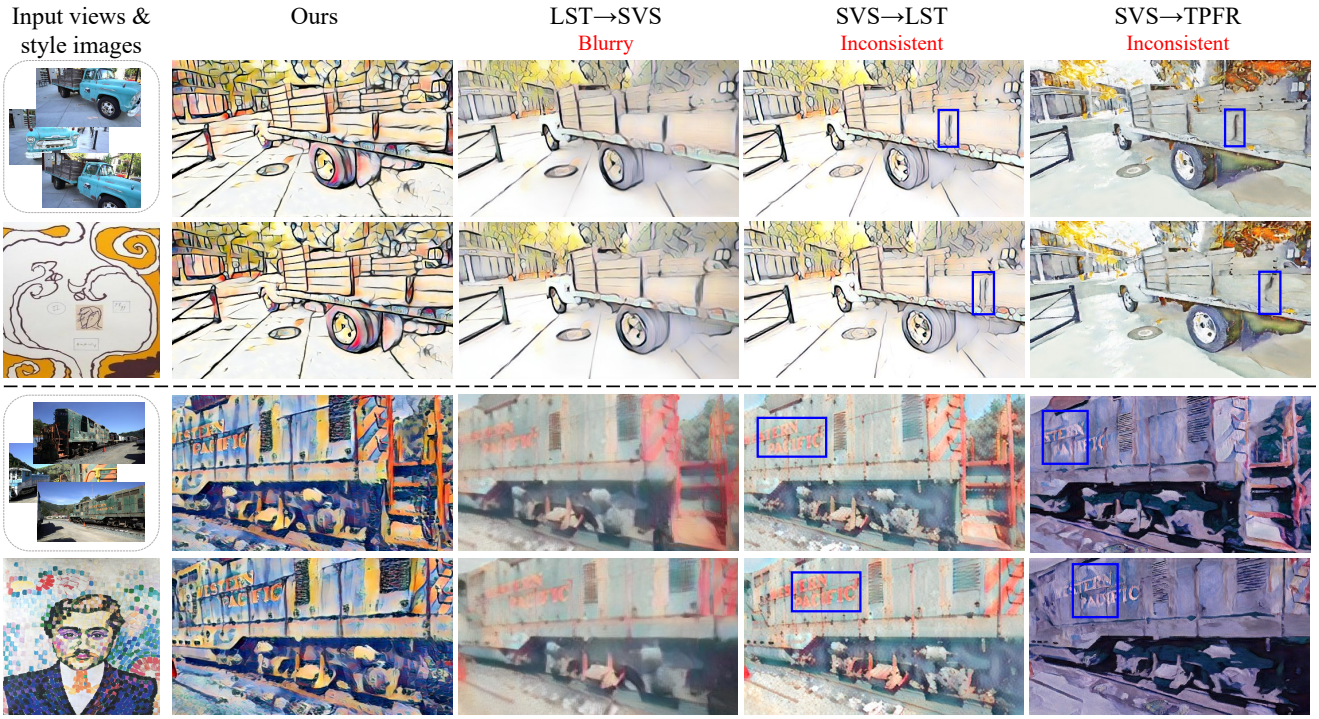

Figure 5. **Visual comparisons to image stylization-based approaches.** We compare the stylized novel view images generated by the three image stylization alternative schemes and our model on Tanks and Temples dataset [25].

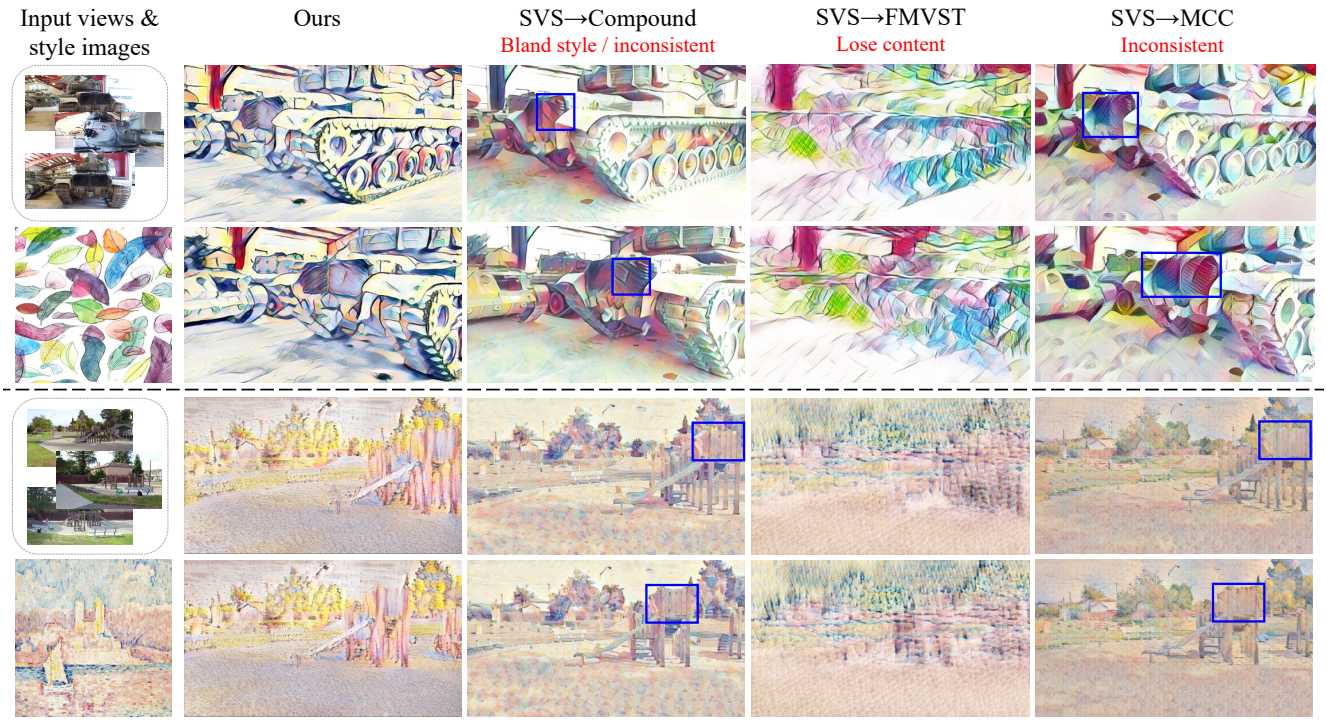

Figure 6. **Visual comparisons to video stylization-based approaches.** We compare the stylized novel view images generated by the three video stylization alternative schemes and our model on Tanks and Temples dataset [25].

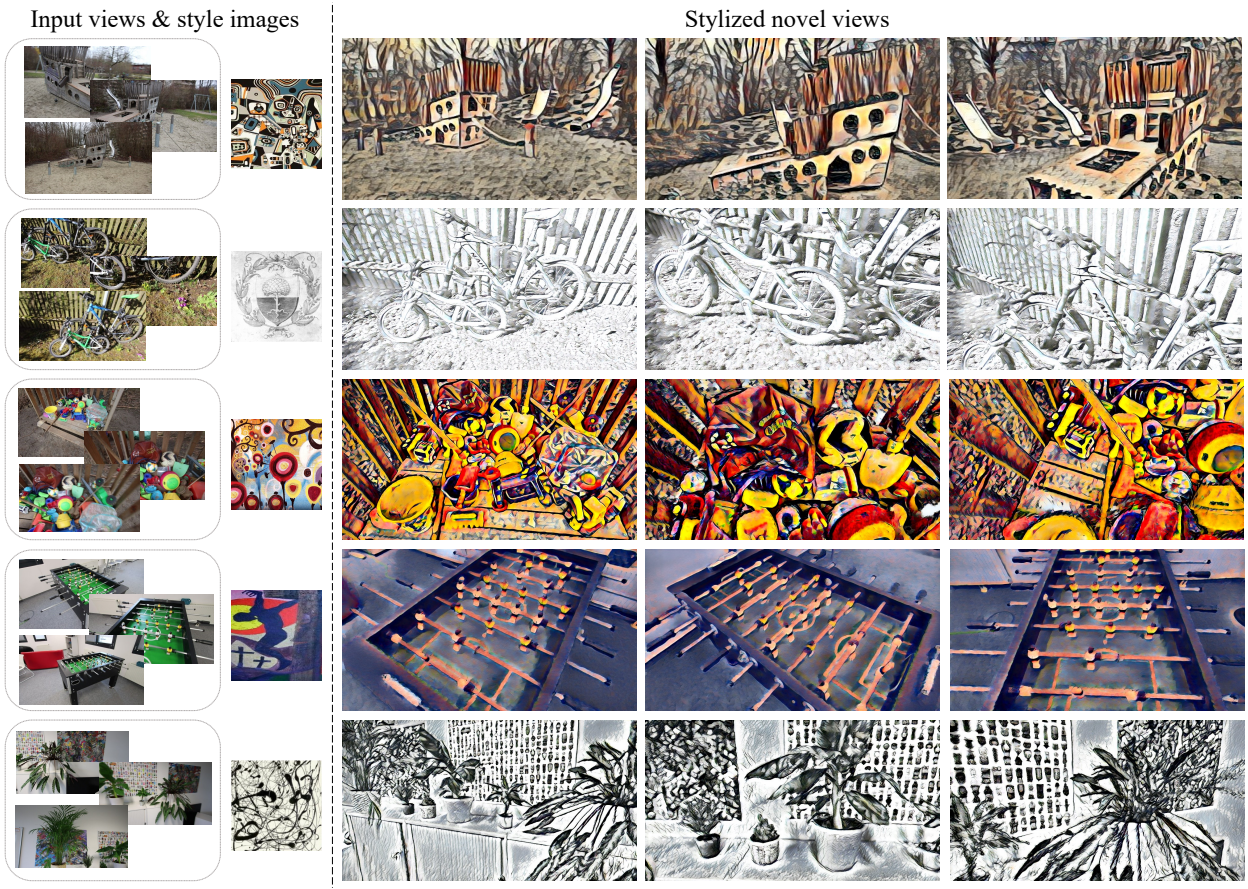

Figure 7. **Qualitative results on the FVS dataset.** We demonstrate the generalization of the proposed approach by training on the Tanks and Temples dataset, then testing on the FVS dataset.

to match the style of the reference image $\mathbf{S}$. Similar to recent style transfer approaches [22, 30], we extract the features at different layers of the pre-trained VGG-19 model, and compute the gram matrix differences.

The overall loss function for training the point cloud transformation module is

$$\mathcal{L} = \mathcal{L}_c(\mathbf{O}_v, \mathbf{I}_v) + \lambda \mathcal{L}_s(\mathbf{O}_v, \mathbf{S}), \tag{2}$$

where λ controls the importance of each loss term.

4. Experimental Results

We conduct extensive experiments on two real-world datasets to validate the efficacy of the proposed 3D scene stylization model.

Datasets. We use the Tanks and Temples [25] dataset for quantitative evaluation. Similar to the setting in FVS [45], we use 17 out of the 21 scenes for the training. The four remaining scenes (Truck, Train, M60 and Playground) are used for testing. We also present qualitative results on the FVS [45] dataset, which consists of 6 scenes: Bike, Flowers, Pirate, Digger, Sandbox and Soccertable. Note that both datasets are collected by *handheld* cameras in unconstraint motions.

Evaluated methods. As the 3D scene stylization task is a relatively new problem, we evaluate our method against alternative approaches built upon the state-of-the-art novel view synthesis NeRF++ [64], SVS [46], and image/video stylization schemes:

- **Image stylization → novel view synthesis:** We first use image stylization schemes LST [30] or TPFR [53] to transfer the style to the input images $\{\mathbf{I}_n\}_{n=1}^N$, then perform novel view synthesis.
- **Novel view synthesis → image stylization:** We apply image stylization to the novel view synthesis results.
- **Novel view synthesis → video stylization:** We use a series of novel view synthesis results to create a *video*, then apply video stylization methods Compound [56], FMVST [11], or MCC [9].

4.1. Qualitative Results

Image stylization. Figure 5 presents the qualitative comparison between the stylized novel view images generated by the three image stylization alternative schemes and the proposed method. Since the images are stylized inde-

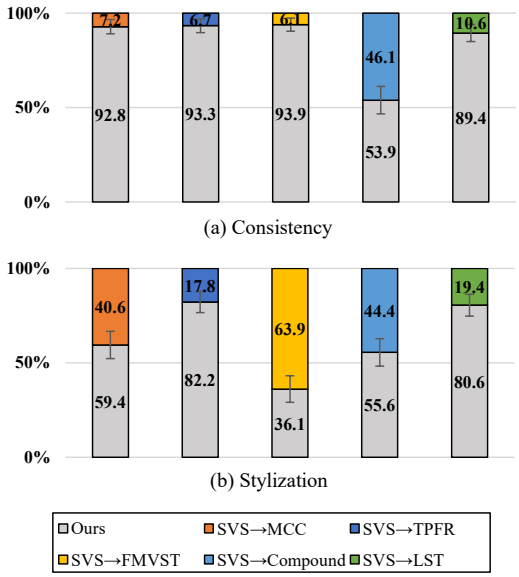

Figure 8. **User preference study.** We conduct a user study and ask subjects to select the results that (a) have more consistent contents across different video frames (*e.g.* less flickering), (b) better match the style of the example image. The number indicates the percentage of preference.

pendently without considering the consistency issue across different viewpoints, we observe two issues in the image stylization-based methods. First, LST → SVS generally produces blurry novel view images. Since the stylized input images are not consistent, the novel view synthesis approach tends to *blend* such inconsistency, which leads to blurry results. Second, the novel view synthesis results are not consistent if we operate in the reverse order, *i.e.* SVS → LST. We highlight the inconsistency using yellow boxes in Figure 5. Note that we observe the same problem if we replace SVS with NeRF++.

Video stylization. We qualitatively evaluate the results by the proposed method and three video stylization alternative approaches in Figure 6. Specifically, we create the videos using a series of novel view synthesis results. All the alternative approaches generate inconsistent results between two relatively far-away viewpoints since the video stylization methods only guarantee *short-term* consistency in the video. Although SVS→Compound generates less inconsistent results, the style of the novel view images is bland and not aligned with that of the reference image. On the other hand, SVS→FMVST creates images that better match the desired style, but fails to preserve the content of the original scene.

In contrast to the image and video stylization alternative approaches, our method 1) generates sharp novel view images with correct scene contents and the desired style, and 2) guarantees the short/long-range consistency. Furthermore, we demonstrate the generalization of the proposed framework in Figure 7 , where we use the model trained on the Tanks and Temples dataset to perform the 3D scene stylization task on the FVS dataset.

Table 1. **Short-range consistency.** We compare the long-range consistency using the warping error (↓) between the viewpoints of $(t-1)$-th and t-th testing video frames in the Tanks and Temples dataset [25]. We report the average errors of 15 diverse styles. The best performance is in **bold** and the second best is underscored.

Method	Truck	Playground	Train	M60	Average
NeRF++→LST	0.215	0.168	0.250	0.274	0.231
SVS→LST	0.192	0.159	0.220	0.241	0.206
NeRF++→TPFR	0.216	0.214	0.299	0.279	0.258
SVS→TPFR	0.235	0.237	0.291	0.276	0.264
NeRF++→Compound	0.188	0.169	0.229	0.208	0.202
SVS→Compound	**0.166**	**0.156**	0.199	**0.160**	0.172
NeRF++→FMVST	0.342	0.300	0.405	0.348	0.354
SVS→FMVST	0.343	0.304	0.412	0.337	0.354
NeRF++→MCC	0.250	0.201	0.269	0.255	0.246
SVS→MCC	0.242	0.198	0.260	0.224	0.232
Ours	0.184	0.158	**0.170**	0.172	**0.170**

Table 2. **Long-range consistency.** We compare the long-range consistency using the warping error (↓) between the viewpoints of $(t-7)$-th and t-th testing video frames in the Tanks and Temples dataset [25]. We report the average errors of 15 diverse styles. The best performance is in **bold** and the second best is underscored.

Method	Truck	Playground	Train	M60	Average
NeRF++→LST	0.570	0.349	0.520	0.639	0.521
SVS→LST	0.567	**0.327**	0.470	0.603	0.489
NeRF++→TPFR	0.579	0.436	0.503	0.655	0.541
SVS→TPFR	0.605	0.430	0.470	0.581	0.513
NeRF++→Compound	0.586	0.398	0.477	0.557	0.498
SVS→Compound	0.573	0.388	0.422	0.460	0.449
NeRF++→FMVST	0.742	0.525	0.636	0.695	0.644
SVS→FMVST	0.732	0.519	0.620	0.662	0.626
NeRF++→MCC	0.691	0.450	0.535	0.646	0.571
SVS→MCC	0.693	0.447	0.516	0.584	0.548
Ours	**0.559**	0.337	**0.412**	**0.458**	**0.431**

4.2. Quantitative Results

Stylization quality. We conduct a user study to understand the user preference between the proposed and the alternative approaches. For each testing scene in the Tanks and Temples dataset, we create a video using a series of stylized novel view synthesis results. By presenting two videos generated by different methods for the same scene, we ask the participants to select the one that (1) has more consistent contents across different video frames (e.g., less flickering), and (2) better matches the style of the reference image. As the results shown in Figure 8, the synthesized images by the proposed method are consistent and close to the reference style. We observe that the users slightly prefer the style generated by SVS→FMVST. However, as illustrated in Section 4.1 and Figure 6, SVS→FMVST fails to preserve the content of the original scene.

Short-range consistency. We use the warped LPIPS metric [65] to measure the consistency of the results across different viewpoints. Given a stylized image at a novel view

Figure 9. **Ablation study on the number of point cloud aggregation modules.** We compare the visual results of using 0/1/**3**/5 modules. We empirically decide to use 3 modules for better visual quality.

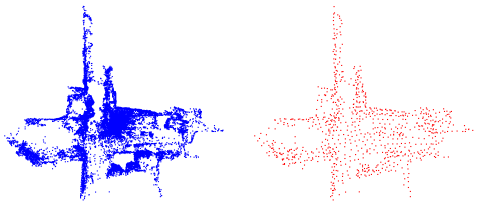

Figure 10. **Role of point aggregation.** We visualize the point distribution before and after the point aggregation. Our point aggregation module obtains a more uniform-distributed point set to fairly estimate the transformation matrix $\mathbf{T}$ that achieves better 3D scene stylization results.

v, we warp the results generated at another novel view v' to the view v according to the 3D proxy geometry described in Section 3.1. We then compute the score by

$$E_{\text{warp}}(\mathbf{O}_v, \mathbf{O}'_v) = \text{LPIPS}(\mathbf{O}_v, W(\mathbf{O}'_v), \mathbf{M}_{v'v}), \quad (3)$$

where W is the warping function and $\mathbf{M}_{v'v}$ is the mask of valid pixels warped from the views v' to v. Note that we only use the values of valid pixels in the mask for the "spatial average" operation in [65]. For each of the testing scenes in the Tanks and Temples dataset, we use 15 style images [11] to compute the average warping error.

We first present the short-range consistency comparison in Table 1. In this experiment, we use the nearby view for a specific novel view to compute the warping error.[1] In general, the image stylization alternative methods produce short-range inconsistent results as they process each novel view independently. In contrast, the proposed method performs comparably against the video stylization-based approach SVS→Compound that considers the short-term consistency in videos. Nevertheless, SVS→Compound synthesizes bland styles that do not match the desired styles, as the results demonstrated in Figure 6 and Figure 8.

Long-range consistency. We also consider the long-range consistency issue in our experiments. In this experiment, we compute the warping error between the results of two (relatively) far-away views.[2] As demonstrated in Table 2, the proposed method performs favorably against the alternative approaches. Despite the capability of ensuring short-range consistency, video stylization-based schemes fail to maintain the long-range consistency.

Number of point cloud aggregation modules. We conduct an ablation study to decide the number of point cloud aggregation modules described in Section 3.2. The results are presented in Figure 9. We empirically choose to use three modules for better visual quality. Moreover, we visualize the point distributions before and after the aggregation in Figure 10 to understand the role of the aggregation module. The point density before the aggregation is higher around the regions of the 3D scene where more input images cover. As a result, the prediction of the transformation matrix $\mathbf{T}$ is dominated by such regions, which leads to low-quality stylization results (2nd column in Figure 9). By using the point cloud aggregation modules, we obtain a more uniform-distributed point set that fairly estimates the matrix $\mathbf{T}$ for the 3D scene stylization task.

5. Conclusions

In this work, we introduce a 3D scene stylization problem that aims to modify the style of the 3D scene and synthesize images at arbitrary novel views. We construct a single 3D representation, *i.e.* point cloud, for the holistic scene, and design a point cloud transformation module to transfer the style of the reference image to the 3D representation. Qualitative and quantitative evaluations validate that our method synthesizes images that 1) contain the desired style and 2) are consistent across various novel views.

Acknowledgements

This work is supported in part by the NSF CAREER Grant #1149783 and a gift from Verisk.

[1] We use the viewpoints of $(t-1)$-th and t-th testing video frames as the views v' and v, respectively.

[2] We use the viewpoints of $(t-7)$-th and t-th testing video frames as the views v' and v, respectively.

References

[1] Kara-Ali Aliev, Artem Sevastopolsky, Maria Kolos, Dmitry Ulyanov, and Victor Lempitsky. Neural point-based graphics. In *ECCV*, 2020. 2

[2] Iro Armeni, Sasha Sax, Amir R Zamir, and Silvio Savarese. Joint 2d-3d-semantic data for indoor scene understanding. *arXiv preprint arXiv:1702.01105*, 2017. 3

[3] Xu Cao, Weimin Wang, Katashi Nagao, and Ryosuke Nakamura. Psnet: A style transfer network for point cloud stylization on geometry and color. In *WACV*, 2020. 3

[4] Dongdong Chen, Jing Liao, Lu Yuan, Nenghai Yu, and Gang Hua. Coherent online video style transfer. In *ICCV*, 2017. 3

[5] Dongdong Chen, Lu Yuan, Jing Liao, Nenghai Yu, and Gang Hua. Stylebank: An explicit representation for neural image style transfer. In *CVPR*, 2017. 2

[6] Dongdong Chen, Lu Yuan, Jing Liao, Nenghai Yu, and Gang Hua. Stereoscopic neural style transfer. In *CVPR*, 2018. 3

[7] Xinghao Chen, Yiman Zhang, Yunhe Wang, Han Shu, Chunjing Xu, and Chang Xu. Optical flow distillation: Towards efficient and stable video style transfer. In *ECCV*, 2020. 3

[8] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009. 4

[9] Yingying Deng, Fan Tang, Weiming Dong, haibin Huang, Ma chongyang, and Changsheng Xu. Arbitrary video style transfer via multi-channel correlation. In *AAAI*, 2021. 2, 3, 6

[10] Chang Gao, Derun Gu, Fangjun Zhang, and Y. Yu. Reconet: Real-time coherent video style transfer network. In *ACCV*, 2018. 3

[11] Wei Gao, Yijun Li, Yihang Yin, and Ming-Hsuan Yang. Fast video multi-style transfer. In *WACV*, 2020. 2, 3, 6, 8

[12] Leon A. Gatys, Alexander S. Ecker, and Matthias Bethge. Image style transfer using convolutional neural networks. In *CVPR*, 2016. 2

[13] Xinyu Gong, Haozhi Huang, Lin Ma, Fumin Shen, Wei Liu, and Tong Zhang. Neural stereoscopic image style transfer. In *ECCV*, 2018. 3

[14] Teofilo F Gonzalez. Clustering to minimize the maximum intercluster distance. *Theoretical computer science*, 38:293–306, 1985. 4

[15] Agrim Gupta, Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Characterizing and improving stability in neural style transfer. In *ICCV*, 2017. 3

[16] Tewodros Habtegebrial, Varun Jampani, Orazio Gallo, and Didier Stricker. Generative view synthesis: From single-view semantics to novel-view images. In *NeurIPS*, 2020. 1

[17] David Hart, Bryan Morse, and Jessica Greenland. Style transfer for light field photography. In *WACV*, 2020. 3

[18] Haozhi Huang, Hao Wang, Wenhan Luo, Lin Ma, Wenhao Jiang, Xiaolong Zhu, Zhifeng Li, and Wei Liu. Real-time neural style transfer for videos. In *CVPR*, 2017. 3

[19] Hsin-Ping Huang, Hung-Yu Tseng, Hsin-Ying Lee, and Jia-Bin Huang. Semantic view synthesis. In *ECCV*, 2020. 1

[20] Xun Huang and Serge Belongie. Arbitrary style transfer in real-time with adaptive instance normalization. In *ICCV*, 2017. 3

[21] Michal Jancosek and Tomas Pajdla. Multi-view reconstruction preserving weakly-supported surfaces. In *CVPR*, 2011. 4

[22] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *ECCV*, 2016. 2, 6

[23] Hiroharu Kato, Yoshitaka Ushiku, and Tatsuya Harada. Neural 3d mesh renderer. In *CVPR*, 2018. 3

[24] Roman Klokov and Victor Lempitsky. Escape from cells: Deep kd-networks for the recognition of 3d point cloud models. In *ICCV*, 2017. 3

[25] Arno Knapitsch, Jaesik Park, Qian-Yi Zhou, and Vladlen Koltun. Tanks and temples: Benchmarking large-scale scene reconstruction. *ACM TOG (Proc. SIGGRAPH)*, 36(4), 2017. 2, 3, 5, 6, 7

[26] Johannes Kopf, Michael Cohen, and Richard Szeliski. First-person hyper-lapse videos. *ACM TOG (Proc. SIGGRAPH)*, 33:1–10, 07 2014. 4

[27] Johannes Kopf, Kevin Matzen, Suhib Alsisan, Ocean Quigley, Francis Ge, Yangming Chong, Josh Patterson, Jan-Michael Frahm, Shu Wu, Matthew Yu, et al. One shot 3d photography. *ACM TOG (Proc. SIGGRAPH)*, 39(4):76–1, 2020. 2

[28] Truc Le and Ye Duan. Pointgrid: A deep network for 3d shape understanding. In *CVPR*, 2018. 3

[29] Jiaxin Li, Ben M Chen, and Gim Hee Lee. So-net: Self-organizing network for point cloud analysis. In *CVPR*, 2018. 3

[30] Xueting Li, Sifei Liu, Jan Kautz, and Ming-Hsuan Yang. Learning linear transformations for fast arbitrary style transfer. In *CVPR*, 2019. 1, 2, 3, 4, 6

[31] Yangyan Li, Rui Bu, Mingchao Sun, Wei Wu, Xinhan Di, and Baoquan Chen. Pointcnn: Convolution on x-transformed points. In *NIPS*, 2018. 3

[32] Yijun Li, Chen Fang, Jimei Yang, Zhaowen Wang, Xin Lu, and Ming-Hsuan Yang. Diversified texture synthesis with feed-forward networks. In *CVPR*, 2017. 2

[33] Yijun Li, Chen Fang, Jimei Yang, Zhaowen Wang, Xin Lu, and Ming-Hsuan Yang. Universal style transfer via feature transforms. In *NIPS*, 2017. 3

[34] Andrew Liu, Richard Tucker, Varun Jampani, Ameesh Makadia, Noah Snavely, and Angjoo Kanazawa. Infinite nature: Perpetual view generation of natural scenes from a single image. *arXiv preprint arXiv:2012.09855*, 2020. 1

[35] Lingjie Liu, Jiatao Gu, Kyaw Zaw Lin, Tat-Seng Chua, and Christian Theobalt. Neural sparse voxel fields. In *NeurIPS*, 2020. 2

[36] Arun Mallya, Ting-Chun Wang, Karan Sapra, and Ming-Yu Liu. World-consistent video-to-video synthesis. In *ECCV*, 2020. 3

[37] Moustafa Meshry, Dan B. Goldman, Sameh Khamis, Hugues Hoppe, Rohit Pandey, Noah Snavely, and Ricardo Martin-Brualla. Neural rerendering in the wild. In *CVPR*, 2019. 2

[38] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. In *ECCV*, 2020. 1, 2

[39] Carsten Moenning and Neil A Dodgson. Fast marching farthest point sampling. Technical report, University of Cambridge, Computer Laboratory, 2003. 4

[40] Simon Niklaus, Long Mai, Jimei Yang, and Feng Liu. 3d ken burns effect from a single image. *ACM TOG (Proc. SIGGRAPH)*, 38(6):1–15, 2019. 2

[41] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *CVPR*, 2017. 3

[42] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *NIPS*, 2017. 3

[43] Charles R Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *NIPS*, 2017. 4

[44] Nikhila Ravi, Jeremy Reizenstein, David Novotny, Taylor Gordon, Wan-Yen Lo, Justin Johnson, and Georgia Gkioxari. Accelerating 3d deep learning with pytorch3d. *arXiv preprint arXiv:2007.08501*, 2020. 4

[45] Gernot Riegler and Vladlen Koltun. Free view synthesis. In *ECCV*, 2020. 2, 6

[46] Gernot Riegler and Vladlen Koltun. Stable view synthesis. In *CVPR*, 2021. 1, 2, 6

[47] Johannes L. Schonberger and Jan-Michael Frahm. Structure-from-motion revisited. In *CVPR*, 2016. 4

[48] Johannes Schönberger, Enliang Zheng, Marc Pollefeys, and Jan-Michael Frahm. Pixelwise view selection for unstructured multi-view stereo. In *ECCV*, 2016. 4

[49] Falong Shen, Shuicheng Yan, and Gang Zeng. Neural style transfer via meta networks. In *CVPR*, 2018. 2

[50] Meng-Li Shih, Shih-Yang Su, Johannes Kopf, and Jia-Bin Huang. 3d photography using context-aware layered depth inpainting. In *CVPR*, 2020. 2

[51] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015. 4

[52] Pratul P. Srinivasan, Richard Tucker, Jonathan T. Barron, Ravi Ramamoorthi, Ren Ng, and Noah Snavely. Pushing the boundaries of view extrapolation with multiplane images. In *CVPR*, 2019. 2

[53] Jan Svoboda, Asha Anoosheh, Christian Osendorfer, and Jonathan Masci. Two-stage peer-regularized feature recombination for arbitrary image style transfer. In *CVPR*, 2020. 1, 3, 6

[54] Richard Tucker and Noah Snavely. Single-view view synthesis with multiplane images. In *CVPR*, 2020. 2

[55] Dmitry Ulyanov, Vadim Lebedev, Andrea Vedaldi, and Victor Lempitsky. Texture networks: Feed-forward synthesis of textures and stylized images. In *ICML*, 2016. 2

[56] Wenjing Wang, Jizheng Xu, Li Zhang, Yue Wang, and Jiaying Liu. Consistent video style transfer via compound regularization. In *AAAI*, 2020. 2, 3, 6

[57] Olivia Wiles, Georgia Gkioxari, Richard Szeliski, and Justin Johnson. SynSin: End-to-end view synthesis from a single image. In *CVPR*, 2020. 2, 4

[58] Suttisak Wizadwongsa, Pakkapon Phongthawee, Jiraphon Yenphraphai, and Supasorn Suwajanakorn. Nex: Real-time view synthesis with neural basis expansion. In *CVPR*, 2021. 2

[59] Wenxuan Wu, Zhongang Qi, and Li Fuxin. Pointconv: Deep convolutional networks on 3d point clouds. In *CVPR*, 2019. 3

[60] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3d shapenets: A deep representation for volumetric shapes. In *CVPR*, 2015. 3

[61] Saining Xie, Sainan Liu, Zeyu Chen, and Zhuowen Tu. Attentional shapecontextnet for point cloud recognition. In *CVPR*, 2018. 3

[62] Li Yi, Vladimir G Kim, Duygu Ceylan, I-Chao Shen, Mengyan Yan, Hao Su, Cewu Lu, Qixing Huang, Alla Sheffer, and Leonidas Guibas. A scalable active framework for region annotation in 3d shape collections. *ACM TOG (Proc. SIGGRAPH)*, 35(6):1–12, 2016. 3

[63] Alex Yu, Vickie Ye, Matthew Tancik, and Angjoo Kanazawa. pixelNeRF: Neural radiance fields from one or few images. In *CVPR*, 2021. 2

[64] Kai Zhang, Gernot Riegler, Noah Snavely, and Vladlen Koltun. Nerf++: Analyzing and improving neural radiance fields. *arXiv preprint arXiv:2010.07492*, 2020. 1, 2, 6

[65] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *CVPR*, 2018. 2, 7, 8

[66] Hengshuang Zhao, Li Jiang, Chi-Wing Fu, and Jiaya Jia. PointWeb: Enhancing local neighborhood features for point cloud processing. In *CVPR*, 2019. 3

[67] Tinghui Zhou, Richard Tucker, John Flynn, Graham Fyffe, and Noah Snavely. Stereo magnification: learning view synthesis using multiplane images. *ACM TOG (Proc. SIGGRAPH)*, 37(4):1–12, 2018. 2