MuVieCAST: Multi-View Consistent Artistic Style Transfer

Nail Ibrahimli, Julian F. P. Kooij, Liangliang Nan
Delft University of Technology
Julianalaan 134, Delft 2628BL, The Netherlands
{n.ibrahimli, j.f.p.kooij, liangliang.nan}@tudelft.nl

Abstract

We introduce MuVieCAST, a modular multi-view consistent style transfer network architecture that enables consistent style transfer between multiple viewpoints of the same scene. This network architecture supports both sparse and dense views, making it versatile enough to handle a wide range of multi-view image datasets. The approach consists of three modules that perform specific tasks related to style transfer, namely content preservation, image transformation, and multi-view consistency enforcement. We extensively evaluate our approach across multiple application domains including depth-map-based point cloud fusion, mesh reconstruction, and novel-view synthesis. Our experiments reveal that the proposed framework achieves an exceptional generation of stylized images, exhibiting consistent outcomes across perspectives. A user study focusing on novel-view synthesis further confirms these results, with approximately 68% of cases participants expressing a preference for our generated outputs compared to the recent state-of-the-art method. Our modular framework is extensible and can easily be integrated with various backbone architectures, making it a flexible solution for multi-view style transfer. More results are demonstrated on our project page: muviecast.github.io

1. Introduction

Style transfer has gained popularity in computer vision for its ability to create unique and visually appealing images [5, 13, 14, 21, 25, 28, 31, 33, 35, 46, 54, 64]. Conventional style transfer techniques [13, 21, 25] involve transforming an input image to mimic the style of a reference image. 3D stylization methods instead transfer styles between 3D scene representations [2, 18, 20, 22, 30, 65]. These methods typically operate on input formats such as point clouds [2, 20, 30], meshes [18], or neural radiance fields [22, 65] to generate style transfers for a single 3D scene representation.

In this work, we address multi-view style transfer, i.e. the challenge of maintaining consistency across multiple views of a static scene or object. We aim for a universal solution that works well in diverse 3D scene representation domains, removes the need for groundtruth data or precomputed 3D information, and can adapt to different datasets and view coverages [50].

More concretely, we seek a function, $f(\mathbf{I}, \mathbf{C}, \mathbf{S})$, that takes a set of calibrated multi-view images of the scene, $\mathbf{I} = I_1, I_2, ..., I_N$, with corresponding camera parameters, $\mathbf{C} = C_1, C_2, ..., C_N$, and a style image, $\mathbf{S}$, and outputs styled images that maintain both photometric and style consistency across all views. In practice, we can use Structure from Motion (SfM) [40, 41, 49] to estimate camera parameters from input images when they are not available. Formally, we define this function as $f(\mathbf{I}, \mathbf{C}, \mathbf{S}) = \mathbf{I}'$, where f transforms each input image I_i to I'_i to adopt the style of image $\mathbf{S}$. The function should capture the knowledge of both the 3D geometry and 2D image structure, while retaining the input image's content and applying the desired style, ultimately generating multi-view consistent images.

To address these challenges, we propose MuVieCAST, a modular framework for consistent, flexible, and fast style transfer between multiple views of the scene. MuVieCAST comprises multiple modules to execute various tasks including feature extraction, style transfer, and consistency enforcement. Its modular design enables integration with different computer vision backbones.

It offers unconstrained and pretrained options for style transfer, giving flexibility in choosing styles. Furthermore, we present multiple backbone options for customized network architectures, such as geometry learning [15, 55], feature extraction [51], and image transformation [21, 47].

In summary, our modular multi-view consistent style transfer framework offers a fast, versatile, and robust solution that accommodates both sparse and dense views, provides flexibility in terms of arbitrary and pretrained options for style transfer, and provides multiple backbone options, making it suitable for a diverse range of tasks and applications, such as novel-view synthesis, point cloud reconstruction, and mesh reconstruction.

2. Related work

Our work is related to neural style transfer for images, videos, and 3D scenes. We discuss each in turn.

2.1. Single-view neural style transfer

Neural style transfer [13, 25, 34] is a technique that utilizes pretrained convolutional neural networks to extract features from a content image and a style image. The goal is to transfer the style of the style image onto the content image by optimizing for loss functions. There are two types of neural style transfer methods: optimization-based and feed-forward-based approaches.

Optimization-based methods involve iterative optimization to minimize content and style losses. The popular approach involves calculating the Gram matrix [13], which captures the correlations between feature maps, to compute the style loss. Researchers have explored alternative style loss formulations to enhance semantic consistency and high-frequency style details. Several methods [5, 29, 31, 35] use nearest neighbor search and minimize distances between features extracted from corresponding content and style patches in a coarse-to-fine manner.

The feed-forward-based methods use neural networks to transfer the style information of the style image to the input image in a single forward pass. Though the naive feed-forward approach is fast [25], it requires training on specific styles and may not generalize well to arbitrary styles. To address this issue, Huang et. al. [21] use first-order statistics to encode the style information and transform the image style via AdaIN layers. This method matches the mean and variance of intermediate features between content and style images. This allows for more flexible style transfer by adjusting the scaling and shifting of the content features based on the statistics of the style features. The WCT [33] approach further explores the covariance instead of the variance, and it also uses whitening and coloring transformation to match the second-order statistics of the input image to those of the reference image. Additionally, the LST [32] scheme leverages convolutional neural networks to reduce the computational cost of solving the transformation matrix in the WCT [33] method for real-time universal style transfer. Several works [24, 38] utilize a monocular depth prediction network to integrate depth preservation to [25] as an additional loss function.

2.2. Video stylization

Video stylization aims to transfer the style of a style image to a sequence of video frames. To solve the problem of flickering that arises with image stylization methods, many techniques [4, 6, 11, 17, 19] use optical flow constraints to train autoencoders networks that can transfer a specific style to videos. Recent developments have allowed video style transfer to be performed for arbitrary styles [9, 12, 57]. However, as shown by Huang et. al. [20], applying these methods for 3D scene stylization can result in undesirable outcomes, including blurry or inconsistent images across different novel views. In light of this, our focus in this work is to enforce consistency across multiple views.

2.3. 3D style transfer

The goal of 3D style transfer is to modify the appearance of a 3D scene to match the style of a desired image when rendered from different viewpoints. Previous methods represent real-world scenes as point clouds, triangle meshes, or radiance fields, which are then fed as input for neural networks to produce stylized renderings. For instance, some methods [20, 42] utilize 3D point clouds that are first processed with the desired style and then passed through a convolutional renderer. Stylemesh [18] applies style transfer on mesh reconstructions of indoor scenes. Recent methods [22, 65] stylize radiance fields [3, 39, 63] which are effective for novel view synthesis of real-world scenes captured at high density. However, common NeRF backbones used in style transfer [3, 22, 26, 39, 63] require dense view coverage to reconstruct accurate radiance fields. Few-shot learning NeRF methods [45, 53] are known to have limitations in handling large baseline shifts, which impact the quality of reconstructed radiance fields.

Contrary to existing methods, MuVieCAST is flexible with the number of views (dense and sparse) and shows no need for precomputing a 3D representation. It directly learns to generate stylized views and produce high-quality stylizations within a short training time. The generated outputs are sufficiently consistent, making them suitable for various downstream applications such as depth estimation, point cloud and mesh reconstruction, and novel-view synthesis.

3. Method

In this section, we present our novel approach for multi-view consistent style transfer, which transfers the style of an artistic image to a set of target images of the same scene or object while maintaining consistency across multiple views. As shown in Fig 1, our method comprises three main components: style transfer network (TransferNet), content-style feature extraction, and a multi-view stereo (MVS) based geometry learning module.

First, we use TransferNet to transform the input RGB space image to the style space. Then, we extract feature representations from both the input and transformed images using a pretrained deep neural network. These features guide the style transfer network to generate images with similar artistic features while preserving the content.

To ensure consistency across multiple viewpoints, we utilize an MVS-based geometry learning module that lever-

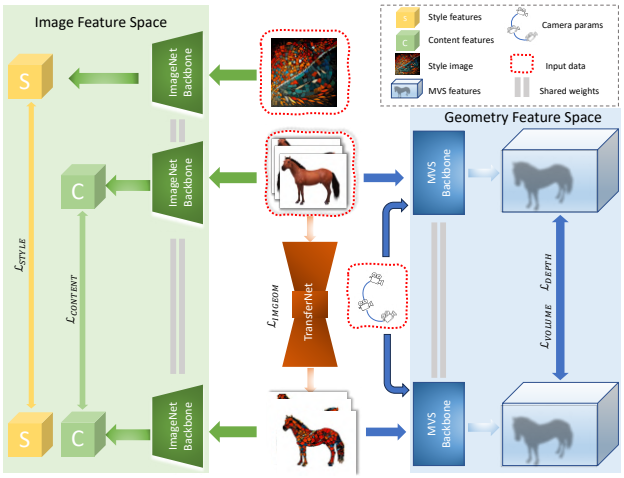


Figure 1. Our proposed network structure (MuVieCAST) has three main components: **Content-style feature extraction** (green) operates on the image feature space for preserving the content and style. **TransferNet** performs image transformation. **Geometry learning module** (blue) operates on the geometry feature space for preserving the geometry.

ages the geometric information provided by multiple views. This module uses input and transformed images as well as camera poses to estimate the 3D geometry of the scene and enforce style consistency across different views.

These components work together to produce high-quality and consistent style transfer results, even in complex multi-view scenarios. In the next subsection, we detail each component, including the content-style feature extraction, the style transfer network, and the MVS-based geometry learning module.

3.1. Modules

TransferNet Our approach employs a TransferNet module for generic style transfer. In contrast to prior multi-view approaches [22, 65], our method utilizing image-geometry loss guidance aims to enhance the texture while preserving the primary gradient structure by considering the first and second-order spatial derivatives of the original input views. This approach also assists in maintaining the finer details of the individual scene image following the stylization. Additionally, we incorporate 3D geometry loss guidance to ensure that the resulting image conforms to the overall scene geometry captured from multiple views.

Content-style feature extraction We utilize content style feature reconstruction, which extracts content and style features from pretrained Imagenet [8] backbones [51]. Pretrained weights from Imagenet are used to extract the features, and the resulting content style features guide the TransferNet to ensure that the content of the input image is preserved while the style of the artistic image is transferred onto it. It is important to note that instead of optimizing the weights of these backbones, we use them solely to extract features and guide the TransferNet. This approach allows us to focus the fast network optimization on the TransferNet, rather than learning content extraction from scratch for a longer period. By using pre-existing knowledge from the pretrained models, the TransferNet not only leads to higher quality style transfer results but also significantly reduces the computational cost of the optimization process.

Geometry learning module We choose to use multi-view stereo-based approaches as our geometry learning backbones for several reasons. First, these methods are generally applicable and can be used without further tuning. Additionally, we can leverage the pretrained modules and focus on optimizing the weights of TransferNet, which makes them particularly attractive for our multi-view consistent style transfer framework. Unlike common neural rendering or neural radiance fields approaches [22, 39, 56, 61–63, 65] used for style transfer, these networks can also robustly work with sparse views, making them more applicable in a wide range of scenarios. We discuss an ablation study involving sparse views within our supplementary materials.

Our framework supports multi-view stereo backbones [15, 55] which employ a feature pyramid network [37] to extract multiple levels of features and guide the stereo matching process. The coarse-to-fine approach involves processing the images at different scales, starting with a low resolution and gradually increasing it. This makes our backbones more memory-efficient and able to handle larger input sizes [15, 55, 59].

3.2. Loss

Our loss function is a weighted combination of loss terms. These loss terms aim to achieve multi-view consistent style transfer, preserving the primary image and geometric structures of the original views while transforming them into the generated stylized views.

Content-style loss To achieve content-aware style transfer, we use a combination of content loss and style loss. The content loss measures the difference between the feature representations of the target image and the reference image, which serves as the source of the content information. Let F_t and F_r be the feature maps of the target image and input reference image, respectively and Ω_F be the number of elements in feature maps. The content loss is then defined as follows,

$$\mathcal{L}_{\text{content}} = \frac{1}{\Omega_F} \sum \|F_t - F_r\|_2^2 \,. \tag{1}$$

We provide two options for the style loss $\mathcal{L}_{\text{style}}$. The first option is to measure the difference between the Gram matrices [13] of the feature maps of the target image and the style image, which contains the texture and color information. Let G_t and G_s be the Gram matrices of the feature maps of the target image and style image, respectively, and Ω_G be the number of elements in the Gram matrix, such that the style loss is defined as

$$\mathcal{L}_{\text{style}} = \frac{1}{\Omega_G} \sum \|G_t - G_s\|_2^2 \, . \tag{2}$$

Our second option is inspired by the work of Li et al. [34] which explores a style loss based on instance normalization statistics. This style loss is defined as

$$\mathcal{L}_{\text{style}} = \sum_{i=1}^{L} \|\mu(F_{i,t}) - \mu(F_{i,s})\|_2^2 \quad + \\ \sum_{i=1}^{L} \|\sigma(F_{i,t}) - \sigma(F_{i,s})\|_2^2 \, , \tag{3}$$

where $F_{i,t}$ and $F_{i,s}$ are the feature maps of the generated image and style image at layer i, respectively, L is the total number of layers used, μ is the mean, and σ is the standard deviation. The style loss is calculated by computing the L2 norm between the instance normalization statistics of the feature maps of the generated and style images.

Image-geometry loss To preserve the primary geometric structures of the input image in the generated image, we introduce an image-geometry loss which consists of three terms: Sobel filter loss, Laplacian operator loss, and differentiable Canny edge detector loss.

These three components capture the geometric characteristics of both the input and generated images. The overall image-geometry loss is defined as the weighted sum of the Sobel filter loss ($\mathcal{L}_{\text{Sobel}}$), Laplacian operator loss ($\mathcal{L}_{\text{Laplace}}$), and Canny edge detector loss ($\mathcal{L}_{\text{Canny}}$), which is expressed as follows:

$$\mathcal{L}_{\text{Sobel}} = \frac{1}{\Omega_I} \sum \text{smoothL1}(\nabla(I_r), \nabla(I_t)) \tag{4}$$

$$\mathcal{L}_{\text{Laplace}} = \frac{1}{\Omega_I} \sum \text{smoothL1}(\Delta(I_r), \Delta(I_t)) \tag{5}$$

$$\mathcal{L}_{\text{Canny}} = \frac{1}{\Omega_I} \sum \text{smoothL1}(C(I_r), C(I_t)) \tag{6}$$

$$\mathcal{L}_{\text{imgeom}} = \lambda_{\nabla} \mathcal{L}_{\text{Sobel}} + \lambda_{\Delta} \mathcal{L}_{\text{Laplace}} + \lambda_{\text{C}} \mathcal{L}_{\text{Canny}} \, , \tag{7}$$

where ∇ is the Sobel filter operator, Δ is the Laplacian operator, and C is the differentiable Canny edge detector. I_r and I_t correspond to the input and generated images, respectively, and Ω_I represents the total number of pixels in the images. The hyperparameters λ_{∇}, λ_{Δ}, and λ_{C} control the relative influence of each component within the overall image-geometry loss.

The smooth L1 loss function is employed to compute the discrepancy between the responses of the input and generated images for each of the three components. The inclusion of these geometric loss components in the overall loss function enables effective regularization of the neural network, preventing excessive divergence from the main geometric features of the input image.

3D geometry loss To enforce geometric consistency between the input and output images in our multi-view consistent style transfer framework, we incorporate two types of losses: volumetric loss and depth loss. Both losses use the smooth L1 loss function to calculate the differences between the corresponding elements of the input and target images.

We apply a coarse-to-fine strategy for computing these losses. Specifically, we calculate the losses at multiple stages, with a weight of 2^{3-l} at each stage l, where $l = 0$ is the finest stage. This allows the network to gradually refine the output image.

For the volumetric loss, we use an estimated probability volume for each depth and calculate the distance between the probability volumes of the input and target images. For the depth loss, we first estimate the depth map of the stylized target image and the input images using multi-view stereo backbones. We then calculate the smooth L1 loss between the estimated depth map of the input images and the generated target images. These losses are computed at each stage of the network as

$$\mathcal{L}_{\text{volume}} = \sum_{l} \frac{2^{3-l}}{\Omega_{V_{(l)}}} \cdot \sum \text{smoothL1}(V_{r(l)}, V_{t(l)}) \tag{8}$$

$$\mathcal{L}_{\text{depth}} = \sum_{l} \frac{2^{3-l}}{\Omega_{D_{(l)}}} \cdot \sum \text{smoothL1}(D_{r(l)}, D_{t(l)}) \, , \tag{9}$$

where $\Omega_{V_{(l)}}$ is the total number of voxels in the probability volume at stage l. $V_{r(l)}$ and $V_{t(l)}$ are the probability volumes of the depth of the input and target images respectively, at stage l. $\Omega_{D_{(l)}}$ is the total number of pixels in the depth map at stage l. $D_{t(l)}$ is the depth values of the stylized image at stage l, and $D_{r(l)}$ is the estimated depth values.

Total loss Total loss function is a weighted combination of loss terms for the content-style ($\mathcal{L}_{\text{content}}$ and $\mathcal{L}_{\text{style}}$), for the image-geometry ($\mathcal{L}_{\text{imgeom}}$), and for the 3D geometry ($\mathcal{L}_{\text{volume}}$ and $\mathcal{L}_{\text{depth}}$), i.e.

$$\mathcal{L}_{\text{total}} = \lambda_{\text{content}} \mathcal{L}_{\text{content}} + \lambda_{\text{style}} \mathcal{L}_{\text{style}} \quad + \\ \lambda_{\text{imgeom}} \mathcal{L}_{\text{imgeom}} + \lambda_{\text{volume}} \mathcal{L}_{\text{volume}} + \lambda_{\text{depth}} \mathcal{L}_{\text{depth}} \, . \tag{10}$$

4. Experiments

In this section, we describe our experiment setup and compare different backbones for our approach. We demonstrate our framework's multi-view consistency on three applications: depth-map-based point cloud fusion, mesh reconstruction, and novel-view synthesis.

4.1. Experiment setup

To demonstrate the modularity of our approach, we perform experiments using several backbones. Specifically, we utilized pretrained VGG16 and VGG19 modules, which were trained on the ImageNet dataset [8], as well as CasMVSNet and PatchmatchNet geometric multi-view stereo backbones pretrained on DTU dataset [1]. Additionally, we used AdaIN and UNet as style transfer backbones, which were pretrained on the MS COCO dataset [36]. Table 1 summarizes these backbones tested in our experiments. To show different network configurations, we limit our discussion to four configurations, which are presented in Table 2.

Modules	Options	Pretrained	Trainable
Image learning	VGG16	ImageNet	no
	VGG19	ImageNet	no
Geometry learning	CasMVSNet	DTU	no
	PatchMatchNet	DTU	no
TransferNet	UNet	MS COCO	yes
	AdaIN	MS COCO	yes

Table 1. Backbones used in our experiments.

Geometry learning module. PatchmatchNet [55] is a patchmatch stereo-based approach [10], where a similarity score is computed between the extracted features of each image in a coarse-to-fine manner to generate a depth map. CasMVSNet [15] uses cascaded cost volume regularization to construct a cost volume [7, 59, 60] and estimate the depth between the images. To further reduce GPU memory demands, we employ groupwise correlation [16, 58] and inplace activated batch normalization [48] within CasMVSNet.

TransferNet. The UNet backbone is a fully convolutional neural network that is widely used in image segmentation tasks [47]. In our framework, we pretrain UNet for each specific style using the MS COCO object detection dataset [36]. The pretraining is performed to learn the style-specific weights of the network, which are then tuned for style transfer. In the process of multi-view consistent style transfer, the optimization is initialized with these pretrained weights. The other backbone, AdaIN (Adaptive Instance Normalization), is a style transfer technique [21, 22] that does not require style-specific pretraining. Instead, it uses a single encoder network to extract content and style information from both the content and style images. The encoder network maps the style image to the feature space, and the content image features are transformed to this feature space using adaptive instance normalization. Finally, the transformed content features are mapped back to the image space by the decoder to generate the stylized output. This decoder is also initialized with pretrained weights on the MS COCO dataset [36].

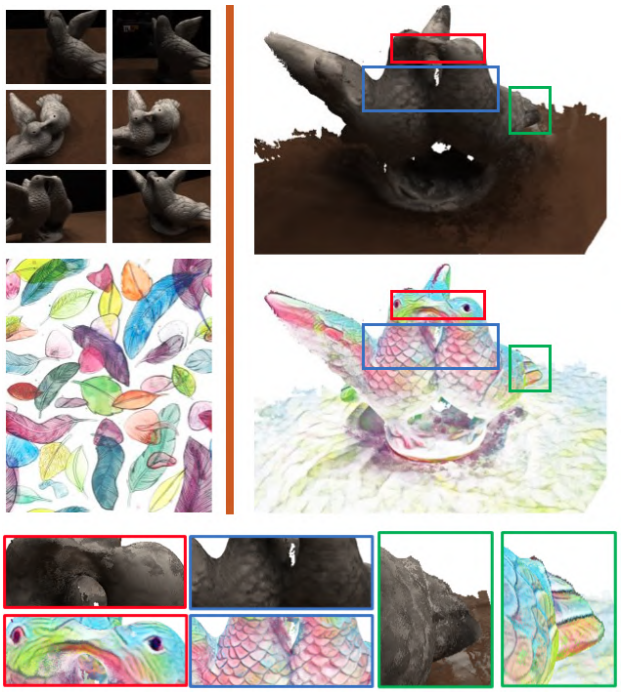


Figure 2. The bird point cloud reconstructed using depth map fusion from the 49 input images [1] and a style image. This experiment demonstrates the stylization capability of our network (CasMVSNet_UNet) to reveal finer details in textured areas, even in shadowed regions. Both point clouds have identical geometry.

In the supplementary material, we expand on our findings about using UNet's pretrained module for quick adaptation to new styles and images. By using weights learned from MS COCO object detection, we can swiftly apply UNet to new style images for given inputs. Pretraining captures a general understanding of style features and textures that can be reused, significantly reducing the time and resources needed for training new models for each style.

While both network configurations yielded visually appealing results, our experiments showed that UNet-based configurations demonstrated greater accuracy and consistency in maintaining geometry. For each experiment on point cloud and mesh reconstruction, we trained our model for 10 epochs, which took less than 5 minutes to complete on dual RTX2080 Ti.

Naming	Geometry	ImageNet	TransferNet	Style loss	Total params	Trainable params
CasMVSNet_UNet	CasMVSNet	VGG16	UNet	Gram	10.2 M	1.7 M
CasMVSNet_AdaIN	CasMVSNet	VGG19	AdaIN	IN statistics	7.9 M	3.5 M
PatchMatchNet_UNet	PatchMatchNet	VGG16	UNet	Gram	9.5 M	1.7 M
PatchMatchNet_AdaIN	PatchMatchNet	VGG19	AdaIN	IN statistics	7.2 M	3.5 M

Table 2. Different network configurations tested in the experiments.

4.2. Point cloud reconstruction

We found that our network is resilient to lighting variations in the input and attentive to texture details, even in shadowed regions. This capability becomes particularly evident when the network is trained using style images characterized by significant color variations. Figure 2 illustrates the point cloud reconstruction generated from the input views using depth map fusion [23, 60], which uses learning-based multi-view stereo matching [15]. Both point clouds contain the same set of points. Since the input images were captured under different lighting conditions, the coloring of the point cloud is blurred and finer details like the birds' eyes and body textures are not properly revealed. However, our network colors the point cloud with better detail.

Figure 3 shows a visual comparison of the point clouds reconstructed from the original input images and the stylized images. The objective of this experiment is to evaluate stylized image consistency across multiple views in this experiment. From the result, we can conclude that MuVieCAST can generate highly consistent stylized multi-view images that allow us to estimate accurate depth from the styled images and yield good 3D point cloud reconstruction. Despite the reconstructed point cloud from stylized views exhibiting more noise in its geometry, the reconstructed point cloud remains adequate for identifying the object's geometry.

To showcase the point cloud reconstruction using the PatchmatchNet backbone, we conducted several experiments, shown in Figure 4. As with our previous experiment, we aim to demonstrate the consistency of the stylization across multiple views with the PatchmatchNet backbone. We observe that the point clouds (d) and (e) are reasonably reconstructed, where the depth maps for the two point clouds were estimated from stylized images by PatchmatchNet_UNet and PatchmatchNet_AdaIN, respectively. We can observe that both point clouds are sufficient for identifying the object. During our experiments, we also observed that in most cases, the UNet backbone performed better than the AdaIN backbone in terms of geometric consistency of multi-view style transfer.

4.3. Mesh reconstruction

In this experiment, we demonstrate the efficacy and robustness of our multi-view style transfer method for neural rendering-based mesh reconstruction [44, 56, 61, 62], which has gained popularity in recent years. Figure 5 shows mesh reconstruction results using IDR [61]. Our stylization method preserves consistency for mesh representation-based geometry estimation, despite some loss of finer details in the reconstructed meshes. These results further confirm the reliability and consistency of our robust multi-view style transfer approach, for mesh reconstruction.

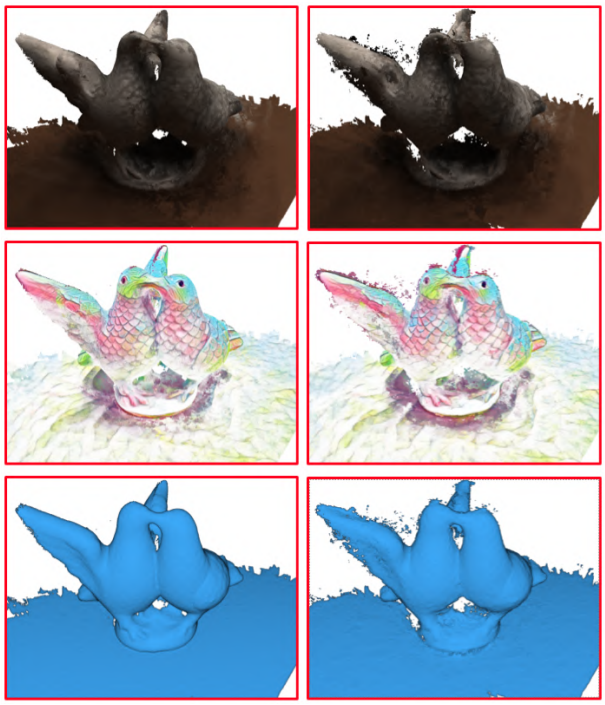


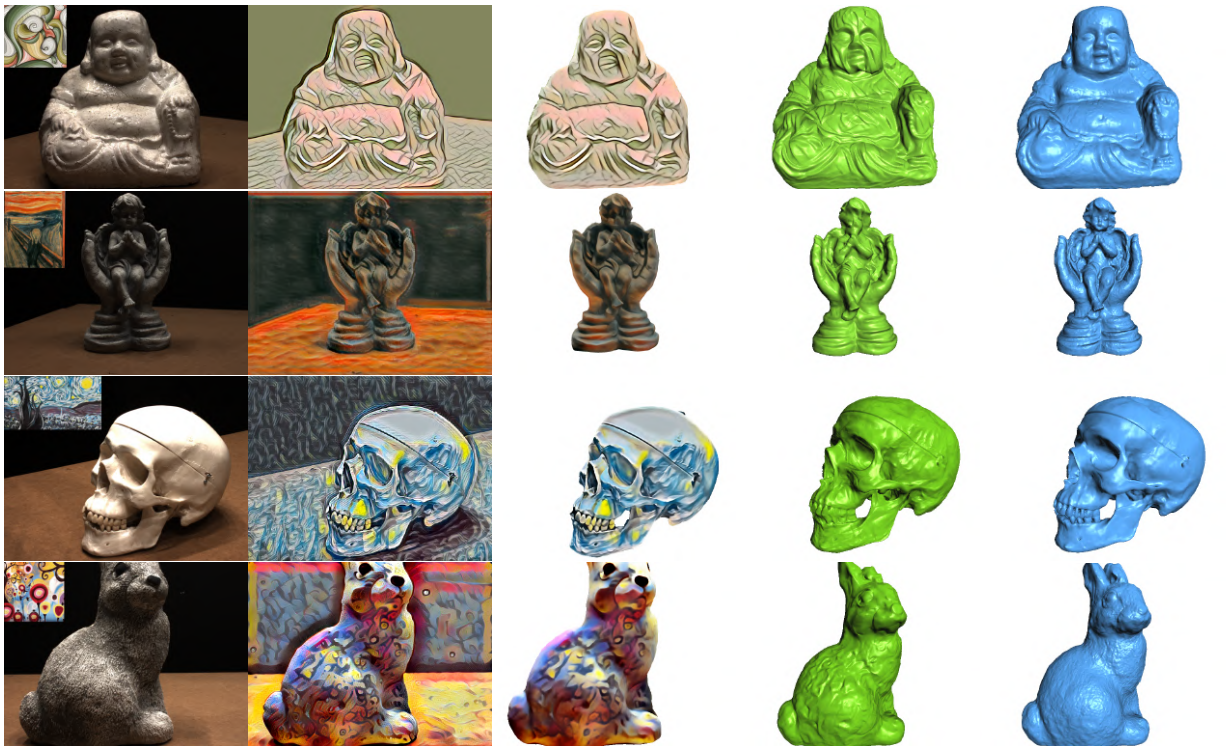
Figure 3. Comparison of the point clouds reconstructed from the original input images (left column) and stylized images (right column). Top row: colored with original inputs. Middle row: colored with stylized colors. Bottom row: uniform coloring. The reconstructed point cloud from stylized views (CasMVSNet_UNet) closely resembles that from the original input, confirming that the stylization is multi-view consistent.

4.4. Novel view synthesis

MuVieCAST offers a significant advantage over other methods [18, 22, 65] as it does not require ground truth 3D data or precomputed 3D implicit fields. To evaluate the effectiveness of our approach compared to the recent state-of-

(a) Input + Input (b) Input + UNet (c) Input + AdaIN (d) UNet + UNet (e) AdaIN + AdaIN

Figure 4. Point cloud reconstruction results using PatchmatchNet backbone (Geometry + Coloring). The point clouds (a), (b), and (c) are identical in terms of geometry, with (a) colored by the 64 original input images, (b) colored by the stylized images using the output of Patchmatchnet_UNet, and (c) colored by the stylized images using the output of Patchmatchnet_AdaIN. (d) and (e) are reconstructed from only the stylized images.



(a) Input image (b) Stylized image (c) Stylized surface rendering (d) Reconstruction from stylized images (e) Reconstruction from input images

Figure 5. Neural mesh reconstruction results. First row: CasMVSNet_UNet. Second row: CasMVSNet_AdaIN. Third row: PatchmatchNet_UNet. Fourth row: PatchmatchNet_AdaIN. The columns in the figure are as follows: (a) Input image. (b) Stylized image, (c) The stylized mesh surface rendering learned by IDR [61]. (d) The mesh reconstructed from the stylized images. (e) The mesh reconstruction from the original input images.

the-art method ARF [65], we conducted an anonymous user survey with 40 participants. During the survey, participants were provided with sample scene images and style images. The evaluation was performed on five scenes from the Tanks and Temples dataset [27], each paired with a distinct style image. For each scene, participants were presented with two videos in random order. One video was generated using the ARF method, utilizing ground truth pose information and recommended parameters.

To ensure a fair assessment, we adhered closely to the default trajectory employed by ARF. Additionally, we previously demonstrated the geometric consistency of our styl-

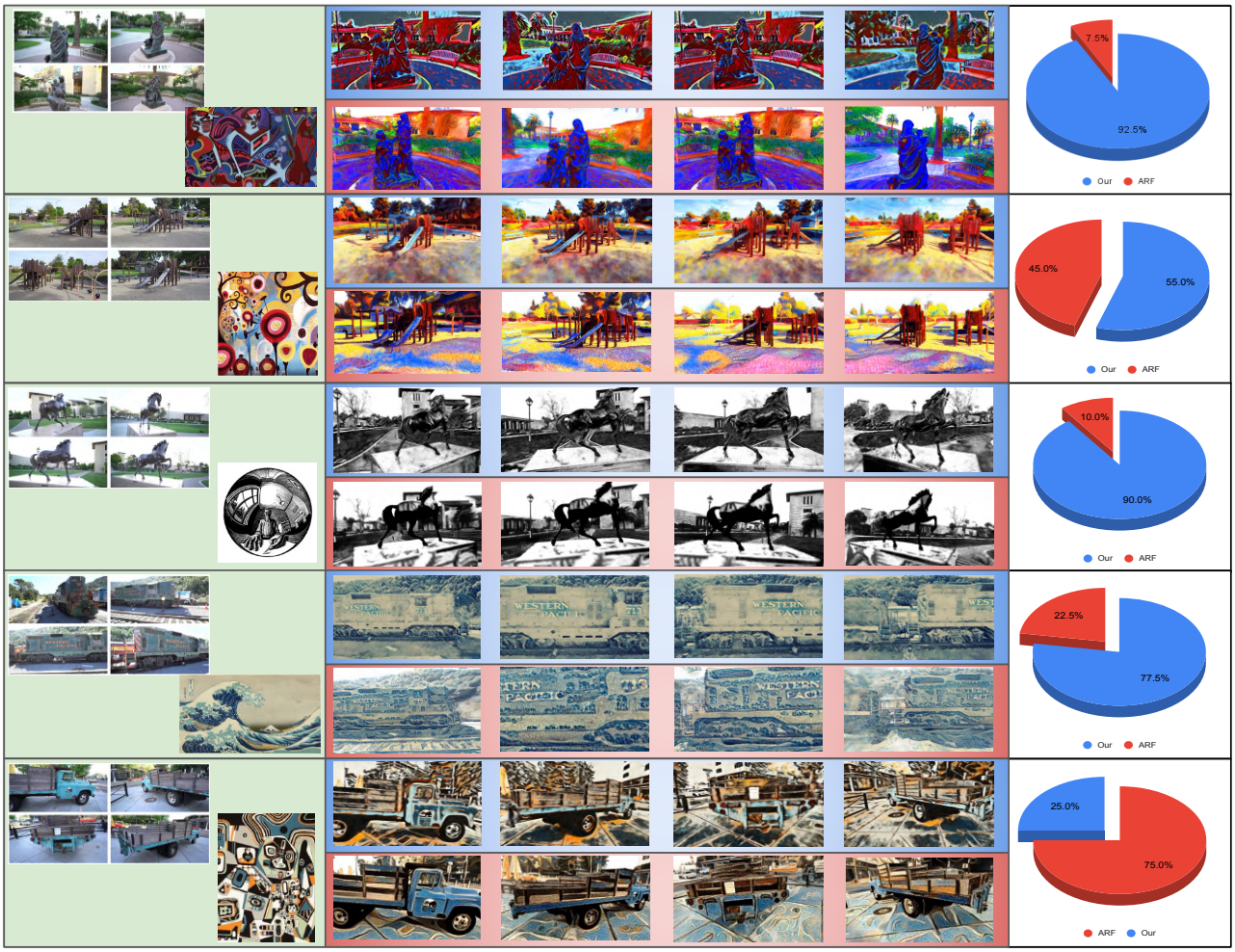


Figure 6. The left column shows the scene samples and style images shared with user study participants. In the middle column, frames from our results are presented on the top rows (on a blue stripe), while frames from the ARF method are displayed on the bottom rows (on a red stripe). The videos were provided to participants in a randomized order. The pie charts indicate the preferences of the 40 participants.

ization through mesh and point cloud reconstruction experiments. In these comparison experiments, we extended our evaluation beyond computing radiance fields [39, 43, 52, 63] to include the direct estimation of camera calibrations from stylized images using the SFM algorithm [49].

The results of the survey indicated that in 68% of the cases, participants preferred our results over the ones generated by ARF. To present the survey results for each scene, Figure 11 shows the comparison outcomes. These findings highlight the efficacy of our network architecture and its potential for practical applications in style transfer and scene generation tasks.

5. Conclusion

We have presented MuVieCAST, a multi-view style transfer architecture, that offers a fast, versatile, and robust solution for various downstream applications, including stereo matching-based point cloud reconstruction, neural mesh reconstruction, and novel-view synthesis. In contrast to other 3D style transfer methods, our proposed method does not require precomputed or groundtruth 3D scene representations (point cloud, mesh, radiance fields, signed distance functions, occupancy fields) and generates consistent stylized views directly from calibrated input views. Our experiments have demonstrated the effectiveness of different backbones in our multi-view consistent style transfer network architecture, validating our architectural ideas. The resulting stylized images are consistent for robust geometry estimation, revealing finer geometric and texture details of the underlying 3D scene representation, thereby enhancing downstream applications. Moreover, our proposed network architecture trains fast, making it an appealing option for researchers and practitioners interested in related application domains.

References

[1] Henrik Aanæs, Rasmus Ramsbøl Jensen, George Vogiatzis, Engin Tola, and Anders Bjorholm Dahl. Large-scale data for multiple-view stereopsis. *IJCV*, pages 1–16, 2016. 5

[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*, pages 3337–3345. IEEE, 2020. 1

[3] Anpei Chen, Zexiang Xu, Andreas Geiger, Jingyi Yu, and Hao Su. Tensorf: Tensorial radiance fields. In *ECCV*, pages 333–350. Springer, 2022. 2

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

[5] Tian Qi Chen and Mark Schmidt. Fast patch-based style transfer of arbitrary style. In *NeurIPSW*, 2016. 1, 2

[6] 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*, pages 614–630. Springer, 2020. 2

[7] Robert T Collins. A space-sweep approach to true multi-image matching. In *CVPR*, pages 358–363. IEEE, 1996. 5

[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*, pages 248–255. IEEE, 2009. 3, 5

[9] Yingying Deng, Fan Tang, Weiming Dong, Haibin Huang, Chongyang Ma, and Changsheng Xu. Arbitrary video style transfer via multi-channel correlation. In *AAAI*, pages 1210–1217, 2021. 2

[10] Silvano Galliani, Katrin Lasinger, and Konrad Schindler. Massively parallel multiview stereopsis by surface normal diffusion. In *ICCV*, pages 873–881. IEEE, 2015. 5

[11] Chang Gao, Derun Gu, Fangjun Zhang, and Y. Yu. Reconet: Real-time coherent video style transfer network. In *ACCV*, pages 637–653. Springer, 2019. 2

[12] Wei Gao, Yijun Li, Yihang Yin, and Ming-Hsuan Yang. Fast video multi-style transfer. In *WACV*, pages 3222–3230. IEEE, 2020. 2

[13] Leon A Gatys, Alexander S Ecker, and Matthias Bethge. Image style transfer using convolutional neural networks. In *CVPR*, pages 2414–2423. IEEE, 2016. 1, 2, 4

[14] Shuyang Gu, Congliang Chen, Jing Liao, and Lu Yuan. Arbitrary style transfer with deep feature reshuffle. In *CVPR*, pages 8222–8231. IEEE, 2018. 1

[15] Xiaodong Gu, Zhiwen Fan, Siyu Zhu, Zuozhuo Dai, Feitong Tan, and Ping Tan. Cascade cost volume for high-resolution multi-view stereo and stereo matching. In *CVPR*, pages 2495–2504. IEEE, 2020. 1, 3, 5, 6

[16] Xiaoyang Guo, Kai Yang, Wukui Yang, Xiaogang Wang, and Hongsheng Li. Group-wise correlation stereo network. In *CVPR*, pages 3273–3282. IEEE, 2019. 5, 1

[17] Agrim Gupta, Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Characterizing and improving stability in neural style transfer. In *ICCV*, pages 4067–4076. IEEE, 2017. 2

[18] Lukas Höllein, Justin Johnson, and Matthias Nießner. Stylemesh: Style transfer for indoor 3d scene reconstructions. In *CVPR*, pages 6198–6208. IEEE, 2022. 1, 2, 6

[19] 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*, pages 783–791. IEEE, 2017. 2

[20] Hsin-Ping Huang, Hung-Yu Tseng, Saurabh Saini, Maneesh Singh, and Ming-Hsuan Yang. Learning to stylize novel views. In *CVPR*, pages 13869–13878. IEEE, 2021. 1, 2

[21] Xun Huang and Serge Belongie. Arbitrary style transfer in real-time with adaptive instance normalization. In *ICCV*, pages 1501–1510. IEEE, 2017. 1, 2, 5

[22] Yi-Hua Huang, Yue He, Yu-Jie Yuan, Yu-Kun Lai, and Lin Gao. Stylizednerf: Consistent 3d scene stylization as stylized nerf via 2d-3d mutual learning. In *CVPR*, pages 18342–18352. IEEE, 2022. 1, 2, 3, 5, 6

[23] Nail Ibrahimli, Hugo Ledoux, Julian FP Kooij, and Liangliang Nan. Ddl-mvs: Depth discontinuity learning for multi-view stereo networks. *Remote Sensing*, 15(12):2970, 2023. 6

[24] E Ioannou and S Maddock. Depth-aware neural style transfer using instance normalization. In *CGVC*. Eurographics Digital Library, 2022. 2

[25] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *ECCV*, pages 694–711. Springer, 2016. 1, 2, 4

[26] Abhishek Kar, Christian Häne, and Jitendra Malik. Learning a multi-view stereo machine. In *NeurIPS*, page 364–375, 2017. 2

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

[28] Nicholas Kolkin, Jason Salavon, and Gregory Shakhnarovich. Style transfer by relaxed optimal transport and self-similarity. In *CVPR*, pages 10051–10060. IEEE, 2019. 1

[29] Nicholas Kolkin, Michal Kucera, Sylvain Paris, Daniel Sykora, Eli Shechtman, and Greg Shakhnarovich. Neural neighbor style transfer. *arXiv e-prints*, pages arXiv–2203, 2022. 2

[30] Georgios Kopanas, Julien Philip, Thomas Leimkühler, and George Drettakis. Point-based neural rendering with per-view optimization. In *Computer Graphics Forum*, pages 29–43. Wiley Online Library, 2021. 1

[31] Chuan Li and Michael Wand. Combining markov random fields and convolutional neural networks for image synthesis. In *CVPR*, pages 2479–2486. IEEE, 2016. 1, 2, 4

[32] Xueting Li, Sifei Liu, Jan Kautz, and Ming-Hsuan Yang. Learning linear transformations for fast image and video style transfer. In *CVPR*, pages 3809–3817. IEEE, 2019. 2

[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] Yanghao Li, Naiyan Wang, Jiaying Liu, and Xiaodi Hou. Demystifying neural style transfer. pages 2230–2236, 2017. 2, 4

[35] Jing Liao, Yuan Yao, Lu Yuan, Gang Hua, and Sing Bing Kang. Visual attribute transfer through deep image analogy. *ACM Trans. Graph.*, 2017. 1, 2

[36] 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*, pages 740–755. Springer, 2014. 5, 1

[37] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *CVPR*, pages 2117–2125. IEEE, 2017. 3

[38] Xiao-Chang Liu, Ming-Ming Cheng, Yu-Kun Lai, and Paul L Rosin. Depth-aware neural style transfer. In *Proceedings of the symposium on non-photorealistic animation and rendering*, pages 1–10, 2017. 2

[39] 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*, pages 99–106. Springer, 2020. 2, 3, 8

[40] Pierre Moulon, Pascal Monasse, and Renaud Marlet. Adaptive structure from motion with a contrario model estimation. In *ACCV*, pages 257–270. Springer, 2012. 1

[41] Pierre Moulon, Pascal Monasse, and Renaud Marlet. Global fusion of relative motions for robust, accurate and scalable structure from motion. In *ICCV*, pages 3248–3255. IEEE, 2013. 1

[42] Fangzhou Mu, Jian Wang, Yicheng Wu, and Yin Li. 3d photo stylization: Learning to generate stylized novel views from a single image. In *CVPR*, pages 16273–16282. IEEE, 2022. 2

[43] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM Trans. Graph.*, 41(4):102:1–102:15, 2022. 8

[44] Michael Niemeyer, Lars Mescheder, Michael Oechsle, and Andreas Geiger. Differentiable volumetric rendering: Learning implicit 3d representations without 3d supervision. In *CVPR*, pages 3504–3515. IEEE, 2020. 6

[45] Michael Niemeyer, Jonathan T Barron, Ben Mildenhall, Mehdi SM Sajjadi, Andreas Geiger, and Noha Radwan. Regnerf: Regularizing neural radiance fields for view synthesis from sparse inputs. In *CVPR*, pages 5480–5490. IEEE, 2022. 2

[46] Eric Risser, Pierre Wilmot, and Connelly Barnes. Stable and controllable neural texture synthesis and style transfer using histogram losses. *arXiv preprint arXiv:1701.08893*, 2017. 1

[47] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention*, pages 234–241. Springer, 2015. 1, 5

[48] Samuel Rota Bulò, Lorenzo Porzi, and Peter Kontschieder. In-place activated batchnorm for memory-optimized training of dnns. In *CVPR*, pages 5639–5647. IEEE, 2018. 5

[49] Johannes Lutz Schönberger and Jan-Michael Frahm. Structure-from-motion revisited. In *CVPR*, pages 4104–4113. IEEE, 2016. 1, 8

[50] Johannes Lutz Schönberger, Enliang Zheng, Marc Pollefeys, and Jan-Michael Frahm. Pixelwise view selection for unstructured multi-view stereo. In *ECCV*, pages 501–518. Springer, 2016. 1

[51] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2014. 1, 3

[52] Matthew Tancik, Ethan Weber, Evonne Ng, Ruilong Li, Brent Yi, Justin Kerr, Terrance Wang, Alexander Kristoffersen, Jake Austin, Kamyar Salahi, Abhik Ahuja, David McAllister, and Angjoo Kanazawa. Nerfstudio: A modular framework for neural radiance field development. In *ACM SIGGRAPH*, 2023. 8

[53] Prune Truong, Marie-Julie Rakotosaona, Fabian Manhardt, and Federico Tombari. Sparf: Neural radiance fields from sparse and noisy poses. In *CVPR*, pages 4190–4200. IEEE, 2023. 2

[54] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. Instance normalization: The missing ingredient for fast stylization. *arXiv preprint arXiv:1607.08022*, 2016. 1

[55] Fangjinhua Wang, Silvano Galliani, Christoph Vogel, Pablo Speciale, and Marc Pollefeys. Patchmatchnet: Learned multi-view patchmatch stereo. In *CVPR*, pages 14194–14203. IEEE, 2021. 1, 3, 5

[56] Peng Wang, Lingjie Liu, Yuan Liu, Christian Theobalt, Taku Komura, and Wenping Wang. Neus: Learning neural implicit surfaces by volume rendering for multi-view reconstruction. *NeurIPS*, 2021. 3, 6

[57] Wenjing Wang, Jizheng Xu, Li Zhang, Yue Wang, and Jiaying Liu. Consistent video style transfer via compound regularization. In *AAAI*, pages 12233–12240, 2020. 2

[58] Qingshan Xu and Wenbing Tao. Learning inverse depth regression for multi-view stereo with correlation cost volume. In *AAAI*, pages 12508–12515, 2020. 5, 1

[59] Jiayu Yang, Wei Mao, Jose M. Alvarez, and Miaomiao Liu. Cost volume pyramid based depth inference for multi-view stereo. In *CVPR*, pages 4877–4886. IEEE, 2020. 3, 5

[60] Yao Yao, Zixin Luo, Shiwei Li, Tian Fang, and Long Quan. Mvsnet: Depth inference for unstructured multi-view stereo. In *ECCV*, pages 767–783. Springer, 2018. 5, 6

[61] Lior Yariv, Yoni Kasten, Dror Moran, Meirav Galun, Matan Atzmon, Basri Ronen, and Yaron Lipman. Multiview neural surface reconstruction by disentangling geometry and appearance. *NeurIPS*, 33, 2020. 3, 6, 7

[62] Lior Yariv, Jiatao Gu, Yoni Kasten, and Yaron Lipman. Volume rendering of neural implicit surfaces. In *NeurIPS*, 2021. 6, 3

[63] Alex Yu, Ruilong Li, Matthew Tancik, Hao Li, Ren Ng, and Angjoo Kanazawa. PlenOctrees for real-time rendering of neural radiance fields. In *ICCV*, pages 5752–5761. IEEE, 2021. 2, 3, 8

[64] Hang Zhang and Kristin Dana. Multi-style generative network for real-time transfer. In *ECCV*. Springer, 2018. 1

[65] Kai Zhang, Nick Kolkin, Sai Bi, Fujun Luan, Zexiang Xu, Eli Shechtman, and Noah Snavely. Arf: Artistic radiance fields. In *ECCV*, pages 717–733. Springer, 2022. 1, 2, 3, 6, 7, 4

MuVieCAST: Multi-View Consistent Artistic Style Transfer

Supplementary Material

A. Configuration details

In this section, we share details of the network architecture configurations referenced in Tables 1 and 2 of the paper. CasMVSNet [15] has 927K parameters, while PatchMatchNet [55] has 222K parameters. We divided CasMVSNet cost volume into 8 groups to calculate group-wise correlation [16, 58]. In CasMVSNet_UNet and PatchmatchNet_UNet, we computed content loss at layer *relu3_3* and used the Gram matrix-based style loss [13] at layers *relu1_2*, *relu2_2*, *relu3_3*, and *relu4_3* of the VGG16 loss network [51]. To achieve this, we trimmed the VGG16 until the *relu4_3* layer, resulting in 7.6M parameters. Our UNet [47] has 1.7M parameters. For CasMVSNet_AdaIN and PatchmatchNet_AdaIN, we computed content loss at layer *relu4_1* and style loss at layers *relu1_1*, *relu2_1*, *relu3_1*, and *relu4_1* of the VGG19 loss network. We trimmed the VGG19 [51] until the *relu4_1* layer, which resulted in 3.5M parameters. Our trainable VGG19-based AdaIN decoder [21] also has 3.5M parameters.

The training time for DTU scan65 with 49 images, a resolution of 640×480, a neighboring view window size of 3, and a batch size of 1 per GPU on dual RTX 2080 Ti was measured. For various network architectures, the training times for 10 epochs are as follows:

Network Architecture	**Training Time** (seconds)
CasMVSNet_UNet	174.44
CasMVSNet_AdaIN	174.52
PatchmatchNet_UNet	153.03
PatchmatchNet_AdaIN	155.00

B. UNet style translation

Although UNet-based style transfer methods [25] require style-specific pretraining, we observed that UNet's pretrained models can quickly adapt to new styles and images. By using weights learned from MS COCO object detection dataset [36], we can swiftly apply UNet to new style images for given inputs. Pretraining captures a general understanding of style features and textures that can be reused, significantly reducing the time and resources needed for training new models for each style. In this experiment, we used pretrained model of UNet trained with the "Starry Night". We pretrained the model with a new hand-drawn style image [21, 64] for 30 mins with a dual RTX2080 Ti. Figure 7 depicts results for style translation.

Figure 7. Results for style translation. First row: original input samples. Second row: results with pretrained style. Third row: finetuning with a new style image. Bottom row: the two style images used in the second and third rows, respectively.

Figure 8. Color adjustment as preprocessing. Top: original images. Bottom: after color adjustment

Figure 9. Color adjustment as postprocessing. Top: styled images. Bottom: after color adjustment

C. Color adjustment

Our framework includes a color adjustment feature that allows users to modify the color of images so that their color distribution is more similar to a specified style image. Similar approaches have been previously explored in the literature [33, 65]. Our framework supports both pre-processing and post-processing color adjustment. In the paper, we have only used color adjustment for novel view synthesis experiments, as both a pre-processing step and a post-processing step.

Our approach is inspired by the work of WCT [33], where we aim to match the RGB color means and covariances of the input images with those of a style image. This color mapping can be represented as an affine transformation

$$\tilde{c} = Mc + t, \tag{11}$$

where c represents the input color that is transformed into $\tilde{c}$. M is responsible for translating color vectors between the content and style domains, which is computed as $U_s \lambda_s^{\frac{1}{2}} V_s^T U_c \lambda_c^{-\frac{1}{2}} V_c^T$. U_c, λ_c, and V_c are computed using singular value decompositions of the colors of the content images, while U_s, λ_s, and V_s are computed using singular value decompositions of the colors of the style image. The translation vector t adjusts the mean and can be computed as $\mu_s - M\mu_c$, where μ_c is the mean color of the content images and μ_s is the mean color of the style image.

As the color mapping is affine, the color adjustment module can be used as either a pre-processing step or a post-processing step for downstream applications related to geometry. Figure 8 and Figure 9 demonstrate the effect of using color adjustment as preprocessing and postprocessing, respectively.

(a) Input + Input (b) Input + Stylized (c) Stylized + Input (d) Stylized + Stylized

Figure 10. Neural mesh reconstruction. Top row: Mesh editing effect. The reconstructed mesh demonstrates stripe-like geometric features. Bottom row: Mesh surface coloring (Geometry + Coloring). Meshes (a) and (b) have the same geometric properties but differ in their coloring. Specifically, (a) and (c) are colored using 64 original input images, while (b) and (d) are colored using the stylized images. The geometry of (a) and (b) is derived from the original inputs, while the geometry of (c) and (d) is learned from the stylized images.

D. Neural mesh reconstruction

By being independent of the 3D representation of the input, MuVieCAST has the potential to be used and extended as a tool for editing 3D scene representations. By using consistent 2D stylized images, we can generate 3D geometric textures. The top row of Figure 10 shows such an example, where the reconstructed mesh demonstrates stripe-like geometric features generated from the stylized images. These consistent features across multiple views are reflected in the reconstructed mesh as a geometric texture.

Our approach can also be applied to mesh coloring, similar to the experiments on point clouds. Recently, there are neural mesh rendering techniques [56, 61, 62] that disentangle geometry reconstructions from view-dependent appearance estimations. Leveraging this architectural advantage, we can utilize our method to color the meshes. We trained the geometry and rendering network of IDR [61] using both the original images and the stylized images. The bottom row of Figure 10 demonstrates our experiment results for mesh coloring.

E. Novel-view synthesis with real-world data

To further assess the capabilities and robustness of MuVieCAST, we conducted novel-view synthesis experiments using real-world data. Similar to our previous Nerf experiments, we derived camera poses from stylized images to show the robustness of our method. Figure 11, the left column shows parts of the original input and stylized images along with the style image. Our proposed stylization approach preserves the essential structure and details of the scene while introducing a unique artistic style, as shown by these stylized images. Moreover, the right column shows the novel view synthesis (NVS) results using the computed radiance fields and camera parameters from stylized images. The novel views exhibit fine details, accurate geometry, and coherent lighting with stylized rendering.

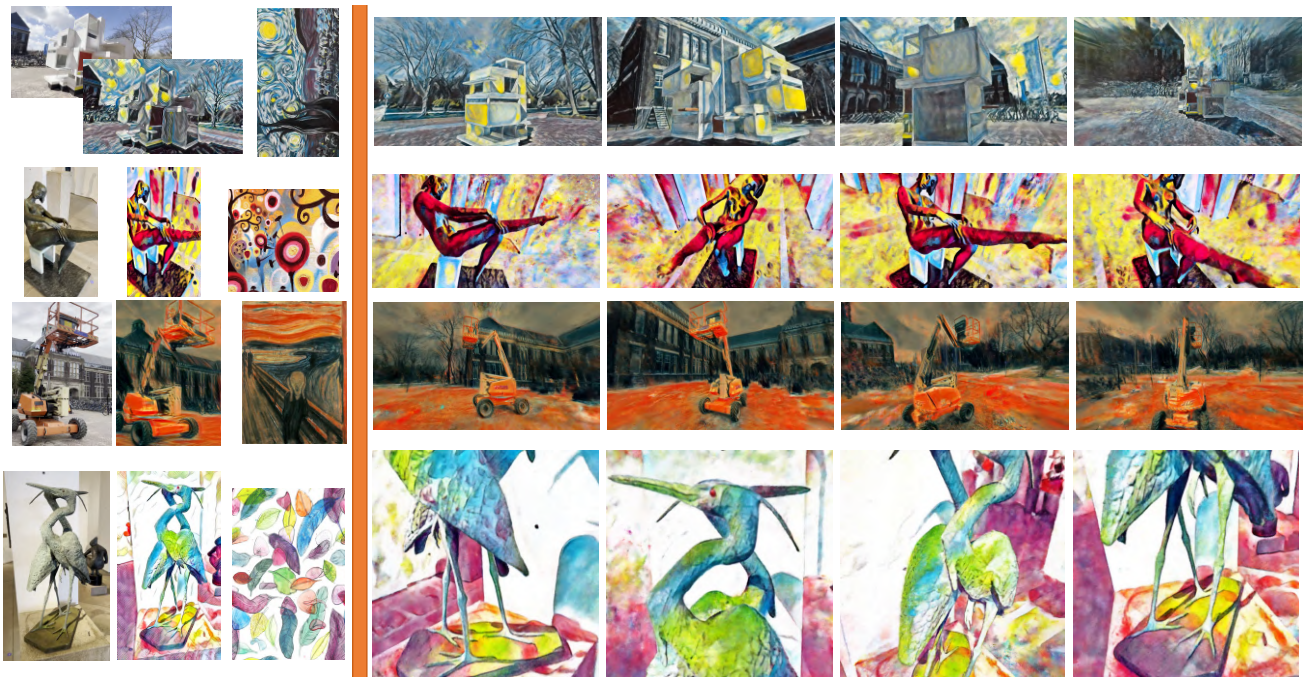


Figure 11. Novel view synthesis using four distinct styles, applied to real-world data. MuVieCAST was trained on each scene for less than 15 minutes, leveraging a dual RTX 2080 GPU. More results are demonstrated on our project page: muviecast.github.io

F. Loss terms

During the development process, we conducted over 500 automated TensorBoard experiments to define the losses.

Our empirical findings indicate that substituting SmoothL1 loss for MSE loss in the context of content loss and interchanging MSE loss with SmoothL1 loss for image geometry loss terms yields comparable outcomes. We extensively investigated different approaches for volume loss, including KL divergence, MSE, SmoothL1, and custom losses. Our findings demonstrated that SmoothL1 loss produced better learning curves. It is important to note that achieving convergence in volume loss without considering depth loss can be challenging. Regarding image geometry loss terms, both MSE and SmoothL1 losses performed similarly.

In addition to the aforementioned loss terms, we also experimented with some other losses, such as nearest-neighbor-based losses [31, 65] and total variations [25], similar to previous works. However, our experiments with nearest-neighbor-based feature matching loss did not result in proper improvement, and it proved to be computationally expensive. Our experiments with total-variation (TV) loss did not generate visually and geometrically better results either. Despite this, our framework still supports these loss terms (nearest-neighbor feature matching NNFM [65] and total-variation [25]) for users who want to conduct further experiments.

In Sec. H, we conduct an ablation study involving loss terms within a sparse view scenario.

G. User survey results

We conducted a user survey involving 40 participants across 5 scenes and 5 styles, following the trajectory set by the author of ARF [65]. Here, we aim to delve into the reasoning behind the outcomes of our user study. We specifically showcase paired renders per scene: one from ARF and one from our model. It is important to note that our approach involves single-step optimization for multi-view consistent style transfer, and our radiance fields utilize estimated camera poses from styled images.

Here we aim to interpret the factors contributing to the outcomes of the user survey. Figure 12 provides a closer examination of the comparison between our results and those of ARF. In Figure 12 (a), our results were preferred by 90% of participants (36 out of 40) because of their richer texture compared to ARF. Figure 12 (b) illustrates that 77.5% of votes (31 out of 40) leaned toward our interpretation as our style transfer maintained geometric and semantic clarity, unlike ARF, which introduced noisy artistic textures, complicating scene understanding. Moving to Figure 12 (c), our approach produced smoother and more visually appealing coloring, preserving semantic and geometric understanding, whereas ARF resulted in patchy high-frequency coloring. This example received 92.5% of the votes (37 out of

40). Figure 12 (d) showed a more balanced preference, with a slight inclination (55% of the votes) toward our results. Upon inspection, our method excelled in recovering distant objects, while ARF rendered sharper depictions of ground and foreground elements. Lastly, in Figure 12 (e), 75% of participants (30 out of 40) opted for ARF results. Our analysis indicated that our results appeared more blurred compared to ARF, highlighting sharper radiance fields in the latter.

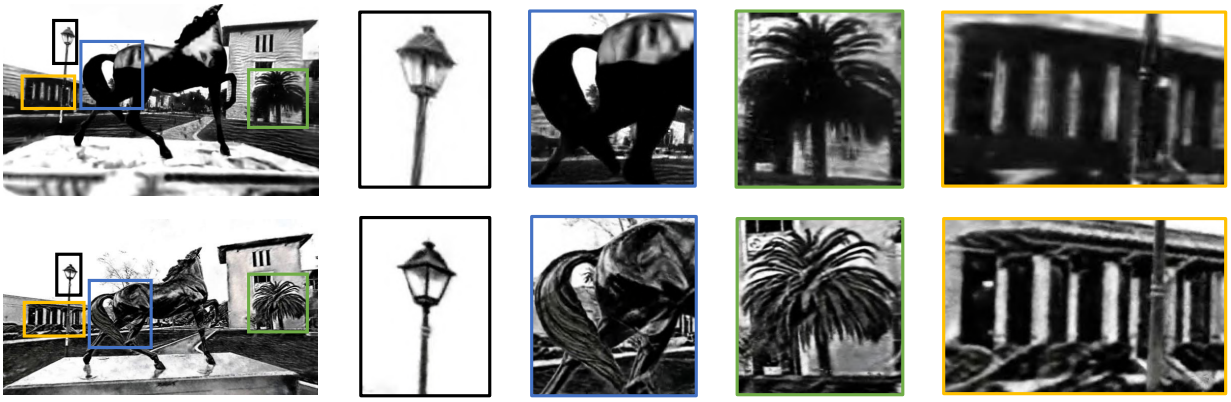
H. Ablation study for sparse views

In this part, we aim to demonstrate the effectiveness and robustness of our pipeline with sparse views by conducting ablation experiments on loss terms. To illustrate this, we utilize the 4 views of the scan65 dataset from DTU [1] as an example and experiment with two different architectures, PatchmathNet_UNet and PatchmathNet_AdaIN, as shown in Figure 13 and Figure 14, respectively.

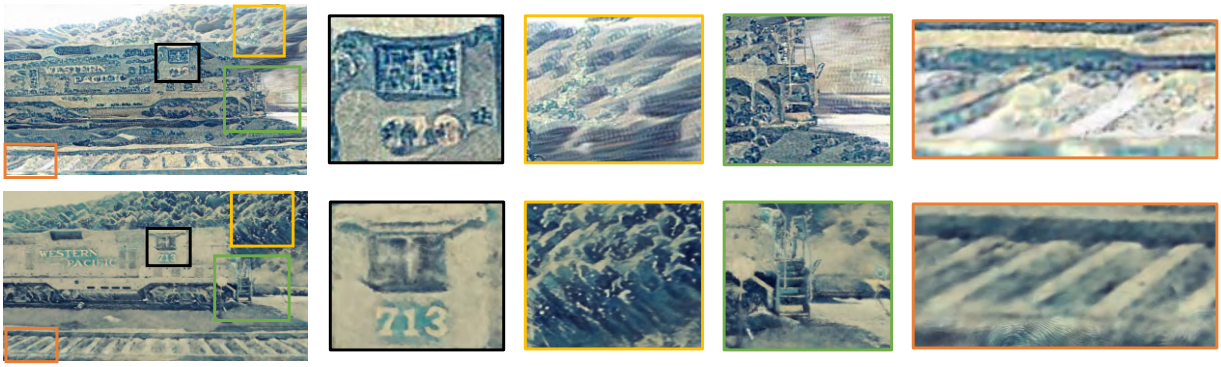
To examine the impact of each loss term, we train the network separately with each loss term and demonstrate the corresponding results. These results indicate that the UNet-based architecture recovers the original image features better with the help of content, image-geometry, and 3D geometry losses. On the other hand, the AdaIN-based architecture can recover the original image features using content and 3D geometry losses. However, the results of 3D geometry loss with AdaIN architecture tend to be blurrier than those of UNet. Additionally, we observed that the image-geometry loss pays attention to photometric edges, while the style loss tends to enhance artistic features in both cases.

From the experiments, we can conclude that the 3D geometry losses is effective in capturing the overall geometry of the scene. However, guiding the network with only this loss term may reduce the artistic features of the images. Note that in our ablation studies, we combined volume and depth loss into a unified 3D geometry loss. Specifically, we noticed that while volume loss struggles to converge independently, its integration alongside depth loss enhances pose estimation, particularly in scenarios involving low-resolution images. This integration offered promising results for improving the network's performance in handling varied image qualities and dimensions.

Meanwhile, the image geometry loss can be attentive to the photometric edges, and the content loss can preserve the original content of the images while reducing the artistic features. Moreover, the style loss can be useful in retaining artistic features. These findings motivated us to use a weighted sum of these loss terms to conduct geometry-aware style transfer.



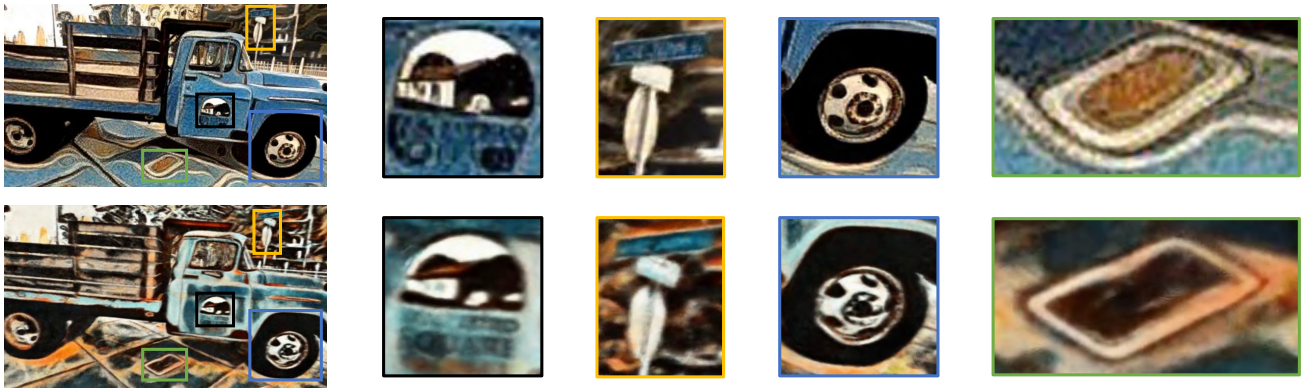
(a) 36 out of 40 participants chose our output (second row).



(b) 31 out of 40 participants chose our output (second row).

(c) 37 out of 40 participants chose our output (second row).

(d) 22 out of 40 participants chose our output (second row).



(e) 30 out of 40 participants chose ARF output (first row).

Figure 12. Comparison between ARF [65] and our results. The first row in each subfigure shows the ARF outcome, while our result is given in the second row.

Figure 13. Ablation study results with PatchMatchNet_UNet by separately training the network with different loss terms. The first row shows the RGB input, and the second row shows the results obtained from the pretrained network. The subsequent rows demonstrate the results of the network trained with each loss term separately, starting with the content loss in the third row, followed by image geometry, 3D geometry loss, and style loss in the fourth, fifth, and sixth rows, respectively. The last row demonstrates the results obtained by incorporating all the loss terms.

Figure 14. Ablation study results with PatchMatchNet_AdaIN by separately training the network with different loss terms. The first row shows the RGB input, and the second row shows the results obtained from the pretrained network. The subsequent rows demonstrate the results of the network trained with each loss term separately, starting with the content loss in the third row, followed by image geometry, 3D geometry loss, and style loss in the fourth, fifth, and sixth rows, respectively. The last row demonstrates the results obtained by incorporating all the loss terms.