3D Photo Stylization: Learning to Generate Stylized Novel Views from a Single Image

Fangzhou Mu[1*] Jian Wang[2†] Yicheng Wu[2†] Yin Li[1†]
[1]University of Wisconsin-Madison [2]Snap Research
[1]{fmu2, yin.li}@wisc.edu [2]{jwang4, yicheng.wu}@snap.com

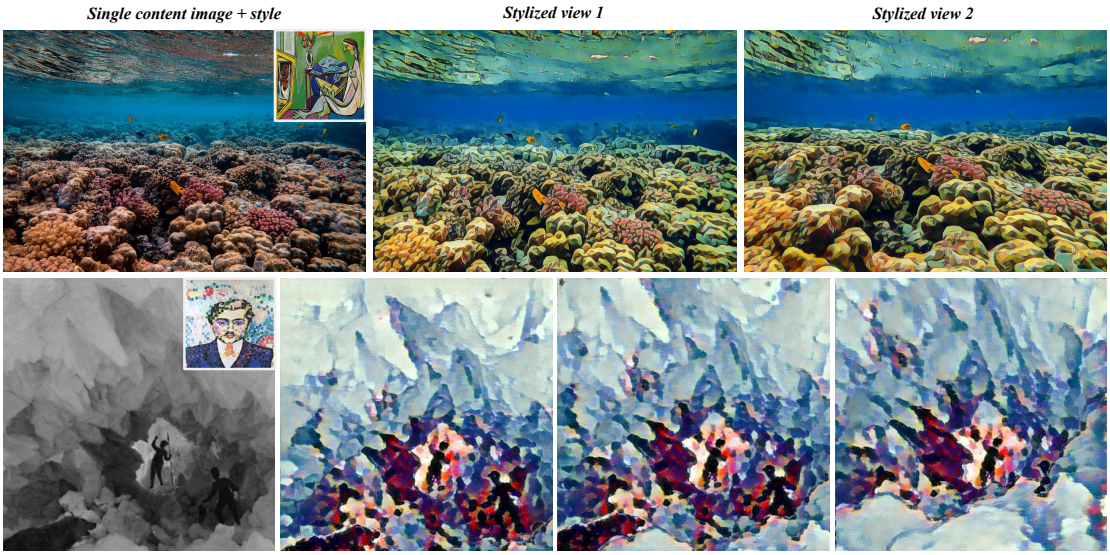


Figure 1. **3D photo stylization.** Given a *single* content image, our method synthesizes novel views of the scene in an arbitrary style. In doing so, our method delivers immersive viewing experience of a memorable moment within existing photos.

Abstract

Visual content creation has spurred a soaring interest given its applications in mobile photography and AR / VR. Style transfer and single-image 3D photography as two representative tasks have so far evolved independently. In this paper, we make a connection between the two, and address the challenging task of 3D photo stylization — generating stylized novel views from a single image given an arbitrary style. Our key intuition is that style transfer and view synthesis have to be jointly modeled for this task. To this end, we propose a deep model that learns geometry-aware content features for stylization from a point cloud representation of the scene, resulting in high-quality stylized images that are consistent across views. Further, we introduce a novel training protocol to enable the learning using only 2D images. We demonstrate the superiority of our method via extensive qualitative and quantitative studies, and showcase key applications of our method in light of the growing demand for 3D content creation from 2D image assets.[1]

1. Introduction

Given an input content image and a reference style image, neural style transfer [4, 13, 14, 16, 22, 25, 33, 36, 41, 49] creates a novel image that "paints" the content with the style. Despite a high quality stylized image, the result is limited to the same viewpoint of the content image. What if we can render stylized images from different views? See Fig. 1 for examples. When displayed with parallax, this capacity will provide drastically more immersive visual experience for 2D images, and support the application of interactive browsing of 3D photos on mobile and AR/VR devices. In this paper, we address the new task of generating stylized images of novel views *from a single input image and an arbitrary reference style image*, as illustrated in Fig 1. We refer to this task as 3D photo stylization — a marriage between style transfer and novel view synthesis.

3D photo stylization has several major technical barriers. As observed in [21], directly combining existing methods of style transfer and novel view synthesis yields blurry or inconsistent stylized images, even with dense 3D geometry obtained from structure from motion and multi-view stereo. This challenge is further manifested with a single content

*Work partially done when Fangzhou was an intern at Snap Research
†co-corresponding authors
[1]Project page: http://pages.cs.wisc.edu/~fmu/style3d

image as the input, where a method must resort to monocular depth estimation with incomplete and noisy 3D geometry, leading to holes and artifacts when synthesizing stylized images of novel views. In addition, training deep models for this task requires a large-scale dataset of diverse scenes with dense geometry annotation that is currently lacking.

To bridge this gap, we draw inspiration from one-shot 3D photography [29, 40, 50], and adopt a point cloud based scene representation [21, 40, 57]. Our key innovation is a deep model that learns 3D geometry-aware features on the point cloud *without using 2D image features from the content image* for rendering novel views with a consistent style. Our method accounts for the input noise from depth maps, and jointly models style transfer and view synthesis. Moreover, we propose a novel training scheme that enables learning our model using standard image datasets (*e.g.*, MS-COCO [34]), without the need of multi-view images or ground-truth depth maps.

Our contributions are summarized into three folds. **(1)** We present the first method to address the new task of 3D photo stylization — synthesizing stylized novel views from a single content image with arbitrary styles. **(2)** Unlike previous methods, our method learns geometry-aware features on a point cloud without using 2D content image features and from only 2D image datasets. **(3)** Our method demonstrates superior qualitative and quantitative results, and supports several interesting applications.

2. Related work

Neural Style Transfer. Neural style transfer has received considerable attention. Image style transfer [12, 13] renders the content of one image in the style of another. Video style transfer [48] injects a style to a sequence of video frames to produce temporally consistent stylization, often by enforcing smoothness constraint on optical flow [3, 20, 48, 55] or in the feature space [9, 36]. Our method faces the same challenge as video style transfer; that the style must be consistent across views. However, our task of 3D photo stylization is more challenging, as it requires the synthesis of novel views and a consistent style among all views.

Technically, early methods formulate style transfer as a slow iterative optimization process [12, 13]. Fast feed-forward models later perform stylization in a single forward pass, but can only accommodate one [25, 54] or a few styles [4, 10]. Most relevant to our work are methods that allow for the transfer of *arbitrary* styles while retaining the efficiency of a feed-forward model [6, 22, 33]. Our style transfer module builds on Liu *et al*. [36], extending an attention-based method to support arbitrary 3D stylization.

Novel View Synthesis from a Single Image. Novel view synthesis from a single image, also known as one-shot 3D photography, has seen recent progress thanks to deep learning. Existing approach can be broadly classified as end-to-end models [7, 18, 31, 46, 52, 53, 57, 60] and modular systems [24, 29, 40, 50]. End-to-end methods often fail to recover accurate scene geometry and have difficulty generalizing beyond the scene categories present in training. Hence, our method builds on modular systems.

Modular systems for one-shot 3D photography combine depth estimation [44,45,59] and inpainting models [35], and have demonstrated strong results for in-the-wild images. Niklaus *et al*. [40] maintains and rasterizes a point cloud representation of the scene to synthesize 3D Ken Burns effect. Later methods [29, 50] improve on synthesis quality via local content and depth inpainting on a layered depth image (LDI) of the scene. Jampani *et al*. [24] further introduces soft scene layering to better preserve appearance details. Our work is closely related to Shih *et al*. [50]. We extend their LDI inpainting method for point cloud, and leverage their system to generate "pseudo" views during training. Our method also uses the differentiable rasterizer from [40].

3D Stylization. There has been a growing interest in the stylization of 3D content for creative shape editing [2, 58], visual effect simulation [17], stereoscopic image editing [5, 15] and novel view synthesis [8, 21]. Our method falls in this category and is most relevant to stylized novel view synthesis [8, 21]. The key difference is that our method generates stylized novel views from a single image, while previous methods need hundreds of calibrated views as input. Another difference is that our model learns 3D geometry aware features on a point cloud. In contrast, Huang *et al*. [21] back-projects 2D image features to 3D space without accounting for scene geometry. While their point aggregation module enables *post hoc* processing of image-derived features, the point features remain 2D, leading to visual artifacts and inadequate stylization in renderings. Our work is also related to point cloud stylization *e.g.*, PSNet [2] and 3DStyleNet [58]. Both our method and [2,58] use point cloud as the representation. The difference is that point cloud is an enabling device for stylization and view synthesis in our method, and not as the end product as in [2, 58].

Deep Models for Point Cloud Processing. Many deep models have been developed for point cloud processing. Among the popular architectures are models of set based [42, 43], graph convolution based [30, 56] and point convolution based [19, 51]. Our model extends a graph based model [56] to handle dense point clouds (one million points) for high quality stylization.

3. 3D Photo Stylization

Given *a single input content image* and *an arbitrary style image*, the goal of 3D photo stylization is to generate stylized novel views of the content image. The key of our method is the learning of 3D geometry aware content features directly from a point cloud representation of the scene for high-quality stylization that is consistent across views.

点云重建，风格迁移，多视角渲染，是否支持纹理贴图？？

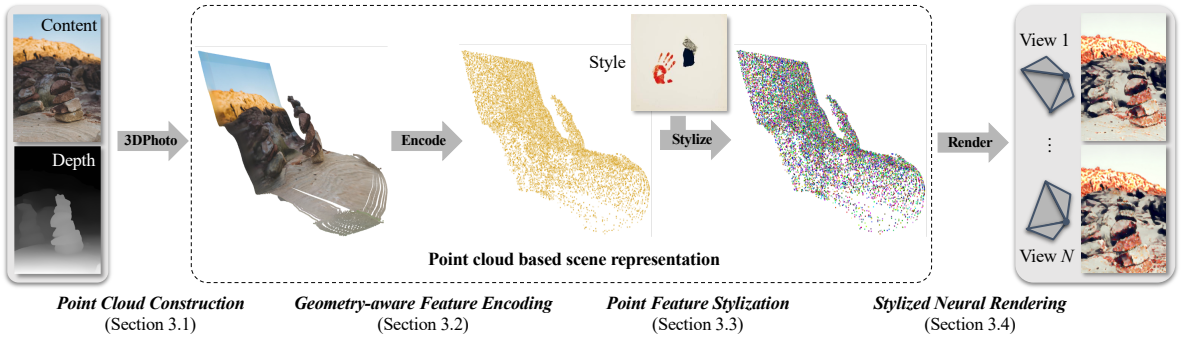


Figure 2. **Method overview.** Central to our method is a point cloud based scene representation that enables geometry-aware feature learning, attention-based feature stylization and consistent stylized renderings across views. Specifically, we first construct an RGB point cloud from the content image and its estimated depth map. Content features are then extracted directly from the point cloud and stylized given an image of the reference style. Finally, the stylized point features are rendered to novel views and decoded into stylized images.

In this section, we describe our workflow at *inference* time.

Method Overview. Fig. 2 presents an overview of our method. Our method starts by back-projecting the input content image into an RGB point cloud using its estimated depth map. The point cloud is further "inpainted" to cover dissoccluded parts of the scene and then "normalized" (Section 3.1). An efficient graph convolutional network is designed to process the point cloud and extract 3D geometry aware features on the point cloud, leading to point-wise features tailored for 3D stylization (Section 3.2). A style transfer module is subsequently adapted to modulate those point-wise features using the input style image (Section 3.3). Finally, a differentiable rasterizer projects the featurized points to novel views for the synthesis of stylized images that are consistent across views (Section 3.4).

图卷积处理点云

3.1. Point Cloud Construction

Our method starts by lifting the content image into an RGB point cloud, and further normalizes the point cloud to account for scale ambiguity and uneven point density.

Depth Estimation and Synthesis of Hidden Geometry. Our method first estimates a dense depth map using an off-the-shelf deep model for monocular depth estimation (LeReS [59]). A key challenge for single-image novel view synthesis is the occlusion in the scene. A dense depth map might expose many "holes" when projected to a different view. Inpainting the occluded geometry is thus critical for view synthesis. To this end, we further employs the method of Shih *et al*. [50] for the synthesis of occluded geometry on a layered depth image (LDI). Thanks to the duality between point cloud and LDI, we map the LDI pixels to an RGB point cloud via perspective back-projection.

Point Cloud Normalization. In light of scale ambiguity and uneven point density characteristic of image-derived point clouds, we transform them into Normalized Device Coordinate (NDC) [38] before further processing. The resulting points fall within the $[-1, 1]$ cube with density adjusted accordingly to account for perspectivity. As shown

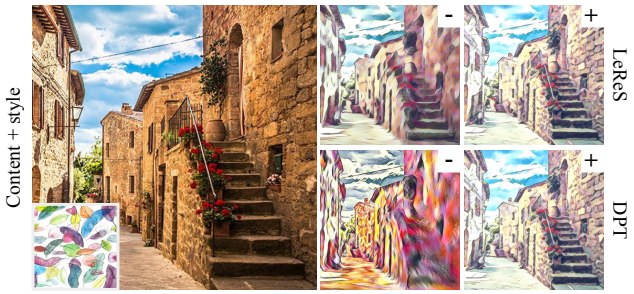


Figure 3. **Effect of point cloud normalization.** Model without normalization (-) performs poorly due to scale ambiguity in depth estimation and non-uniformity in point distribution. In contrast, model with normalization (+) captures fine appearance detail and produces strong stylization irrespective of depth estimator in use.

有归一化操作可以获得精细外观细节，风格化效果更好

in Fig 3, this simple procedure is crucial for our method to generalize across scene categories, and allows us to switch to different depth estimators without re-training our model.

3.2. Encoding Features on Point Cloud

Our next step is to learn features amenable to stylization. While virtually all existing style transfer algorithms make use of ImageNet pre-trained VGG features, we found that associating 3D points with back-projected VGG features (such as in Huang *et al*. [21]) is sub-optimal for stylized novel view synthesis, leading to geometric distortion and structural artifacts as shown in our ablation. We argue that features from a network pre-trained on 2D images are incompetent to describe the intricacy of 3D geometry. This leads us to design an efficient graph convolutional network (GCN) that learns geometry aware features directly from an RGB point cloud, as opposed to using 2D image features.

Efficient GCN. One common drawback for GCN architectures lies in their scalability. Existing GCNs are designed for points clouds with a few thousand points [30], whereas an image at 1K resolution results in *one million* points after inpainting. To bridge this gap, we propose a highly efficient GCN encoder by drawing strength from multiple point-based network architectures.

直接对反投影的VGG特征处理会导致几何失真和结构异常

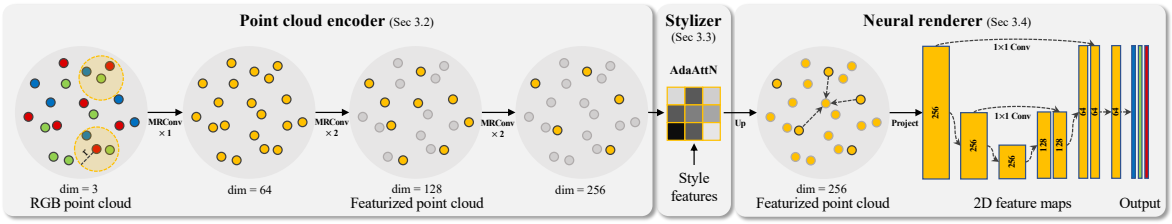


Figure 4. **Components of our deep model.** Our model includes three modules — a point cloud encoder, a stylizer and a neural renderer. The encoder applies MRConvs [30] along with farthest point sampling to embed and sub-sample the input RGB point cloud. The stylizer computes attention between the embedded content and style features, and uses attention-weighted affine transformation to modulate the content features for stylization. The neural render consists of a rasterizer that anti-aliases the modulated point features and projects them to novel views, and a U-Net [47] that refines the resulting 2D feature maps and decodes them into stylized images.

Our GCN encoder adopts the max-relative convolution [30] for its computational and memory efficiency. To further improve the efficiency, we replace the expensive dynamic k-NN graphs with radius-based ball queries [43] for point aggregation. Moreover, we follow the hierarchical design of VGG network by repeatedly sub-sampling the point cloud via farthest point sampling, as opposed to maintaining the full set of points throughout the model [30]. We illustrate our encoder design in Fig. 4. The output of our encoder is a sub-sampled, featurized point cloud.

3.3. Stylizing the Point Cloud

Going further, our model injects style into the content features. The technical barrier here is the misalignment of content and style features, as the former are defined on a 3D point cloud while the latter (from a pre-trained VGG network) lie in a 2D plane. To address this discrepancy, we make use of learned feature mappings and Adaptive Attention Normalization (AdaAttN) [36] to match and combine the content and style features. Let F_c be the point-wise content features and F_s the style features on a 2D grid. Our style transfer operation is given by

$$F_{cs} = \psi(\text{AdaAttN}(\phi(F_c), F_s)), \tag{1}$$

where ϕ and ψ, implemented as point-wise multi-layer perceptrons (MLPs), are learned mappings between the content and style feature spaces, and AdaAttN is the attention-weighted adaptive instance normalization from [36]. AdaAttN computes attention between every content feature (a point) and each style feature (a pixel), and uses the attention map to modulate the affine parameters within the instance normalization applied on content features. As a result, F_{cs} incorporates both content and style, and will be further used to render stylized images.

3.4. Stylized Neural Rendering

Our final step is to render stylized point features F_{cs} into stylized images with specified viewpoints. As illustrated in Fig 4, this is accomplished by (1) projecting point features to an image plane given camera pose and intrinsics; and (2) decoding the projected features into an image using a 2D convolutional network.

Feature Rasterization. Our rasterizer follows Niklaus *et al*. [40], and projects the point cloud features F_{cs} into a single-view 2D feature map F_{2d}. There is one important difference: we up-sample F_{cs} using inverse distance weighted interpolation [43] *before rasterization*. This is reminiscent of super-sampling — a classical anti-aliasing technique in graphics. In doing so, we grant more flexibility for decoding the projected features into stylized images.

Image Decoding. Our decoder further maps the 2D feature map F_{2d} to a stylized RGB image at input resolution. The decoder is realized using a 2D convolutional network, following the architecture of U-Net [47], with transposed convolutions at the entry of each stage for up-sampling.

4. Learning from 2D Images

We now present our training scheme. Our model is trained using 2D images following a two-stage approach.

Generating Multi-view Images for Training. Training our model requires images from multiple views of the same scene. Unfortunately, a large-scale multi-view image dataset with a diverse set of scenes is lacking. To bridge this gap, we propose to learn from the results of existing one-shot 3D photography methods. Concretely, we use 3DPhoto [50] to convert images from a standard dataset (MS-COCO) into high-quality 3D meshes, from which we synthesize *arbitrary* pseudo target views to train our model. In doing so, our model learns from a diverse collection of scenes present in MS-COCO. Learning from synthesized images leads to an inevitable bias residing in 3DPhoto results in trade of dataset diversity. Through our experiments, we show that our model generalizes well across a large set of in-the-wild images at inference time.

4.1. Two-Stage Training

The training of our model is divided into a *view synthesis* stage where the model learns 3D geometry aware features for novel view synthesis, and a *stylization* stage where the model is further trained for novel view stylization.

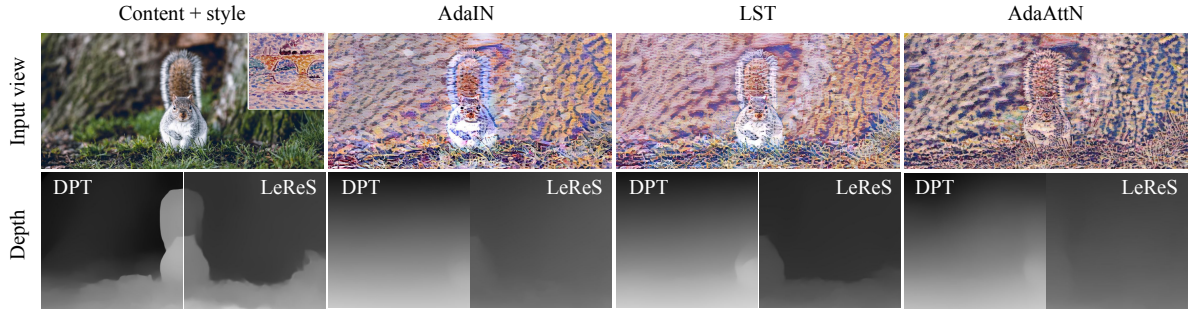


Figure 5. **Depth estimation fails on stylized images.** One alternative to 3D photo stylization is to combine stylized content image and its depth estimate. Unfortunately, strong depth estimators such as DPT [44] and LeReS [59] fail on image style transfer output from AdaIN [20], LST [32] and AdaAttN [36] because stylized images do not follow natural image statistics.

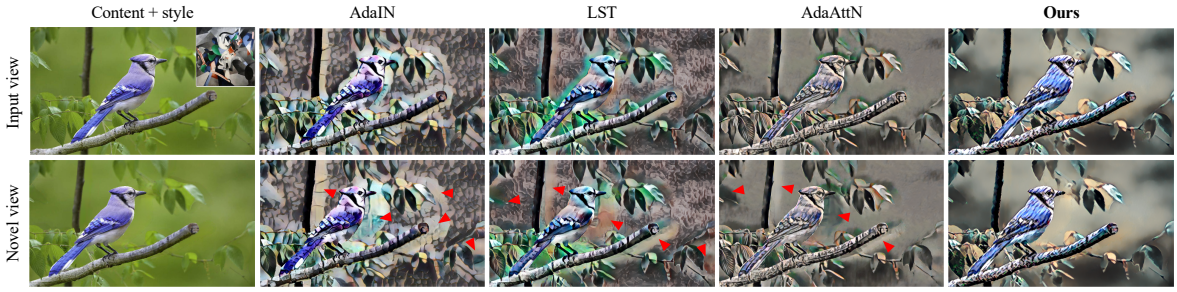


Figure 6. **3D photo of a stylized content image manifests ubiquitous visual artifacts.** Another alternative to stylizing 3D photos is to combine stylized content image with depth estimate from the *original* content image. While depth estimation is unaffected, the style effect bleeds through depth discontinuities. 3D photo inpainting thus fails, with ubiquitous visual artifacts (**red** arrows) in novel view renderings.

Enforcing Multi-view Consistency. A key technical contribution of our work is a multi-view consistency loss. Building a point cloud representation of the input content image allows us to impose additional constraint on *pixel values* of the rendered images.[2] The key idea is that a scene point $\mathbf{p}$ in the point cloud $\mathbf{P}$ should produce the same pixel color in the views to which it is visible. To this end, we define our consistency loss as

$$\mathcal{L}_{cns} = \sum_{\mathbf{p}\in\mathbf{P}} \sum_{i,j\in\mathbf{V}} \mathcal{V}(p;i,j) \cdot \|\mathbf{I}_i(\pi_i(\mathbf{p})) - \mathbf{I}_j(\pi_j(\mathbf{p}))\|_1, \quad (2)$$

where $\mathbf{V}$ is the set of sampled views, $\mathbf{I}_i$ the rendered image from view i, $\pi_i(\cdot)$ the projection to view i, and $\mathcal{V}(p;\cdot,\cdot)$ a visibility function which evaluates to 1 if p is visible to both views and 0 otherwise. Computing the loss incurs minimal overhead since the evaluation of π and $\mathcal{V}$ is part of rasterization. As evidenced by our ablation study, our proposed loss significantly improves consistency of stylized renderings.

View Synthesis Stage. We first train our model for view synthesis, a surrogate task that drives the learning of geometry aware content features. Given an input image, we randomly sample novel views of the scene and ask the model to reconstruct them. To train our model, we make use of an L1 loss $\mathcal{L}_{rgb}$ defined on pixel values, a VGG perceptual loss $\mathcal{L}_{feat}$ defined on network features, and our multi-view consistency loss $\mathcal{L}_{cns}$. The overall loss function is

$$\mathcal{L}_{view} = \mathcal{L}_{rgb} + \mathcal{L}_{feat} + \mathcal{L}_{cns}, \quad (3)$$

Stylization Stage. Our model learns to stylize novel views in the second stage. We freeze the encoder for content feature extraction, train the stylizer, and fine-tune the neural renderer. This is done by randomly sampling novel views of the scene and style images from WikiArt [39], and training our model using

$$\mathcal{L}_{style} = \mathcal{L}_{adaattn} + \mathcal{L}_{cns}, \quad (4)$$

where $\mathcal{L}_{adaattn}$ is the same AdaAttN loss from [36] and $\mathcal{L}_{cns}$ is again our multi-view consistency loss.

Training Details. For view synthesis, we train for 20K iterations (2 epochs) on MS-COCO with a batch size of 8 using Adam [26] and set the learning rate to 1e-4. We apply the same training schedule for stylization.

5. Experiments

We now present the main results of our paper and leave additional results to the supplementary material.

5.1. Qualitative results

By permuting the steps of (1) depth estimation, (2) inpainting, (3) rendering and (4) style transfer, one could imagine two alternative workflows that combine existing models for 3D photo stylization. To compare them with our method, we instantiate these baselines by combining six

[2] While the sharing of a featurized point cloud entails multi-view consistency of rasterized *feature maps*, the features are subject to a learnable decoding process, through which inconsistency will be introduced.

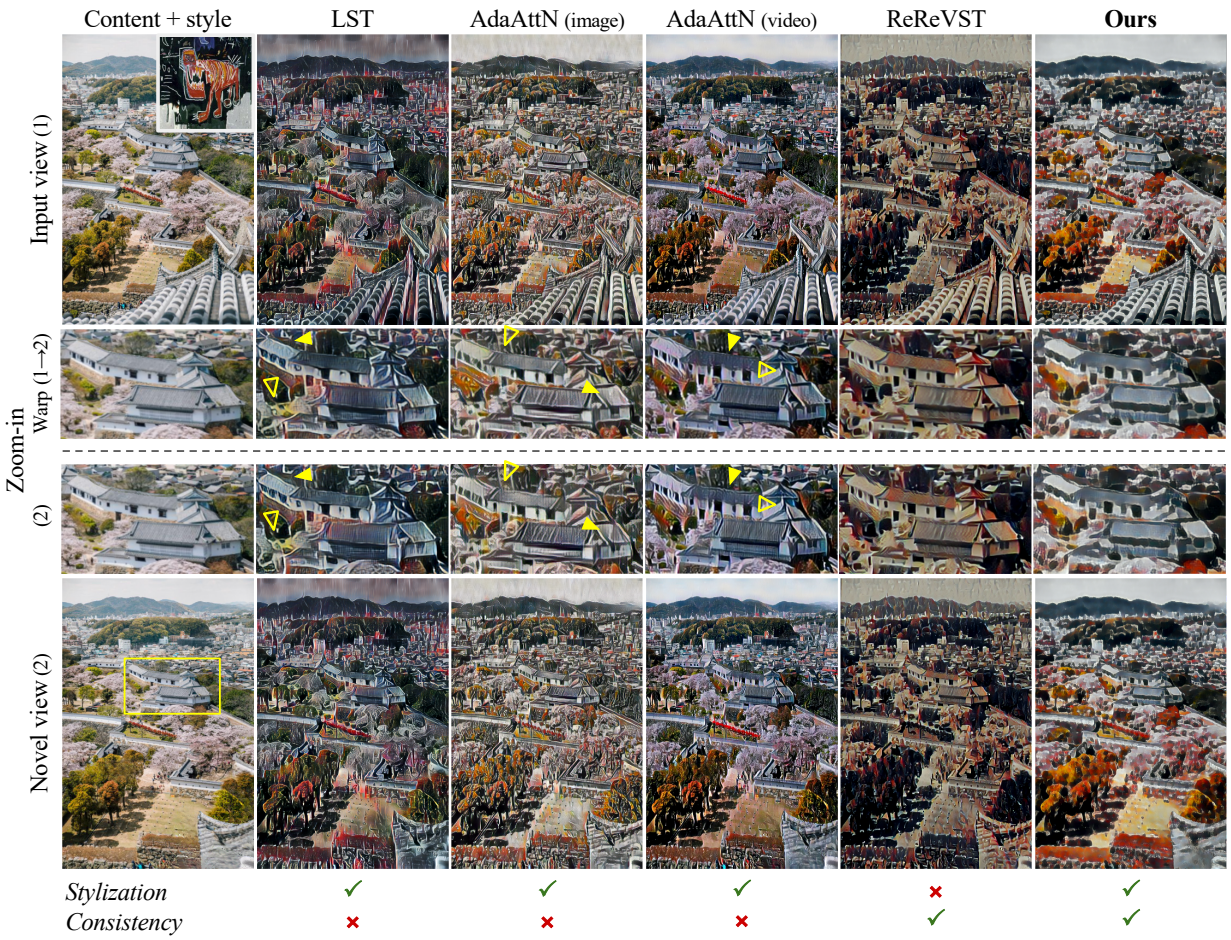


Figure 7. **Stylizing rendered images from a 3D photo introduces inconsistency in stylization.** A third baseline is to naïvely build a 3D photo from the raw content image, then stylize its renderings either one view at a time (*e.g.*, using LST [32] or AdaAttN [36]) or collectively as a video (*e.g.*, using ReReVST [55] or the video variant of AdaAttN). Despite stronger results than the other two baselines, the stylization is agnostic to the scene geometry shared by all views and thus produces inconsistent results (yellow arrows).

different style transfer methods (AdaIN [22], LST [32] and AdaAttN [36] for image style transfer, and ReReVST [55], MCC [9] and the video variant of AdaAttN for video style transfer) with DPT [44] for depth estimation and 3DPhoto [50] for inpainting and rendering. Results are created using images from Unsplash [1], a free-licensed, professional-grade dataset of in-the-wild images.

(1) *Style* → *Depth* → *Inpainting* → *Rendering*: While geometric consistency is granted, depth estimation fails catastrophically on stylized images (Figure 5). One may alternatively back-project a stylized image using depth estimation from the raw input. Despite better geometry, inpainting remains error-prone due to color bleed-through and shift in color distribution caused by stylization (Figure 6).

(2) *Depth* → *Inpainting* → *Rendering* → *Style*: This baseline often produces inconsistent stylization across views (Figure 7), as each view's style is independent and agnostic to the underlying scene geometry.

In contrast, our method manages to generate high-quality stylized renderings free of visual artifacts and inconsistency. The second baseline produces gentle inconsistency under small viewpoint change typical to 3D photo browsing. This is more benign than the visual artifacts produced by the first baseline. We further compare our method with the second baseline via quantitative experiments and a user study.

5.2. Quantitative results

Given that evaluation of style quality is a very subjective matter, we defer it to the user study and focus on the evaluation of consistency in our quantitative experiments.

Evaluation Protocol and Metrics. We run our method and the baseline on ten diverse content images from the web and 40 styles sampled from the compilation of Gao *et al*. [11]. The baseline, as discussed before, runs 3DPhoto to synthesize *plain* novel-view images, then stylizes them using one of the six style transfer algorithms. Ultimately, this results in 400 stylized 3D photos from each of the seven candidate methods. To quantify inconsistency between a pair of stylized views, we warp one view to the other according to the point cloud based scene geometry, and compute RMSE and

Method		RMSE	LPIPS
3DPhoto [50] →	AdaIN [22]	0.222	0.304
	LST [32]	0.195	0.287
	AdaAttN (image) [36]	0.187	0.329
	ReReVST [55]	0.115	0.213
	MCC [9]	0.092	0.200
	AdaAttN (video) [36]	0.135	0.209
Ours		**0.086**	**0.133**

Table 1. **Results on consistency.** We compare our model against baselines that sequentially combine 3DPhoto and image/video style transfer on consistency using RMSE (↓) and LPIPS (↓).

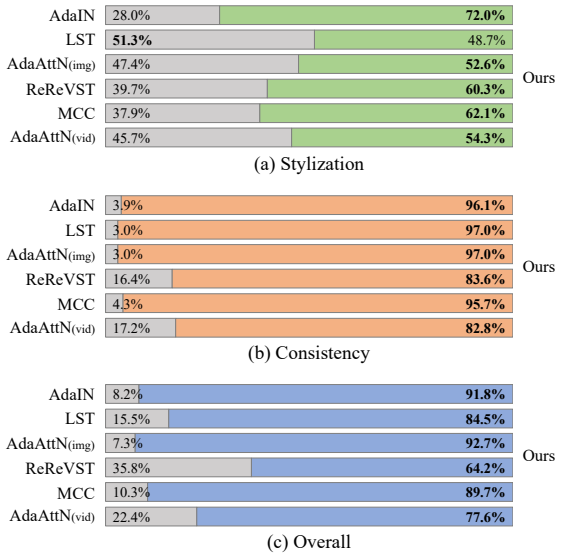


Figure 8. **User study.** We conduct a user study to compare our method against baselines that sequentially combine 3DPhoto and image/video style transfer. Methods are evaluated on (a) style quality, (b) multi-view consistency and (c) overall synthesis quality. Results show percentage of users voting for an algorithm.

the masked LPIPS metric as defined in Huang *et al*. [21]. We average the result over 400 pairs of views for each stylized 3D photo and report the mean over all available photos.

Results. Our results are summarized in Table 1. Our method outperforms all six instantiations of the baseline by a significant margin in terms of both RMSE and LPIPS. Not surprisingly, video style transfer methods produce more consistent results than image style transfer methods owing to their extra smoothness constraint. The fact that our method performs even better without such a constraint shows the effectiveness of maintaining a central featurized point cloud for 3D photo stylization.

5.3. User study

Going further, we conduct a user study to better understand the perceptual quality of stylized images produced by our method and the baselines. Our study includes three sections for the assessment of style quality, multi-view consistency and overall synthesis quality. Our analysis is based on 5,400 votes from 30 participants. We elaborate on our study design in the supplementary material.

Results. We visualize the results in Figure 8. For style quality, our method is consistently rated better than the alternatives, with the only exception being LST, which our method is on par with. Not coincidentally, our method excels at multi-view consistency, harvesting an overwhelming 95 percent of the votes in four of the six tests. Finally, our method remains the most preferred for overall synthesis quality, beating all alternatives by a large gap. Putting things together, our results provide solid validation on the strength of our approach in producing high-quality stylization that is consistent across views.

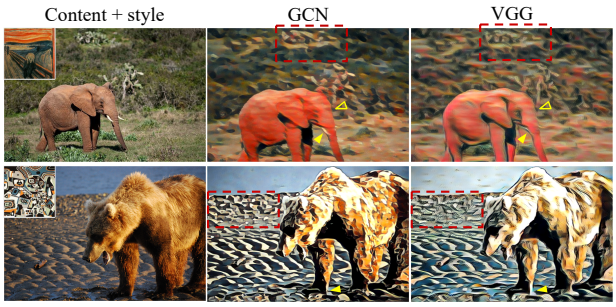


Figure 9. **Effect of geometry-aware feature learning.** 3D photo stylization with back-projected 2D VGG features suffers from geometric distortion (yellow arrows) and visual artifacts (**red** boxes). In contrast, our geometry-aware learning scheme better maintains content structure and produces more pleasant texture.

Training stage		RMSE	LPIPS
ViewSyn	*Stylize*		
−	−	0.113	0.199
+	−	0.109	0.190
−	+	**0.081**	0.132
+	+	0.086	**0.128**

Table 2. **Effect of consistency loss.** We compare models trained with (+) or without (-) the loss using RMSE (↓) and LPIPS (↓).

5.4. Ablation studies

Effect of Geometry-aware Feature Learning. We study the strength of geometry-aware feature learning. Specifically, we construct a variant of our model with the only difference that content features are not learned on the point cloud, but rather come from a pre-trained VGG network as in 2D style transfer methods. In particular, we sidestep our proposed GCN encoding scheme by projecting an RGB point cloud to eight extreme views defined by a bounding volume, running the VGG encoder for feature extraction, and back-projecting the 2D features to a point cloud from which stylization and rendering proceed as before. As shown in Fig 9, this VGG-based variant produces geometric distortion and visual artifacts in stylized images, as opposed to our model using geometry-aware feature learning.

Effect of Consistency Loss. We evaluate the contribution of our consistency loss in Table 2. Despite a shared point cloud, model trained without the consistency loss pro-

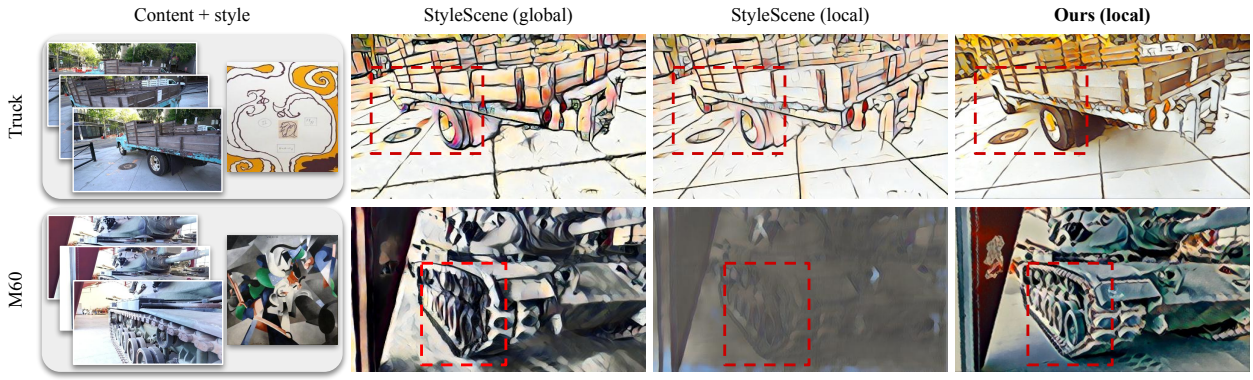


Figure 10. **Extension to multi-view input.** Compared with StyleScene [21], our method more closely resembles the reference style, better preserves the content geometry (**red** boxes), and is more robust to change in viewpoint distribution (second row).

Method	Short-range consistency								Long-range consistency							
	Truck		*Playground*		*Train*		*M60*		*Truck*		*Playground*		*Train*		*M60*	
	RMSE	LPIPS	RMSE	LPIPS	RMSE	LPIPS	RMSE	LPIPS	RMSE	LPIPS	RMSE	LPIPS	RMSE	LPIPS	RMSE	LPIPS
StyleScene (global)	0.124	0.143	0.108	0.142	0.121	0.157	0.120	0.143	0.163	0.188	0.146	0.189	0.159	0.213	0.160	0.192
StyleScene (local)	0.119	0.168	0.127	0.169	0.161	0.169	N/A	N/A	0.152	0.203	0.166	0.205	0.204	0.220	N/A	N/A
Ours (local)	**0.099**	**0.107**	**0.093**	**0.111**	**0.104**	**0.112**	**0.117**	**0.112**	**0.113**	**0.128**	**0.110**	**0.127**	**0.120**	**0.145**	**0.136**	**0.136**

Table 3. **Consistency in the multi-view scenario.** On the Tanks and Temples dataset [28], we compare our method with StyleScene on short- and long-range consistency as defined in [21] using RMSE (↓) and LPIPS (↓).

duces less consistent renderings measured in RMSE and LPIPS. We attribute this to the learnable feature decoding step, which is too flexible to preserve consistency in output images in the absence of a constraint. In this respect, our consistency loss, especially when applied in the stylization stage of training, acts as a strong regularizer on the decoder.

5.5. Extension to Multi-view Inputs

Our method can be easily extended for stylized novel view synthesis given multi-view inputs. We compare our extension with StyleScene [21], which similarly operates on point cloud but requires multiple input views. We perform experiments on the Tanks and Temples dataset [28] under two protocols. The *global* protocol uses all available views (up to 300) as in [21] for point cloud reconstruction, whereas the more challenging *local* protocol uses a sparse set of 6-8 views on the camera trajectory for novel view synthesis. In Fig 10 and Table 3, we show that our method is better in terms of style quality, short- and long-range consistency, and robustness to the distribution of input views.

5.6. Applications

Layered Stylization for AR applications. Human centered photography is of central interest in mobile AR applications. As a proof-of-concept experiment to demonstrate our method's potential in AR, we apply PointRend [27] to segment foreground human subjects in images from Unsplash [1], and stylize the background scene using our method while leaving the foreground human untouched (Fig 11a). The final stylized 3D photo upon rendering initiates a virtual tour into a 3D environment in an artistic style.
3D Exploration of Stylized Historical Photos. Historical photos represent a large fraction of existing image assets

Figure 11. **Demonstration of Applications.** Layered stylization for AR *(upper)* and 3D browsing of a stylized historical photo[3] *(lower)*—"A small arch welcomes the President to Metlakatla, Alaska, created by D. L. Hollandy 1923."

and remain under-explored in computer vision and graphics. As we demonstrate on the Keystone dataset [37] (Fig 11b), our method can be readily applied for the 3D browsing of historical photos in an artistic style, bringing past moments back alive in an unexpected way.

6. Discussion

In this paper, we connected neural style transfer and one-shot 3D photography for the first time, and introduced the novel task of 3D photo stylization – generating stylized novel views from a single image given an arbitrary style. We showed that a naïve combination of solutions from the two worlds do not work well, and proposed a deep model that jointly models style transfer and view synthesis for high-quality 3D photo stylization. We demonstrated the strength of our approach using extensive qualitative and quantitative studies, and presented interesting applications of our method for 3D content creation. We hope our method will open an exciting avenue of applications in 3D content creation from 2D photos.

References

[1] Unsplash dataset. https://unsplash.com/data, 2020. 6, 8

[2] 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. 2

[3] Dongdong Chen, Jing Liao, Lu Yuan, Nenghai Yu, and Gang Hua. Coherent online video style transfer. In *ICCV*, pages 1105–1114, 2017. 2

[4] Dongdong Chen, Lu Yuan, Jing Liao, Nenghai Yu, and Gang Hua. Stylebank: An explicit representation for neural image style transfer. In *CVPR*, 2017. 1, 2

[5] Dongdong Chen, Lu Yuan, Jing Liao, Nenghai Yu, and Gang Hua. Stereoscopic neural style transfer. In *CVPR*, 2018. 2

[6] Tian Qi Chen and Mark Schmidt. Fast patch-based style transfer of arbitrary style. *Workshop in Constructive Machine Learning, NeurIPS*, 2016. 2

[7] Xu Chen, Jie Song, and Otmar Hilliges. Monocular neural image based rendering with continuous view control. In *ICCV*, 2019. 2

[8] Pei-Ze Chiang, Meng-Shiun Tsai, Hung-Yu Tseng, Wei-sheng Lai, and Wei-Chen Chiu. Stylizing 3d scene via implicit representation and hypernetwork. *arXiv*, 2021. 2

[9] Yingying Deng, Fan Tang, Weiming Dong, Haibin Huang, Chongyang Ma, and Changsheng Xu. Arbitrary video style transfer via multi-channel correlation. In *AAAI*, 2021. 2, 6, 7, 11

[10] Vincent Dumoulin, Jonathon Shlens, and Manjunath Kudlur. A learned representation for artistic style. In *ICLR*, 2017. 2

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

[12] Leon A Gatys, Alexander S Ecker, and Matthias Bethge. A neural algorithm of artistic style. *arXiv*, 2015. 2

[13] Leon A Gatys, Alexander S Ecker, and Matthias Bethge. Image style transfer using convolutional neural networks. In *CVPR*, 2016. 1, 2

[14] Golnaz Ghiasi, Honglak Lee, Manjunath Kudlur, Vincent Dumoulin, and Jonathon Shlens. Exploring the structure of a real-time, arbitrary neural artistic stylization network. In *BMVC*, 2017. 1

[15] Xinyu Gong, Haozhi Huang, Lin Ma, Fumin Shen, Wei Liu, and Tong Zhang. Neural stereoscopic image style transfer. In *ECCV*, 2018. 2

[16] Shuyang Gu, Congliang Chen, Jing Liao, and Lu Yuan. Arbitrary style transfer with deep feature reshuffle. In *CVPR*, 2018. 1

[17] Jie Guo, Mengtian Li, Zijing Zong, Yuntao Liu, Jingwu He, Yanwen Guo, and Ling-Qi Yan. Volumetric appearance stylization with stylizing kernel prediction network. *TOG*, 2021. 2

[18] Ronghang Hu, Nikhila Ravi, Alexander C Berg, and Deepak Pathak. Worldsheet: Wrapping the world in a 3d sheet for view synthesis from a single image. In *ICCV*, 2021. 2

[19] Binh-Son Hua, Minh-Khoi Tran, and Sai-Kit Yeung. Pointwise convolutional neural networks. In *CVPR*, 2018. 2

[20] 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. 2, 5

[21] Hsin-Ping Huang, Hung-Yu Tseng, Saurabh Saini, Maneesh Singh, and Ming-Hsuan Yang. Learning to stylize novel views. In *ICCV*, 2021. 1, 2, 3, 7, 8, 11, 12

[22] Xun Huang and Serge Belongie. Arbitrary style transfer in real-time with adaptive instance normalization. In *ICCV*, 2017. 1, 2, 6, 7, 11

[23] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, 2015. 11

[24] Varun Jampani, Huiwen Chang, Kyle Sargent, Abhishek Kar, Richard Tucker, Michael Krainin, Dominik Kaeser, William T Freeman, David Salesin, Brian Curless, et al. Slide: Single image 3d photography with soft layering and depth-aware inpainting. In *ICCV*, 2021. 2

[25] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *ECCV*, 2016. 1, 2

[26] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015. 5

[27] Alexander Kirillov, Yuxin Wu, Kaiming He, and Ross Girshick. Pointrend: Image segmentation as rendering. In *CVPR*, 2020. 8

[28] Arno Knapitsch, Jaesik Park, Qian-Yi Zhou, and Vladlen Koltun. Tanks and temples: Benchmarking large-scale scene reconstruction. *TOG*, 2017. 8

[29] 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. *TOG*, 2020. 2

[30] Guohao Li, Matthias Müller, Guocheng Qian, Itzel Carolina Delgadillo Perez, Abdulellah Abualshour, Ali Kassem Thabet, and Bernard Ghanem. Deepgcns: Making gcns go as deep as cnns. *TPAMI*, 2021. 2, 3, 4, 11

[31] Jiaxin Li, Zijian Feng, Qi She, Henghui Ding, Changhu Wang, and Gim Hee Lee. Mine: Towards continuous depth mpi with nerf for novel view synthesis. In *ICCV*, 2021. 2

[32] Xueting Li, Sifei Liu, Jan Kautz, and Ming-Hsuan Yang. Learning linear transformations for fast image and video style transfer. In *CVPR*, 2019. 5, 6, 7, 11

[33] Yijun Li, Chen Fang, Jimei Yang, Zhaowen Wang, Xin Lu, and Ming-Hsuan Yang. Universal style transfer via feature transforms. In *NeurIPS*, 2017. 1, 2

[34] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *ECCV*, 2014. 2

[35] Guilin Liu, Fitsum A Reda, Kevin J Shih, Ting-Chun Wang, Andrew Tao, and Bryan Catanzaro. Image inpainting for irregular holes using partial convolutions. In *ECCV*, 2018. 2

[36] Songhua Liu, Tianwei Lin, Dongliang He, Fu Li, Meiling Wang, Xin Li, Zhengxing Sun, Qian Li, and Errui Ding. Adaattn: Revisit attention mechanism in arbitrary neural style transfer. In *ICCV*, 2021. 1, 2, 4, 5, 6, 7, 11

[37] Xuan Luo, Yanmeng Kong, Jason Lawrence, Ricardo Martin-Brualla, and Steven M. Seitz. Keystonedepth: History in 3d. In *3DV*, 2020. 8

[38] Steve Marschner and Peter Shirley. *Fundamentals of computer graphics*. 2021. 3

[39] Kiri Nichol. Painters by numbers, wikiart. https://www.kaggle.com/c/painter-by-numbers, 2016. 5

[40] Simon Niklaus, Long Mai, Jimei Yang, and Feng Liu. 3d ken burns effect from a single image. *TOG*, 2019. 2, 4, 11

[41] Dae Young Park and Kwang Hee Lee. Arbitrary style transfer with style-attentional networks. In *CVPR*, 2019. 1

[42] 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. 2

[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 *NeurIPS*, 2017. 2, 4, 11

[44] René Ranftl, Alexey Bochkovskiy, and Vladlen Koltun. Vision transformers for dense prediction. 2021. 2, 5, 6

[45] René Ranftl, Katrin Lasinger, David Hafner, Konrad Schindler, and Vladlen Koltun. Towards robust monocular depth estimation: Mixing datasets for zero-shot cross-dataset transfer. *TPAMI*, 2020. 2

[46] Chris Rockwell, David F Fouhey, and Justin Johnson. Pixelsynth: Generating a 3d-consistent experience from a single image. In *ICCV*, 2021. 2

[47] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *MICCAI*, 2015. 4, 11

[48] Manuel Ruder, Alexey Dosovitskiy, and Thomas Brox. Artistic style transfer for videos and spherical images. *IJCV*, 2018. 2

[49] Lu Sheng, Ziyi Lin, Jing Shao, and Xiaogang Wang. Avatar-net: Multi-scale zero-shot style transfer by feature decoration. In *CVPR*, 2018. 1

[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, 3, 4, 6, 7, 11

[51] Hugues Thomas, Charles R. Qi, Jean-Emmanuel Deschaud, Beatriz Marcotegui, François Goulette, and Leonidas J. Guibas. Kpconv: Flexible and deformable convolution for point clouds. In *ICCV*, 2019. 2

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

[53] Shubham Tulsiani, Richard Tucker, and Noah Snavely. Layer-structured 3d scene inference via view synthesis. In *ECCV*, 2018. 2

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

[55] Wenjing Wang, Jizheng Xu, Li Zhang, Yue Wang, and Jiaying Liu. Consistent video style transfer via compound regularization. In *AAAI*, 2020. 2, 6, 7, 11

[56] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E. Sarma, Michael M. Bronstein, and Justin M. Solomon. Dynamic graph cnn for learning on point clouds. *TOG*, 2019. 2

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

[58] Kangxue Yin, Jun Gao, Maria Shugrina, Sameh Khamis, and Sanja Fidler. 3dstylenet: Creating 3d shapes with geometric and texture style variations. In *ICCV*, 2021. 2

[59] Wei Yin, Jianming Zhang, Oliver Wang, Simon Niklaus, Long Mai, Simon Chen, and Chunhua Shen. Learning to recover 3d scene shape from a single image. In *CVPR*, 2021. 2, 3, 5

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

Supplementary Material

We refer to our *supplementary video* for an overview of our results and comparisons with the baselines. This document describes the technical details, the design of our user study, the details of extending our method to multi-view inputs, as well as a discussion on the limitation of our method.

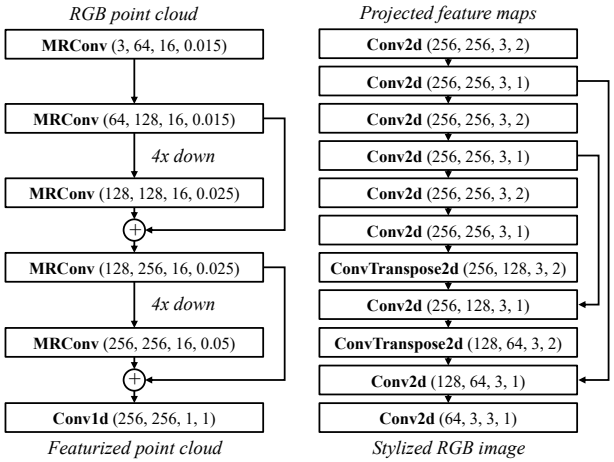
A. Implementation Details

Our model architecture is illustrated in Fig 12. We now present our implementation details.

Point Cloud Encoder Architecture. Our GCN encoder adopts a hierarchical design for computational and memory efficiency. It takes an input RGB point cloud and processes it in three stages with 1, 2 and 2 MRConv layers [30] respectively. The point features are 64, 128 and 256 dimensional after each stage. Contrary to [30], our MRConv variant performs point aggregation using ball queries, and we progressively increase the ball radius throughout the encoder to enlarge its receptive field. At the entry of each stage, we apply farthest point sampling to sub-sample the point cloud by a factor of 4. A residual connection is introduced every two layers to facilitate gradient flow during training. We apply batch normalization [23] after each layer and use ReLU as the non-linearity.

Stylizer Architecture. Our stylizer follows AdaAttN [36]. As discussed in the main paper, we apply a multi-layer perceptron (MLP) with two fully-connected layers of 256 units to map content features to the style feature space before stylization. A symmetric MLP is applied after stylization to bring the modulated features back to the content feature space. The MLPs use ReLU as the non-linearity.

Neural Renderer Architecture. Our neural renderer first up-samples the 256-dimensional encoder output via inverse distance weighted interpolation [43] until the output resolution is the same as the encoder input. The rasterizer [40] projects the up-sampled point features to the image plane of a novel view given camera pose and intrinsics. The resulting 2D feature maps have 256 dimensions and are further processed by a U-Net [47] with three levels. The encoder part of the U-Net downsamples the feature maps *without inflating the channel dimension*. We interpret this as a learnable anti-aliasing step in the same spirit as widely used supersampling in computer graphics. The decoder part subsequently up-samples the feature maps via transposed convolution and meanwhile halves the channel dimension. The skip connections, implemented as 1×1 convs, pass along feature from the encoder to the decoder to facilitate gradient flow. All layers in the U-Net except the skip convs have a kernel size of 3×3. We apply leaky ReLU with a slope of 0.2 in the encoder and ReLU in the decoder as the non-linearity.



(a) Point cloud encoder **(b) Neural renderer**

Figure 12. **Model architecture.** Architecture of our point cloud encoder and neural renderer. The layer specifications are as follows: **Conv1/2d** (input channel, output channel, kernel size, stride); **MRConv** (input channel, output channel, maximum number of neighboring points, ball radius).

B. User Study Design

We conduct a user study to compare our method with baselines that sequentially combine 3DPhoto [50] and one of the six image [22, 32, 36] or video style transfer methods [9, 36, 55]. The study includes three sections for the assessment of style quality, multi-view consistency and overall synthesis quality. Each section consists of 60 random binary choice questions that compare our method with one of the baselines. For convenience, a stylized 3D photo is displayed as a 90-frame snippet following a random camera trajectory. For fair evaluation of style quality, we only display stylized image of the input view so as not to bias participants toward more consistent renderings. Similarly, we hide the content and style images when consistency is evaluated to minimize the impact of style quality. Our analysis is based on a total of 5,400 votes collected from 30 volunteers. We show a screenshot of our user study in Fig 13. Our user study is anonymous and does not involve the collection of personally identifiable data.

C. Details on Extension to Multi-view Inputs

Extending our method to the multi-view setting is immediate after a small modification on point cloud normalization. Now that more than one input views are available, we back-project all views to a point cloud and transform it into the NDC space anchored to the center view. Everything else stays exactly the same, and importantly, the model need not be re-trained thanks to the normalization step. In our experiments, we use the same depth maps from [21] for a fair comparison with StyleScene [21]. Those results were shown in Table 3 and Figure 10 of our main paper.

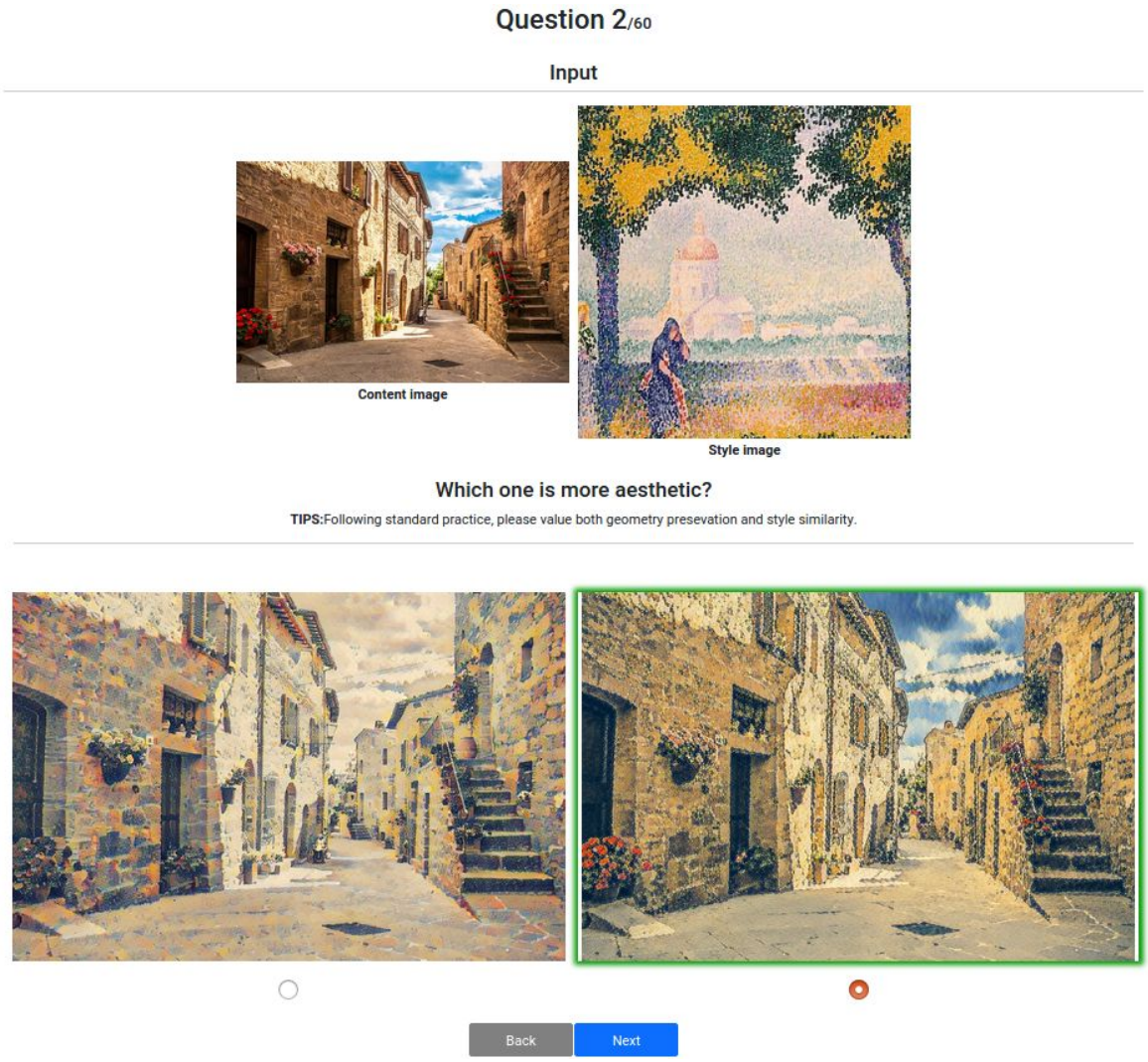


Figure 13. **Screenshot of our user study.** A randomly picked question in our user study.

D. Limitations

Despite steady progress in monocular depth estimation, current state of the arts do not always produce reliable depth maps for complex scenes, and in particular for those pixels near depth discontinuities. Our method relies on monocular depth estimation on the input image and thus inherits the failure mode of the underlying depth estimators. As a partial remedy, we have demonstrated an extension of our method to use mutli-view inputs with more reliable depth estimations. Another limitation our method shares with StyleScene [21] lies in the run-time speed. While our method renders stylized images of 1K resolution at interactive rate on a TITAN Xp GPU, the current implementation may not support interactive exploration of a high-resolution stylized 3D photo on mobile devices. Future work may focus on improving rendering speed for 3D photo stylization.

Societal impacts: We anticipate that our research would facilitate new applications of 3D content creation from 2D photos. Similar to other image manipulation methods like neural style transfer, our method might face potential copyright infringement, when copyright-protected content images are modified and improperly used.