

# Locally Stylized Neural Radiance Fields

Hong-Wing Pang<sup>1</sup>, Binh-Son Hua<sup>2,3</sup>, and Sai-Kit Yeung<sup>1</sup>

<sup>1</sup>Hong Kong University of Science and Technology <sup>2</sup>Trinity College Dublin <sup>3</sup>VinAI Research, Vietnam

## Abstract

In recent years, there has been increasing interest in applying stylization on 3D scenes from a reference style image, in particular onto neural radiance fields (NeRF). While performing stylization directly on NeRF guarantees appearance consistency over arbitrary novel views, it is a challenging problem to guide the transfer of patterns from the style image onto different parts of the NeRF scene. In this work, we propose a stylization framework for NeRF based on local style transfer. In particular, we use a hash-grid encoding to learn the embedding of the appearance and geometry components, and show that the mapping defined by the hash table allows us to control the stylization to a certain extent. Stylization is then achieved by optimizing the appearance branch while keeping the geometry branch fixed. To support local style transfer, we propose a new loss function that utilizes a segmentation network and bipartite matching to establish region correspondences between the style image and the content images obtained from volume rendering. Our experiments show that our method yields plausible stylization results with novel view synthesis while having flexible controllability via manipulating and customizing the region correspondences.

## 1. Introduction

Stylizing a visual world is an increasingly popular and demanding task in games, movies, or extended reality applications. Imagine that one can navigate in an artistic virtual world that resembles the painting styles by different renowned artists. This problem is generally known as 3D style transfer.

Traditionally, 3D style transfer can be achieved via post-processing. For example, in the well-known traditional computer graphics pipeline, it typically involves a programmable shading stage to post-process the appearance of the rendered geometry or screen images. Neural radiance field [20] is a recent advance in 3D deep learning that aims to represent a 3D scene implicitly by using a neural network trained with multi-view images and differentiable volume render-

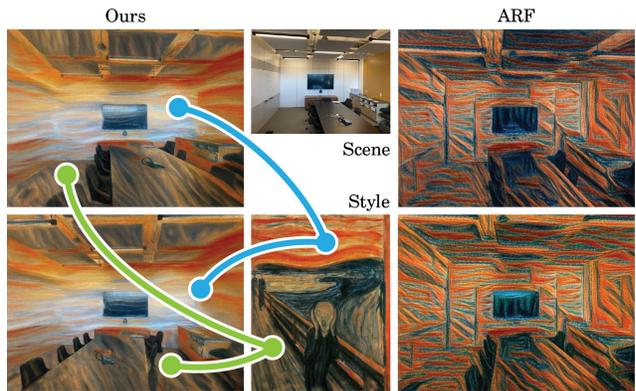


Figure 1. We propose a stylization method for NeRF, aware of correspondences between different style patterns and local regions within the rendered image.

ing. Since this pioneer work, significant milestones have been made to greatly improve the performance of neural radiance fields in practice, including improved spatial representation [17], training convergence [21], explicit geometry representation [31]. It is therefore promising to revisit the 3D style transfer problem by stylizing a 3D scene implicitly represented by a neural radiance field.

In this work, our goal is to transfer the appearance from a reference style image to a neural radiance field and keep the style transfer consistent across novel views rendered from the radiance field. We are inspired by works in image [3, 7] and video style transfer methods that have received great attention since the introduction of modern neural networks. While previous works [22, 32] have proposed adapting the style transfer problem to neural radiance fields as well, the stylization results lack diversity and controllability.

To address these limitations, we devise a new style transfer method that considers the local transfer between a reference style image and the radiance field rendering. In particular, our method treats style transfer as a post-processing step after the original geometry and appearance of the neural radiance field is learned and therefore aim to perform style transfer while keeping the geometry implicitly represented

by the neural radiance field unchanged. We propose a new backbone suitable for stylizing neural radiance fields that has a dual-branch architecture to learn the density and the appearance field, respectively. We devise a hash-grid encoding scheme with an extended hash function to support storing multiple styles in a single parametric embedding of the appearance field. Given the same reference style image, it is possible to diversify the stylization results by customizing the hash function used for positional encoding.

We further propose a new segmentation-based stylization loss which subdivides both the 3D scene and style image into subregions; different regions in the scene are matched with a region in the style image and stylized accordingly. The matching between scene and style image regions are formulated as a bipartite matching problem and solved by the Hungarian algorithm. We show that our method automatically generates plausible stylization results with high-quality geometry and appearance, reflecting a diverse range of local styles found in the style image. In addition, the generated matching can also be manually edited by the user, making the stylization process controllable.

To summarize, our contributions are:

- A reference-guided style transfer method for neural radiance fields. Our architecture for learning a neural radiance field is a dual-branch network that aims at optimizing the appearance while keeping the geometry fixed during stylization;
- An extended hash-encoding scheme for stylization. We provide an analysis of hash-encoding and its influence on the style transfer on radiance fields and multiple style support;
- A new style loss that adopts optimal assignment from bipartite matching on segmented regions between the reference style image and the radiance field rendering for style transfer.

## 2. Related Work

**Image and video style transfers.** Image style transfer can be dated back to early work for image analogies [4], which uses a pair of images as the training data to learn a filter which can be subsequently applied on new images to simulate analogous filtered results. Deep image analogies [16] performs visual attribute transfers by using deep features from a neural network to derive dense correspondences between the input and the style image, which is only applicable when these images are semantically similar, e.g., images of human faces. Neural style transfer [3] instead uses deep features to build content and style constraints and optimizes a random noise image to output a new image with matched content and style statistics. The optimization can be replaced by a feed-forward prediction by training a neural network on

pairs of original and stylized images with improved results using instance normalization [29] and perceptual loss [10]. Recent methods aim at supporting arbitrary style images at test time without the need to retrain the stylization network by leveraging adaptive instance normalization [7], patch-based transfer [27, 26], feature transform [15] as well as multiple style support [23].

As an extension of image style transfer methods, video style transfer methods further deal with the temporal consistency problem for transferring styles across video frames [9, 28]. Different from video style transfer methods, we focus on stylization of 3D scenes, where view-consistent stylization is required to achieve high-quality novel view synthesis.

**3D style transfers.** Stylizing a 3D scene can be performed explicitly on point clouds and mesh representations [11, 6, 5]. However, this approach is error-prone due to imperfect geometry and texture rendering. In contrast, NeRF [20] allows representing a 3D scene by learning a neural network using multiple-view images and differentiable volume rendering. As a result, NeRF can smoothly and consistently interpolate its rendering at different angles, which is an ideal application for novel view synthesis. The last few years witness tremendous research interests in this area with significant progress in improving NeRF in various aspects including its spatial representation [17], training convergence [21], explicit geometry representation [31], to name a few. Stylizing a 3D scene by using a NeRF representation is therefore gaining attention recently.

A simple approach to stylize a NeRF model is to regress a NeRF and constrain its rendering to 2D image stylization results with a content loss and a style loss. SNeRF [22] follows this approach and demonstrate consistent stylization across novel views. Their style loss is a global constraint and ignores any spatial correspondences between the rendering and the style image. ARF [32] defines a nearest-neighbor feature matching loss but since their correspondence assignment is fixed, their results do not support diverse stylization results. Our work differs in that we propose a spatial matching between the style image and NeRF rendering so that the dynamically assigned correspondences can be used to guide the stylization with diverse results.

To speed up stylization process, the optimization on NeRF can be replaced by a feed-forward prediction. Chiang et al. [1] used a hypernetwork to predict the weights of a NeRF MLP given an arbitrary style. Our work utilizes the hash encoding [21] for style transfer, which significantly speeds up the optimization process.

Other attempts have been made to improve consistency in the 2D and 3D space during style transfer such as using a point cloud as an intermediate representation for 2D-3D feature transfer [6], or using mutual learning by distilling spatial consistency and stylized rendering between a NeRF and a 2D style transfer network [8].



Figure 2. Stylization results from modifying the hash function coefficients. This yields similar and consistent global styles while having local diversity.

### 3. Our Method

The stylization of NeRF models are typically completed in two stages - the *reconstruction stage*, where a base NeRF model is trained with respect to the MSE reconstruction loss, similar to regular NeRF training; followed by the *stylization stage*, where the parameters of the NeRF model is fine tuned against style-transfer specific losses. Our method follows this paradigm, but we reconsider several design choices discussed below.

#### 3.1. Dual-branch NeRF model

In this work, we propose a dual-branch architecture for our neural radiance fields. Our model is illustrated in Figure 3. Our dual-branch architecture consists of a geometry branch to encode the density component of the neural radiance field, and an appearance branch to encode the color component, respectively. This design allows us to subsequently only optimize the appearance branch for the stylization task while keeping the geometry unchanged.

We utilize contributions from Instant-NGP [21] to represent each branch, which uses a parametric embedding to replace the original fixed positional embedding used in vanilla NeRF. Particularly, the bounded scene volume is subdivided into a large number of voxels each corresponding to a set of learnable parameters, which is stored in a fixed-size hash map. Thus, the learned embedding provides a rich, descriptive set of features representing the scene geometry and appearance. The depth and no. of parameters of the subsequent MLP networks can be greatly reduced, improving the training time by a few orders of magnitude. The proposed model differs from the vanilla architecture proposed in Instant-NGP [21] in the following ways. First, we train two separate sets of hash-grid encodings,  $E_C$  and  $E_D$  to represent the appearance and geometry, respectively. Second, we discard the view direction input. This is a common practice found in prior NeRF stylization work, considering that style-transferred scenes often do not require view-dependent effects during rendering.

During the reconstruction stage, we train our model with multi-view images and the MSE loss, similar to [21]. During

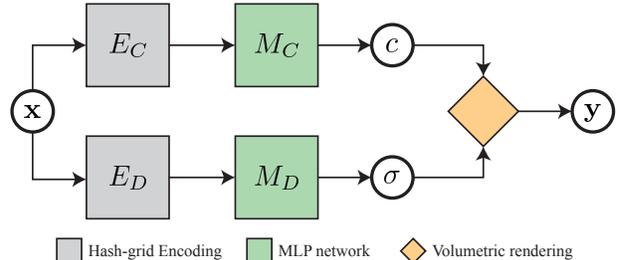


Figure 3. Our dual-branch architecture for neural radiance fields. We optimize only the appearance branch for the stylization task.

the stylization stage, we fix the weights of components  $E_D$  and  $M_D$ , such that the density value  $\sigma$  corresponding to each input point  $x$  remains fixed. We then optimize the appearance component  $E_C$  and  $M_C$  with a novel style loss, to be discussed in Section 3.3.

**Effect of hash-grid encoding in stylization.** In our formulation, the large number of parameters in the learned hash-grid encoding  $E_C$  describes the rendered appearance of the radiance field. Specifically, the encoding corresponding to the voxel at  $(x, y, z)$  is given by the geometric hash function  $H$  described in [21]:

$$H(x, y, z) = (h_1x \oplus h_2y \oplus h_3z) \bmod N_H, \quad (1)$$

where  $h_1, h_2, h_3$  are large prime numbers and  $N_H$  is the total size of the hash-grid. By picking different values for  $h_1, h_2, h_3$ , we demonstrate in Figure 2 that different patterns can be generated after the stylization training stage. Compared with existing stylization methods where a single stylization result is generated per style image / scene, this gives a wider level of variation.

In addition, our architecture can be used to stylize multiple styles simultaneously, as opposed to prior methods that focus on transferring a single style. This is done by adding a fourth term to the hash function:

$$H(x, y, z) = (h_1x \oplus h_2y \oplus h_3z \oplus h_4s_i) \bmod N_H, \quad (2)$$

where  $s_i$  is a discrete style index. In other words, we store encodings for multiple styles inside the same hash-grid, allowing stylization of more than one style at inference time. We demonstrate the results of training multiple styles simultaneously in Section 3 of the supplementary material.

### 3.2. Preliminaries on style loss

Given a pair of rendered output image  $\mathbf{y}$  and style image  $\mathbf{s}$ , stylization losses typically operate on high-level features  $\mathbf{f}_y = \mathcal{F}(\mathbf{y}), \mathbf{f}_s = \mathcal{F}(\mathbf{s})$  extracted with a pre-trained and fixed feature extraction network  $\mathcal{F}$ ; for example, StylizedNeRF [8] follows the training setup of AdaIN [7], which uses a style loss that matches statistics (i.e. mean and standard deviation) between  $\mathbf{f}_y$  and  $\mathbf{f}_s$ , where  $\mathcal{F}$  is a pretrained VGG-16 [24] network.

However, the matching of global statistics means that the style may not be properly transferred across every local region in  $I$ . The nearest-neighbor feature matching (NNFM) loss introduced in [32] uses the following formulation instead:

$$\mathcal{L}_{\text{NNFM}}(\mathbf{y}, \mathbf{s}) = \frac{1}{N_F} \sum_{f_i \in \mathbf{f}_y} \min_{f_j \in \mathbf{f}_s} d(f_i, f_j) \quad (3)$$

where every individual feature vector  $f_i$  in  $\mathbf{f}_y$  is paired with the closest style feature vector  $f_j$  in  $\mathbf{f}_s$ , in terms of the cosine angle distance  $d$ . The NNFM loss is then defined as the mean of distances over all  $N_F$  pairs of vectors, where  $N_F$  is the no. of vectors in  $\mathbf{f}_y$ .

However, matching up nearest neighbor features between  $\mathbf{f}_y$  and  $\mathbf{f}_s$  may not always lead to good results. Figure 4 shows two stylization results by ARF onto the same scene. We see that in both cases, recurrent patterns are generated over the entire scene. In (a), different parts of the scene (e.g. floor, walls) gets the same pattern, even though there are multiple distinct patterns in the original style image to choose from. In (b), recurrent patterns are generated on the “blank” walls, even when equally “blank” regions are present in the style image. We suggest that applying nearest neighbor search over the entire image does not lead to the best choice in stylization, especially in cases where vectors in  $\mathbf{f}_y$  are dissimilar to all vectors in  $\mathbf{f}_s$ .

### 3.3. Style loss with region correspondences

Our key observation is that, instead of only comparing between nearest neighbors on the feature level, we can divide up  $\mathbf{y}$  and  $\mathbf{s}$  into coarse regions  $\{\mathbf{y}_i\}$  and  $\{\mathbf{s}_j\}$ , where features in region  $\mathbf{y}_i$  is matched towards features within a corresponding region  $\mathbf{s}_j$ . The idea of matching feature statistics between regions has previously been explored in [18], which segments content and style images into regions by semantic segmentation and pair-up the regions by semantic labels. It is nontrivial, however, to extend this method to NeRF stylization, for the following reasons. First, in order



Figure 4. Stylization results using nearest-neighbor feature matching (NNFM) [32] are not always satisfactory, for example, recurrent patterns occur in the stylization results and not all regional styles are transferred.

to stylize an entire scene, the scene will need to be dissected into regions in a manner that is consistent over any arbitrary novel view. Second, the method proposed in [18] is designed for photorealistic style images; in contrast, semantic segmentation is unlikely to give meaningful results on artistic style images. Third, the lack of semantic segmentation labels also means that there is no intuitive way to pair-up the regions.

To this end, we propose a training pipeline which segments *both*  $\mathbf{y}$  and  $\mathbf{s}$ . A matching is derived automatically from the segmented regions, which is used to influence the calculation of the style loss. An overview of this pipeline is described in Figure 5.

**Defining regions for  $\mathbf{y}$  and  $\mathbf{s}$ .** The first step to our pipeline is to subdivide  $\mathbf{y}$  and  $\mathbf{s}$  into  $C$  *scene regions* and  $S$  *style regions*. We use the unsupervised segmentation method *Segment Anything* [13] to segment  $\mathbf{s}$  into  $S$  style regions with diverse local patterns and styles.

Computing the segmentation of  $\mathbf{y}$  is nontrivial, as we require a consistent segmentation that assigns regions consistently over any arbitrary novel views from the same scene. To solve this problem, we first segment the set of training images  $\{\hat{\mathbf{y}}\}$  into  $C$  regions, using a variant of the *unsupervised* image segmentation method proposed by Kim et. al. [12]; which allows simultaneous segmentation of multiple unseen images without prior semantic knowledge.

We then introduce an additional MLP network  $M_K$  in our NeRF backbone as shown in Figure 5, which produces a  $C$ -dimensional vector output:

$$\tilde{k} = M_K(E_c(x)). \quad (4)$$

The list of  $\tilde{k}$ s computed over a single ray are integrated together with density values from  $M_D$  using the volumetric equation, and subsequently passed through the softmax function. The result is a probability vector  $k \in \mathbb{R}^C$  describing the probabilities of the pixel belonging to each of the  $C$  scene regions. During the reconstruction stage,  $M_K$  is trained simultaneously with other components with respect

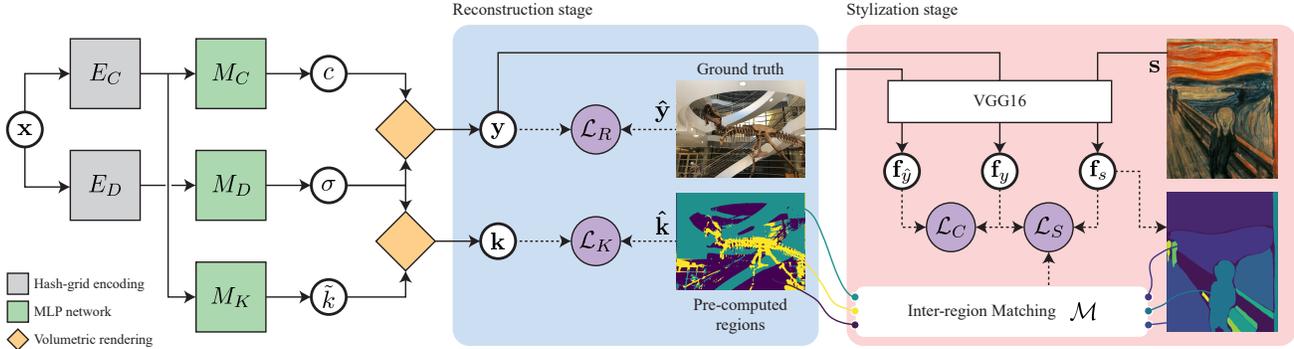


Figure 5. Our two-stage pipeline for NeRF stylization. During the reconstruction stage, the entire network is trained to simultaneously render a novel view  $\mathbf{y}$  of the target scene and generate a segmentation map  $\mathbf{k}$ . During the stylization stage, the appearance components ( $E_C$  and  $M_C$ ) are fine-tuned with our novel style loss  $\mathcal{L}_S$  which considers the matching between regions in  $\mathbf{y}$  and  $\mathbf{s}$ .

to the cross-entropy loss  $\mathcal{L}_K$ . For each vector  $k_i$  computed from a pixel, we have

$$\mathcal{L}_K(k_i, \hat{k}_i) = \sum_{i=1}^C -\hat{k}_i k_i \log k_i, \quad (5)$$

where  $\hat{k}_i$  is the corresponding scene region (in one-hot vector form) from the segmentation map  $\hat{\mathbf{k}}$ , pre-computed from the ground truth training image  $\hat{\mathbf{y}}$ . After the reconstruction stage is completed, the NeRF backbone can simultaneously render a novel view  $\mathbf{y}$  and generate its corresponding segmentation map  $\mathbf{k}$ . Note that our stylization pipeline is agnostic to the segmentation method used to segment  $\mathbf{s}$  and  $\{\hat{\mathbf{y}}\}$ . The choice of  $C$  and  $S$  are automatically determined during unsupervised segmentation; for the results shown on this paper, we have  $C \approx 5$  and  $S \approx 10$ .

In Section 1 of the supplementary material, we describe the implementation of segmentation in greater detail, as well as provide an ablation experiment to demonstrate the effect of using higher values of  $C$  through segmenting the scene and style image into finer regions.

**Style loss.** Following the procedure in Section 3.3, we have  $C$  scene regions  $\{\mathbf{y}_i\}$  segmented from  $\mathbf{y}$ ; as well as  $S$  style regions  $\{\mathbf{s}_j\}$  segmented from  $\mathbf{s}$ . To avoid multiple regions being mapped to a single local pattern in  $\mathbf{s}$ , we formulate this as a bipartite matching problem where no two scene regions are mapped to the same style region.

To do so, we construct a cost matrix  $\mathbf{W} \in \mathbb{R}^{C \times S}$ , where each individual entry  $W_{ij}$  represents the affinity between regions  $\mathbf{y}_i$  and  $\mathbf{s}_j$ . The cost is determined by the *feature distance* and the *patch distance*, defined as follows. The feature distance is defined as the cosine angle distance between the means of features in  $\mathbf{y}_i$  and  $\mathbf{s}_j$ . The patch distance is defined as the Euclidean distance between the centroids (i.e. arithmetic mean position of all pixels constituting the region) of  $\mathbf{y}_i$  and  $\mathbf{s}_j$ ; each centroid position is normalized

as a value in  $[0, 1]$  given the image dimensions. The patch distance dictates that scene regions are more likely to be paired with style regions that are roughly located in the same relative position. Using the previous example in Figure 4 (a), the scene regions corresponding to the ceiling and floor are more likely to be mapped to patches in the sky and floor in the style image.

After computing  $\mathbf{W}$ , an optimal injective mapping  $\mathcal{M} : [1, C] \mapsto [1, S]$  can be obtained by applying the Hungarian algorithm, such that the following mapping cost  $\sum_{i \in C} W_{i, \mathcal{M}(i)}$  is minimized. Given a mapping  $\mathcal{M}$ , our updated style loss is formulated as follows:

$$\mathcal{L}_S(\mathbf{y}, \mathbf{s}, \mathcal{M}) = \frac{1}{N_F} \sum_{f_i \in \mathbf{f}_y} \min_{f_j \in \mathbf{f}_s} d(f_i, f_j) \quad (6)$$

such that if  $f_i \in \mathbf{y}_i$ , then  $\mathbf{s}_j$  is the corresponding style region of  $\mathbf{y}_i$  in  $\mathcal{M}$ .

**Custom matching.** While the above method provides an automatic matching between  $\{\mathbf{y}_i\}$  and  $\{\mathbf{s}_j\}$ , the exact pairing can be further modified to produce a wide variety of style transfer results for different scenarios. We demonstrate this capability in Section 4.3.

### 3.4. Training and implementation details

**Reconstruction stage.** The model is first trained for 20,000 iterations with the objective function:

$$\mathcal{L}_R(\mathbf{y}, \hat{\mathbf{y}}) + \lambda_{CE} \sum_{i=1}^{N_T} \mathcal{L}_K(k_i, \hat{k}_i), \quad (7)$$

where we mix the reconstruction loss  $\mathcal{L}_R$  with the cross-entropy loss  $\mathcal{L}_K$  multiplied by a fixed constant  $\lambda_{CE} = 0.01$ .  $N_T$  is the number of pixels sampled during each training iteration. It is further trained for 2,000 iterations after applying color transformation from  $\mathbf{s}$  to  $\mathbf{y}$ , as proposed in [32].

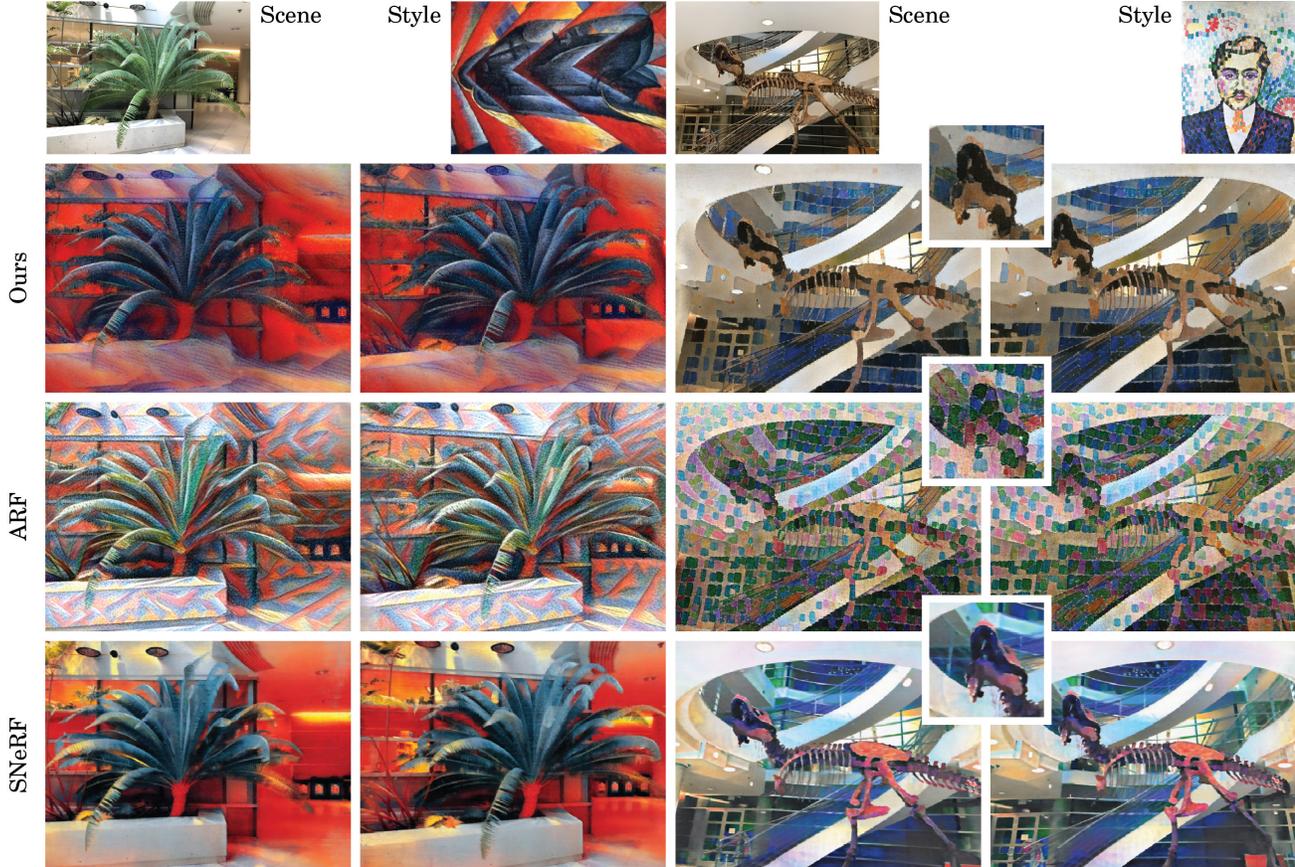


Figure 6. Qualitative comparison results with SNeRF [22] and ARF [32] on the LLFF dataset [19].

**Stylization stage.** The pairing  $\mathcal{M}$  between scene regions and style regions is first computed using the heuristic functions described above. We fix  $\mathcal{M}$  through the entire stylization stage to avoid instability during training. We then train the model with the following objective function:

$$\lambda_C \mathcal{L}_C(\mathbf{y}, \hat{\mathbf{y}}) + \lambda_S \mathcal{L}_S(\mathbf{y}, \mathbf{s}, \mathcal{M}), \quad (8)$$

where  $\mathcal{L}_C$  is the content loss  $\|\mathbf{f}_y - \mathcal{F}(\hat{\mathbf{y}})\|_2^2$ ;  $\lambda_C$  is fixed to 0.001; and  $\mathcal{L}_S$  is the style loss with the pairing  $\mathcal{M}$  enforced.

In addition, we follow the experimental settings in [32], where the feature extractor  $\mathcal{F}$  collects all the post-ReLU features from the `conv3` block of VGG-16 and concatenates them together; as well as utilizing *deferred backpropagation* to allow optimizing the loss function over entire images under limited GPU memory.

## 4. Experiments

### 4.1. Qualitative comparisons

We demonstrate qualitative comparisons in Figure 6 with ARF [32] and SNeRF [22], another relatively recent baseline for NeRF stylization. As SNeRF do not release their code

implementation, we perform comparison on the same scenes and style images used in Figure 5 in their paper, using scenes in the LLFF [19] dataset.

In general, the results produced by SNeRF do not have significant change in terms of the pattern of the image; only the color has been altered to match the style image. SNeRF directly computes the MSE loss between  $\mathbf{f}_y, \mathbf{f}_s$  as their style loss, meaning that the style similarity is being compared on an image-wide level. Thus, it is hard to predict which style region will be transferred to which scene region. ARF is successful in transferring local patterns from the style image; however, in either scenes large parts of the scene has been blanketed with the same pattern. In the second scene, the dinosaur head is difficult to identify after applying the mosaic pattern.

Our results strike a balance between the two baselines. In the second scene of Figure 6, the background area is mapped to the dark blue suit in the style image; whereas the top area is mapped to the top of the style image where the mosaic pattern is not as obvious. The matching is both consistent and predictable, resulting in a diverse combination of patterns over the entire scene.

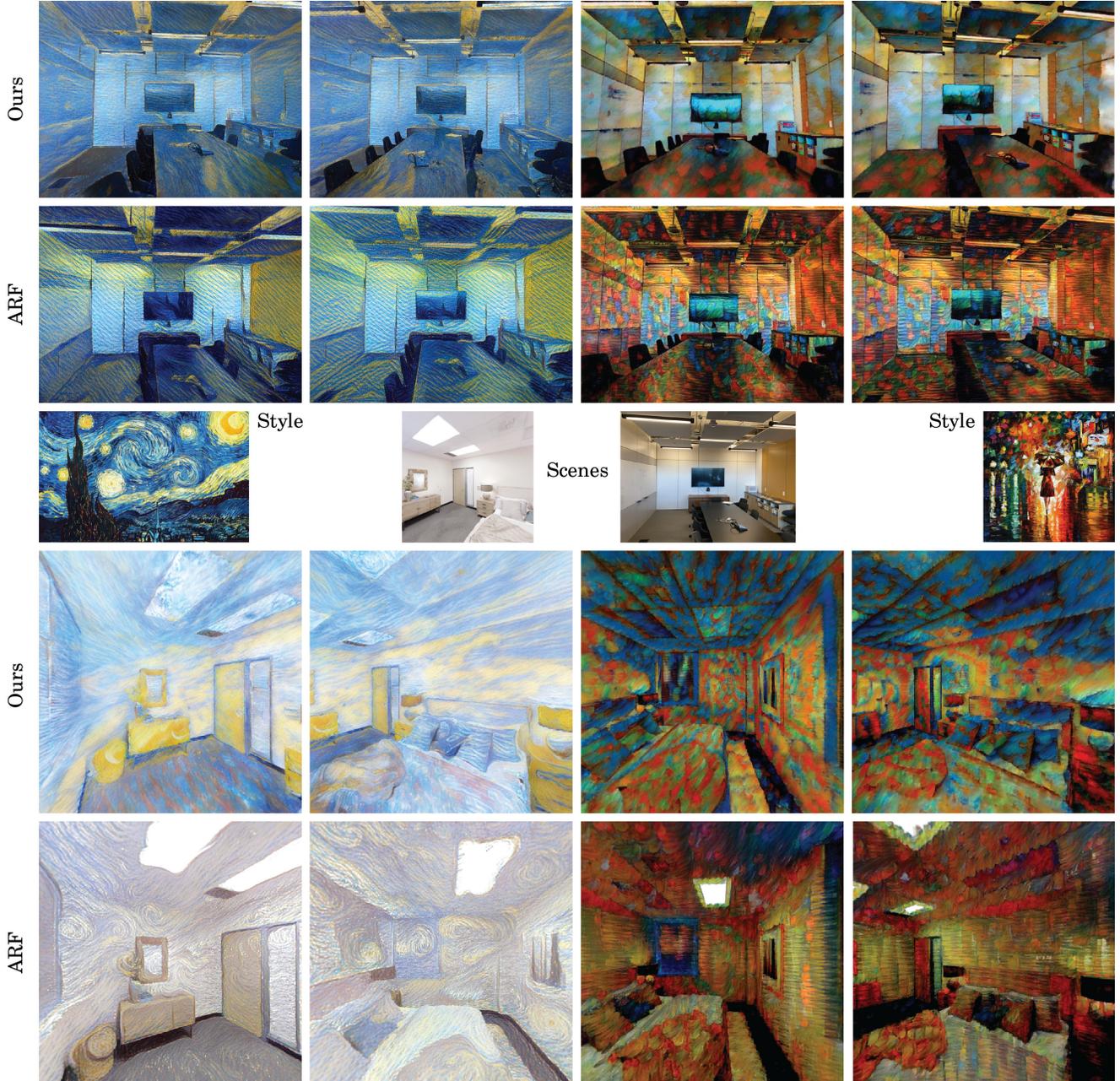


Figure 7. Further qualitative comparison results with ARF [32] on an LLFF scene [19] (top rows) and on a Replica scene [25] (last rows).

We further compare our stylization results with ARF on more examples in Figure 7. We use another scene in the LLFF dataset, followed by another scene in the Replica [25] dataset. Our method is capable of drawing diverse patterns and styles within the given style image, applying to consistent regions across the content image. For example, in the style image to the right, we can better distinguish different parts of the scene by the use of different patterns and colors. We provide further qualitative comparisons of even more scenes and style images in the supplementary material.

## 4.2. Ablation experiments

It is mentioned previously in Section 3.4 that the pairing  $\mathcal{M}$  is determined at the start of the stylization stage, and influences the computation of style loss  $\mathcal{L}_S$ . Without this pairing, the style loss falls back to the NNFM loss which looks for the nearest neighbor feature across the entire style image. In Figure 8, we evaluate the effect of computing  $\mathcal{M}$  by comparing the results with using the vanilla NNFM loss on our NGP-based rendering backbone.



Figure 8. Ablation comparison with (a) and without (b) pairing between scene and style regions.

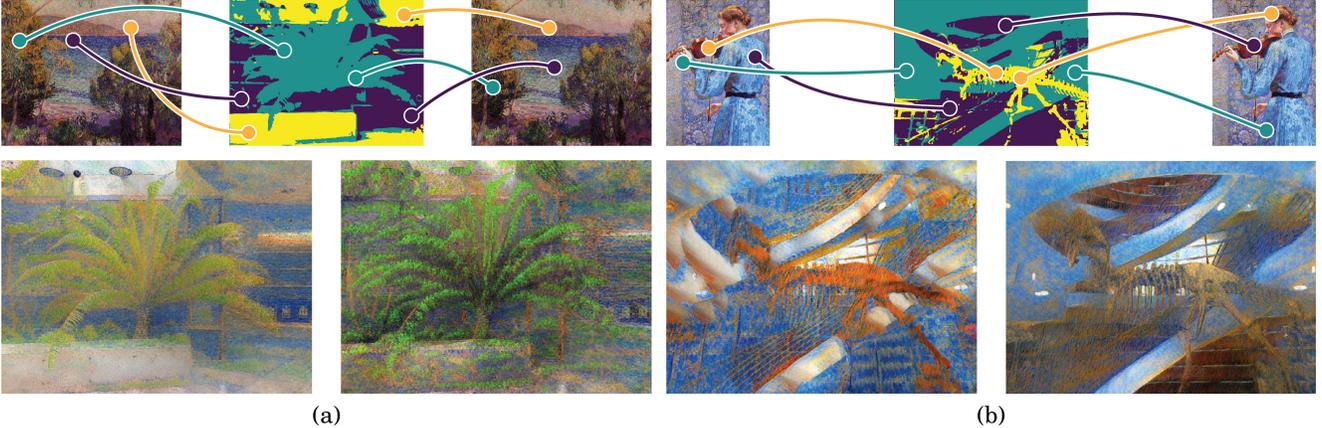


Figure 9. Stylization results via custom pairings between scene and style regions. Best view with zoom.

### 4.3. Effect of different pairings

In Figure 9, we demonstrate the capability to customize the stylization result via modifying the pairing  $\mathcal{M}$ , given the same scene and style image. In both cases, the scene is segmented into 3 classes, which are manually mapped to two different set of style regions. In (a), the tree in the foreground undergoes a significant change in appearance when it is mapped to different plants in the style image; whereas in (b), the scene regions also changes appearance based on the different pairing.

## 5. Further discussion

**Extension to unbounded scenes.** While Instant-NGP provides a form of 3D scene representation that is efficient to render, it requires the scene to be contained within a bounded volume, which will be mapped to the 3D hash grid for parameter storage. This makes it non-trivial to extend our method towards unbounded scenery, e.g. Tanks & Temples [14]. It would be helpful to extend the hash-grid encoding towards unbounded scenes by reparametrizing the coordinates within a bounded volume, such as the inverse-distance approach used in NeRF++ [33].

**Optimizing density.** In contrast to stylization methods where the density component of the backbone is fixed, recent methods such as SNeRF [22] and NerfArt [30]) has suggested

that optimizing both the density and appearance components will lead to improvement in stylization results. It is debatable whether optimizing the density field is really necessary or not. We opt for leaving density untouched.

## 6. Conclusion

In this work we present a novel framework for stylizing a neural radiance field. We optimize the appearance branch in a dual-branch radiance field represented by a parametric embedding learned with an extended hash function. We devise a new style loss based on region correspondences between the style image and the content images from rendering the radiance field. Our method generates diverse stylization results, where the stylization can be controlled by the extended hash function and the region correspondences.

Our method is not without limitations. First, our method is not (yet) suitable for interactive use even though the hash-encoding scheme reduces the required time for optimizing the neural radiance fields to minutes. Second, our method support multiple styles but these styles have to be known in advance of the stylization. Supporting arbitrary style transfer at test time would be an interesting future work.

**Acknowledgment.** This paper was partially supported by an internal grant from HKUST (R9429) and the HKUST-WeBank Joint Lab.

## Supplementary material

In this supplemental document, we provide additional details and experiment results for our proposed method. In particular, we provide an in-depth description of the segmentation procedure used in our stylization procedure; a user study; and more qualitative results on the LLFF and the Replica dataset.

### A. Segmentation procedure

In this section, we describe in greater detail the segmentation procedures used for creating scene regions and style regions. As mentioned in the main paper, our stylization method can be performed irrespective of the segmentation method used.

#### A.1. Scene regions

Given a set of training images  $\{\hat{\mathbf{y}}\}$ , we want to generate a set of segmentation maps  $\{\hat{\mathbf{k}}\}$ , where every pixel is classified into a finite set of  $C$  scene regions. The segmentation process should be unsupervised (i.e.  $C$  is not set to a fixed value, and is determined during segmentation), and can be applied to any arbitrary set of scenes.

To satisfy these requirements, we base our implementation on the segmentation procedure proposed by Kim et. al. [12]. In their method, a single image is passed into a CNN model producing a *response map*  $\mathbf{r} \in \mathbb{R}^{H \times W \times Q}$ , where  $Q$  is the upper bound of no. of scene regions. A segmentation map  $\mathbf{c} \in \mathbb{R}^{H \times W}$  can be obtained from  $\mathbf{r}$  by taking the argmax function.

This network is trained with a combination of the *similarity loss*  $\mathcal{L}_{sim}$  and *continuity loss*  $\mathcal{L}_{con}$ . The similarity loss is defined as the sum of cross-entropies between response vector  $r_n$  and the target vector  $c_n$ :

$$\mathcal{L}_{sim}(\mathbf{r}) = - \sum_n \sum_{i=1}^Q c_{n,i} \log r_{n,i}, \quad (9)$$

where  $c_n$  is the one-hot vector in  $\mathbf{c}$  corresponding to  $r_n$ . The continuity loss is defined as the sum of L1 distances between horizontally and vertically adjacent features in  $\mathbf{r}$ :

$$\mathcal{L}_{con}(\mathbf{r}) = \sum_{i=1}^{W-1} \sum_{j=1}^{H-1} \|r_{i+1,j} - r_{i,j}\|_1 + \|r_{i,j+1} - r_{i,j}\|_1. \quad (10)$$

It can be seen that  $\mathcal{L}_{sim}$  encourages similar features in  $\mathbf{r}$  to be grouped together and form a single cluster;  $\mathcal{L}_{con}$  ensures spatial continuity of clusters and prevents the segmentation from being too fragmented. In general, the unique number of classes in  $\mathbf{c}$  is initially high (i.e. close to  $Q$ ), and gradually decreases over time as more and more feature vectors in  $\mathbf{r}$  are clustered into the same region.

To extend this method to segmentation of multiple images simultaneously, we train the segmentation network by sampling a batch of  $B$  images during each iteration, instead of a single image. The loss values for each individual response maps  $\{\mathbf{r}_1, \dots, \mathbf{r}_B\}$  are computed and summed together. After the network is trained, we can run segmentation over all of  $\{\hat{\mathbf{y}}\}$ , obtain the set of  $C$  remaining active classes, and re-index the segmentation maps from 1 to  $C$ .

#### A.2. Style regions

Given a style image  $\mathbf{s}$ , we want to segment it into a set of  $S$  style regions  $\{\mathbf{s}_j\}$ , once again without explicit supervision. Unlike section A.1, we only need to apply segmentation on a single image. We use the robust *Segment-Anything* method [13] as it has good performance outside real-life images, which is the case for style images in artistic style transfer. We use the official pretrained weights based on ViT-H [2].

Directly applying the method results in a set of regions  $\{\mathbf{s}_j\}$  which may overlap with each other. To fix this issue, we first sort the regions in decreasing order of size, and run the procedure in Algorithm 1. Here,  $\{\mathbf{s}_1, \dots, \mathbf{s}_N\}$  is the

---

#### Algorithm 1 Filtering overlapping style regions

---

```

 $\mathbf{m} \leftarrow 0$ 
 $\mathbf{k} \leftarrow -1$ 
 $i \leftarrow 0$ 
for  $\mathbf{s}_j$  in  $\{\mathbf{s}_1, \dots, \mathbf{s}_N\}$  do
  if  $\sum \mathbf{m}[\mathbf{s}_j] / |\mathbf{s}_j| \geq \lambda_t$  and  $|\mathbf{s}_j| / \|\mathbf{s}\| \geq \lambda_m$  then
     $\mathbf{m}[\mathbf{s}_j] \leftarrow 1$ 
     $\mathbf{k}[\mathbf{s}_j] \leftarrow i$ 
     $i \leftarrow i + 1$ 
  end if
end for

```

---

list of regions from largest to smallest;  $\mathbf{m} \in \mathbb{R}^{H \times W}$  is a binary map that keep tracks if a pixel has been assigned to a style region; and  $\mathbf{k} \in \mathbb{R}^{H \times W}$  is the segmentation map. The notation  $\mathbf{m}[\mathbf{s}_j]$  and  $\mathbf{k}[\mathbf{s}_j]$  represents the subset of pixels in  $\mathbf{m}$  and  $\mathbf{k}$  belonging to region  $\mathbf{s}_j$ .  $\lambda_t$  determines if the current  $\mathbf{s}_j$  is overlapping with previous regions;  $\lambda_m$  ensures that regions too small are not considered. We set  $\lambda_t$  as 0.05 and  $\lambda_m$  as 0.004 (i.e. 0.4% of total image area).

After the procedure, each pixel of  $\mathbf{k}$  should be given an integer value from  $[-1, S - 1]$ , where 0 to  $S - 1$  indicates the  $S$  style regions. An index of -1 means that the pixel is not assigned to any region, and is not considered during stylization.

Last but not least,  $\mathbf{k}$  is downscaled by nearest neighbor interpolation to match the dimensions of  $\mathbf{f}_s$ , the VGG features extracted from  $\mathbf{s}$ .

### A.3. Fine vs. coarse regions

The values of  $C$  and  $S$  determines the level of fineness during the segmentation of scene and style regions. We provide an ablation experiment to experiment on the effect of using larger values of  $C$  and  $S$  on the stylization result.

The value of  $C$  can be indirectly controlled by modifying the number of iterations of training the unsupervised segmentation network; in general, by using a smaller number of iterations, the network output will contain a larger number of classes as it has not fully converged. For this experiment, we segment the style image  $s$  with the same procedure as scene regions, creating different segmentation maps with different values of  $S$ .

Figure 11 demonstrates segmentation results under three sets of scene regions and style regions. In the first example, the change from  $C = 8, S = 14$  to  $C = 19, S = 25$  results in a more varied stylization result; the walls and ceiling are dissected into smaller scene regions which are given different styles. However, "over-segmentation" of the style image  $s$  will only result in smaller style regions with similar patterns and colors, i.e. it will not create significant changes towards the stylization result, as demonstrated by the results of increasing  $C, S$  to  $C = 41, S = 47$ . A similar trend can be observed for the second example as well.

### A.4. Injective and surjective mapping

When computing the mapping  $\mathcal{M}$ , our current method assumes that  $C \leq S$ , i.e.  $\mathcal{M}$  is injective. Under this assumption, every scene region is matched to a unique style region, which prevents a single local style from dominating the stylization.

However, the computation of our style loss  $\mathcal{L}_S$  can take in any arbitrary mapping function. For example, in the case where  $C < S$ , we can produce a *surjective mapping* where every style region has to be used for stylization.

To demonstrate this point, we provide an additional ablation experiment comparing between injective mapping and surjective mapping in Fig. 10. Under our default injective setting, we have  $C = 4, S = 15$ . By reducing the number of training iterations during the segmentation of scene regions, we can increase the number of  $C$  from 4 to 26. To generate a surjective mapping with 26 scene regions, we run our current Hungarian algorithm matching to obtain a bijective mapping for 15 scene regions; then run the algorithm again to match the remaining 9 scene regions.

In this example, we observe that the increase in scene regions means a more diverse stylization result; for example, the chairs are no longer constrained to be stylized in the same style as the ceiling. Nevertheless, many scene regions similar in nature are stylized similarly. This arises from the fact that multiple scene regions are now assigned to the same style region, or style regions that have similar patterns and appearances.

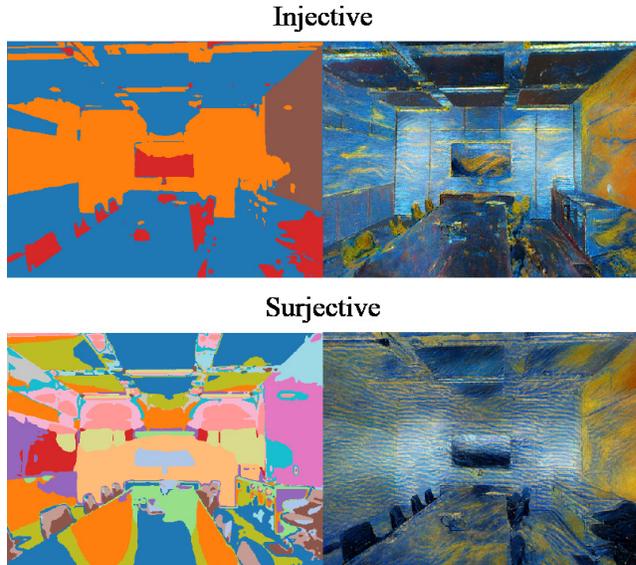


Figure 10. Injective vs. surjective mapping comparison.

One future direction to extended our current method is to improve the automatic matching algorithm such that any general mapping  $\mathcal{M}$  can be considered as a candidate.

### B. User study

We conduct a user study to verify the performance of our proposed method compared with ARF [32]. The study consists of 20 questions presented in random order; each question consists of two images rendered from a scene stylized by our method; as well as the two ARF-generated images with the same scene and camera pose. The order of the two choices are randomized. In addition, the corresponding ground truth training image and the style images are also shown. The user is asked to select the choice that preserves the content of the ground truth image, and simultaneously appears similar to that of the style image. Out of the 20 questions, 12 of them correspond to images rendered from the `trex`, `room` and `fern` scenes from the LLFF [19] dataset; and 8 of them correspond to images rendered from the `office3` and `f1r1_apartment3` scenes in the Replica dataset [25].

We collected a total of 23 replies and the percentage of picking each method is summarized in Figure 12. The study shows that on average our method is picked at a higher percentage than ARF on both the LLFF and Replica datasets.

### C. Further qualitative results

We show in this section the results of simultaneously training the stylization of multiple styles within a single hash grid. Figure 13 shows the `fern` scene from LLFF stylized in four distinct styles. Figure 14 compares the difference

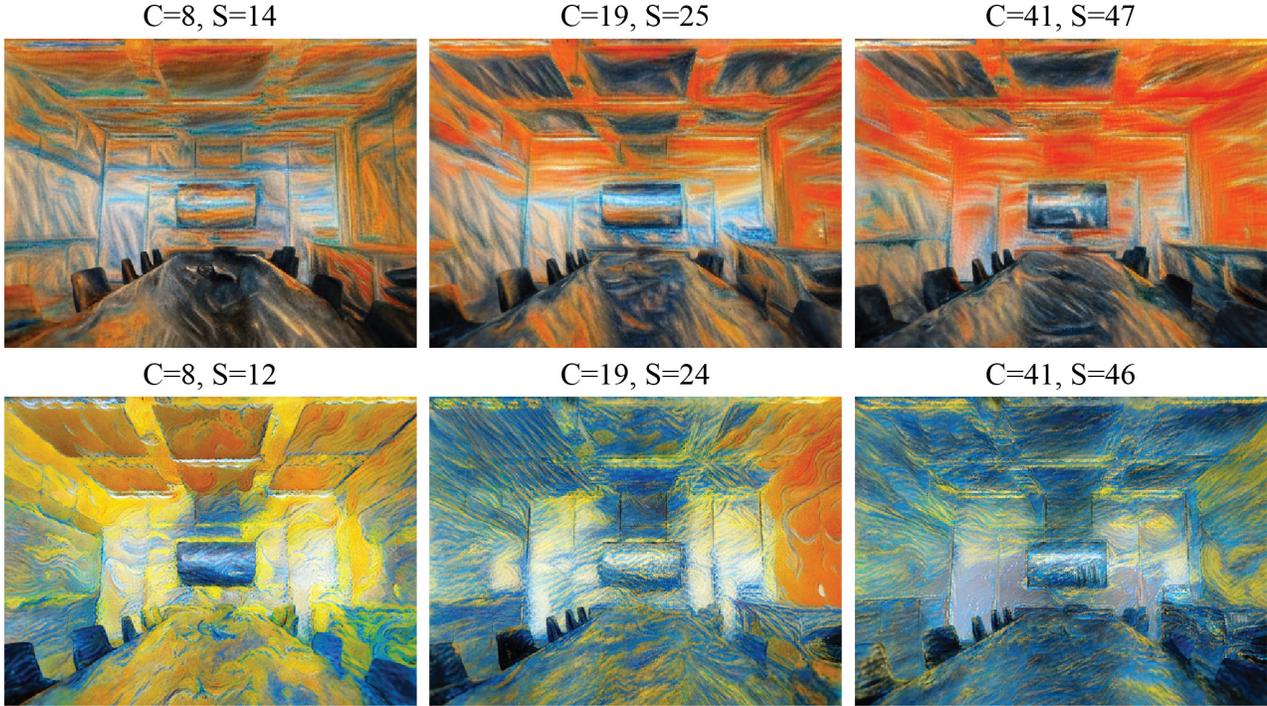


Figure 11. Stylization results by using segmentation maps of different degree of fineness.

	Ours	ARF
LLFF	72.3%	27.7%
Replica	64.1%	35.9%

Figure 12. User study results.

between with and without regional matching.

We provide further qualitative results for the LLFF dataset in Figure 15, and for the Replica dataset in Figure 16. Both figures illustrate that our method is less likely to transfer similar, repetitive patterns to the NeRF scene. This is especially the case in low-frequency regions, e.g. concrete walls and bare surfaces.

Our algorithm for matching content-style regions work by assuming that the regions can be paired up in a meaningful sense. However, even in cases where there is little to no correlation between regions from the NeRF scene and style image, our method is able to transfer local patterns on the scene.

Finally, we provide two further examples of modifying the pairing between content and style regions in Figure 17. We demonstrate that our method can achieve diverse and customizable stylization results via adjusting the pairing.

## References

- [1] Pei-Ze Chiang, Meng-Shiun Tsai, Hung-Yu Tseng, Wei-Sheng Lai, and Wei-Chen Chiu. Stylizing 3d scene via implicit representation and hypernetwork. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, 2022. 2
- [2] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *9th International Conference on Learning Representations*, 2021. 9
- [3] Leon A Gatys, Alexander S Ecker, and Matthias Bethge. Image style transfer using convolutional neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2414–2423, 2016. 1, 2
- [4] Aaron Hertzmann, Charles E. Jacobs, Nuria Oliver, Brian Curless, and David H. Salesin. Image analogies. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques*, page 327–340, 2001. 2
- [5] Lukas Höllein, Justin Johnson, and Matthias Nießner. Stylemesh: Style transfer for indoor 3d scene reconstructions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6198–6208, 2022. 2
- [6] Hsin-Ping Huang, Hung-Yu Tseng, Saurabh Saini, Maneesh Singh, and Ming-Hsuan Yang. Learning to stylize novel views. In *ICCV*, 2021. 2



Figure 13. Multiple styles rendered from the same model.

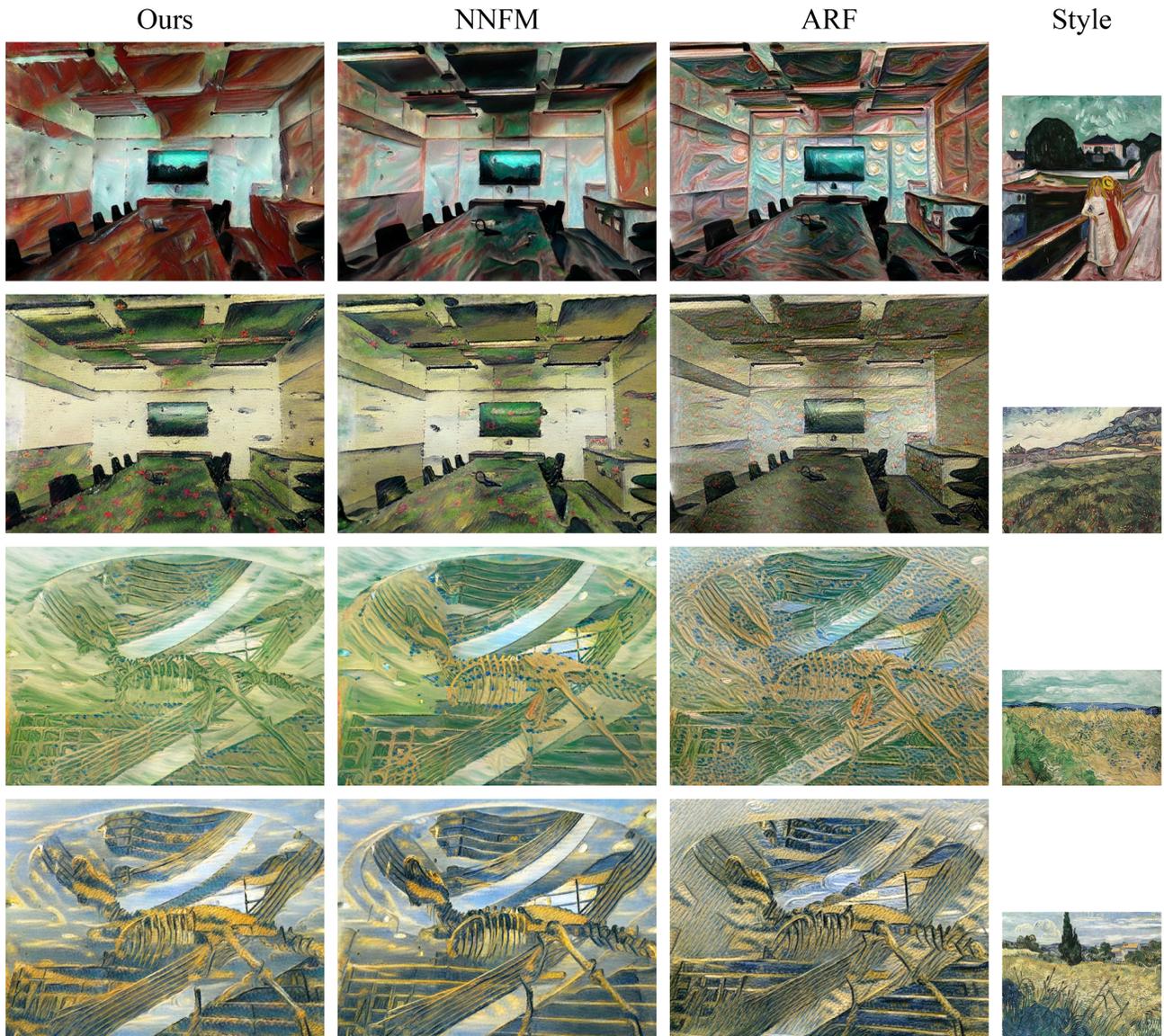


Figure 14. Comparison with and without regional matching. Left column uses regional matching; middle column do not use regional matching, i.e. reduces to the NNFM loss which looks for closest feature vector across entire image; Right column shows result from ARF.

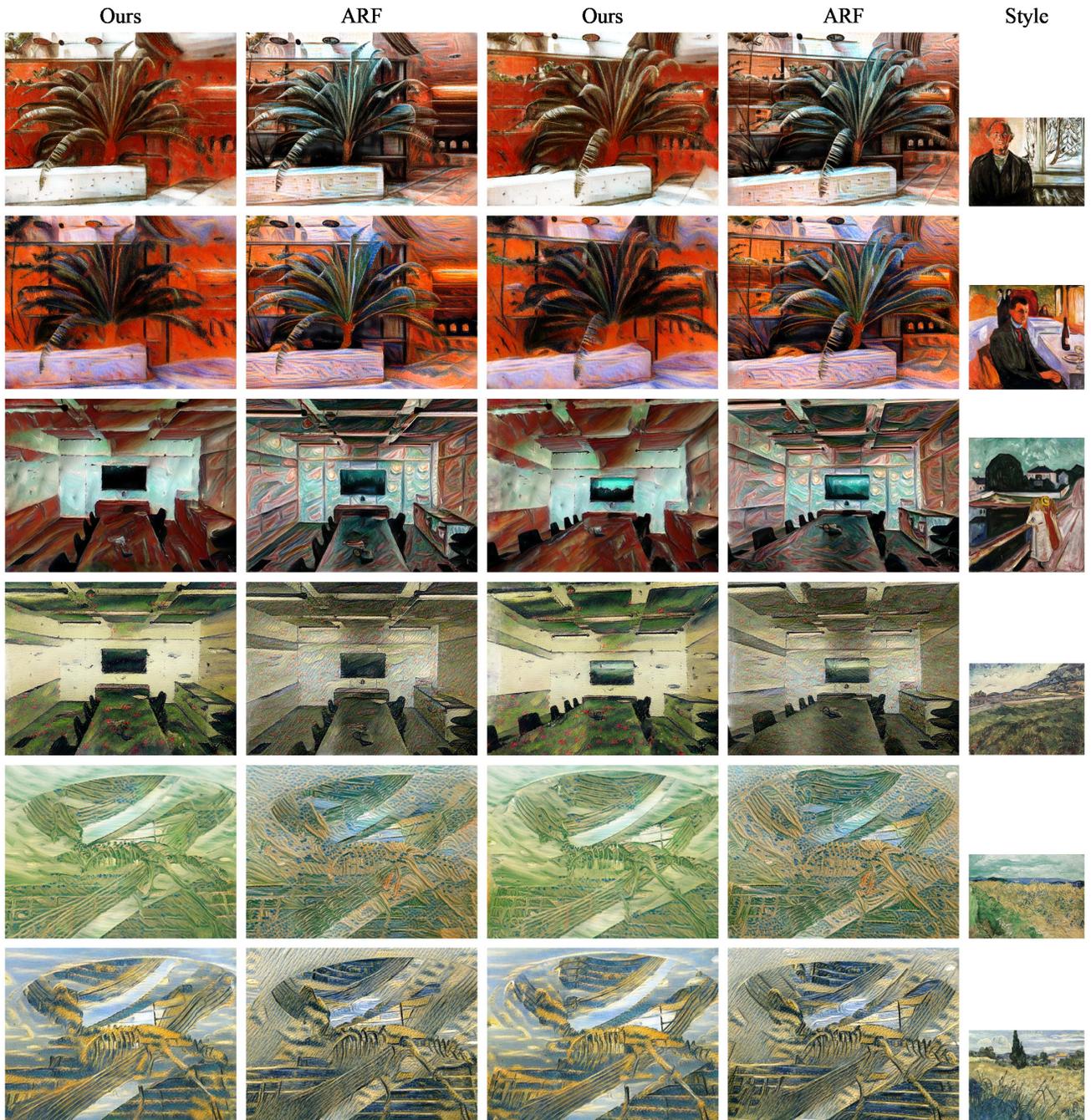


Figure 15. Further qualitative comparison on LLFF dataset.



Figure 16. Further qualitative comparison on Replica dataset.

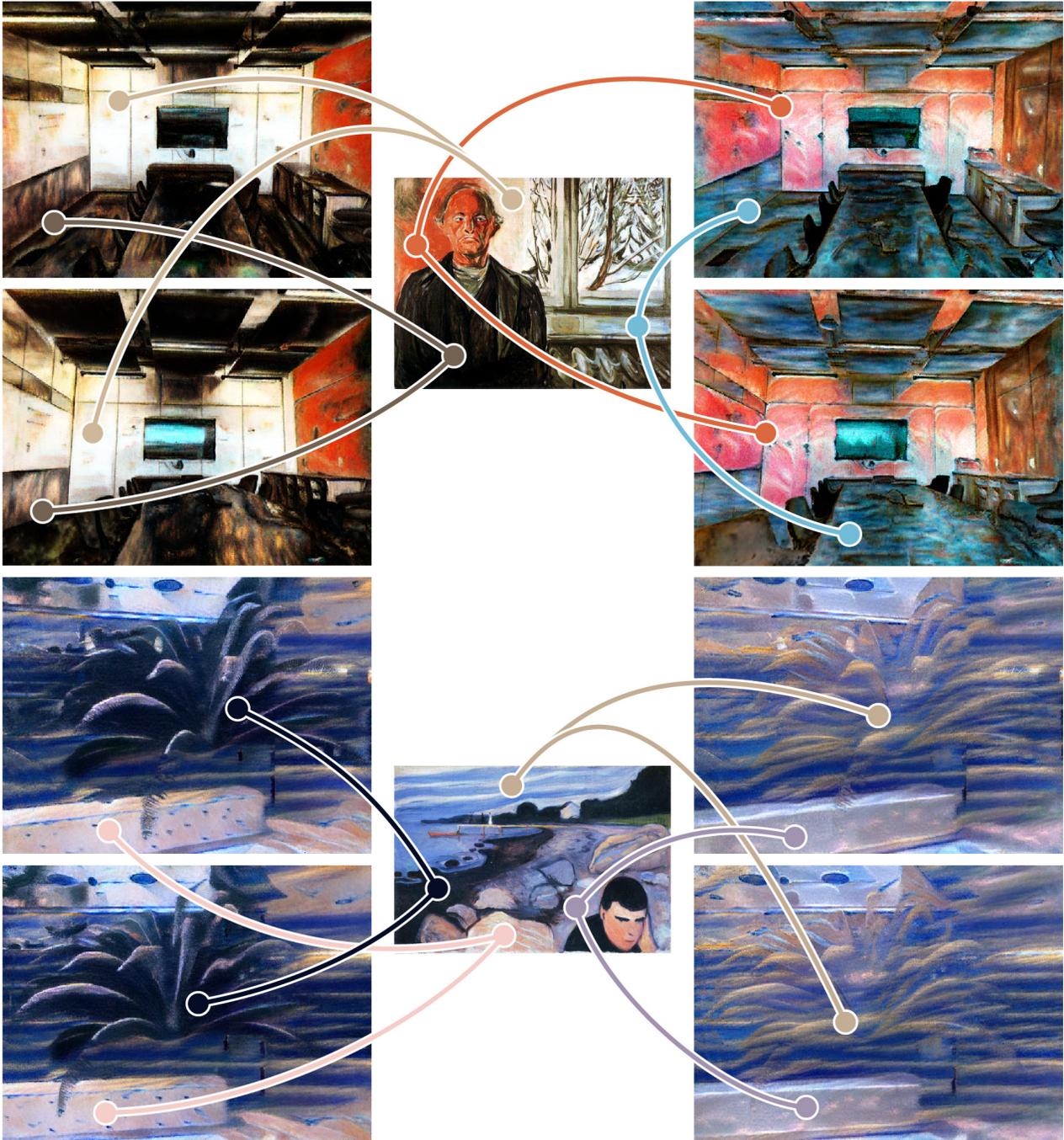


Figure 17. Effect of modifying the pairing between content and style regions. In each example, two content regions have been mapped to two different style regions in the style image (middle column), leading to two completely different stylization results (left and right columns).

- [7] Xun Huang and Serge Belongie. Arbitrary style transfer in real-time with adaptive instance normalization. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2017. 1, 2, 4
- [8] 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*, 2022. 2, 4
- [9] Ondřej Jamriška, Šárka Sochorová, Ondřej Texler, Michal Lukáč, Jakub Fišer, Jingwan Lu, Eli Shechtman, and Daniel Šykora. Stylizing video by example. *ACM Transactions on Graphics*, 38(4), 2019. 2
- [10] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *ECCV*, 2016. 2
- [11] Hiroharu Kato, Yoshitaka Ushiku, and Tatsuya Harada. Neural 3d mesh renderer. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 2
- [12] Wonjik Kim, Asako Kanezaki, and Masayuki Tanaka. Unsupervised learning of image segmentation based on differentiable feature clustering. *IEEE Transaction on Image Processing*, 2020. 4, 9
- [13] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C. Berg, Wan-Yen Lo, Piotr Dollár, and Ross Girshick. Segment anything. *arXiv:2304.02643*, 2023. 4, 9
- [14] Arno Knapitsch, Jaesik Park, Qian-Yi Zhou, and Vladlen Koltun. Tanks and temples: Benchmarking large-scale scene reconstruction. *ACM Transactions on Graphics*, 36(4), 2017. 8
- [15] Yijun Li, Chen Fang, Jimei Yang, Zhaowen Wang, Xin Lu, and Ming-Hsuan Yang. Universal style transfer via feature transforms. *Advances in neural information processing systems*, 30, 2017. 2
- [16] Jing Liao, Yuan Yao, Lu Yuan, Gang Hua, and Sing Bing Kang. Visual attribute transfer through deep image analogy. *ACM Trans. Graph.*, 36(4), July 2017. 2
- [17] Lingjie Liu, Jiatao Gu, Kyaw Zaw Lin, Tat-Seng Chua, and Christian Theobalt. Neural sparse voxel fields. *NeurIPS*, 2020. 1, 2
- [18] Fujun Luan, Sylvain Paris, Eli Shechtman, and Kavita Bala. Deep photo style transfer. *arXiv preprint arXiv:1703.07511*, 2017. 4
- [19] Ben Mildenhall, Pratul P. Srinivasan, Rodrigo Ortiz-Cayon, Nima Khademi Kalantari, Ravi Ramamoorthi, Ren Ng, and Abhishek Kar. Local light field fusion: Practical view synthesis with prescriptive sampling guidelines. *ACM Transactions on Graphics (TOG)*, 2019. 6, 7, 10
- [20] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. In *ECCV*, 2020. 1, 2
- [21] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM Trans. Graph.*, 2022. 1, 2, 3
- [22] Thu Nguyen-Phuoc, Feng Liu, and Lei Xiao. Snerf: Stylized neural implicit representations for 3d scenes. *ACM Trans. Graph.*, 2022. 1, 2, 6, 8
- [23] Artsiom Sanakoyeu, Dmytro Kotovenko, Sabine Lang, and Björn Ommer. A style-aware content loss for real-time hd style transfer. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 698–714, 10 2018. 2
- [24] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015. 4
- [25] Julian Straub, Thomas Whelan, Lingni Ma, Yufan Chen, Erik Wijmans, Simon Green, Jakob J. Engel, Raul Mur-Artal, Carl Ren, Shobhit Verma, Anton Clarkson, Mingfei Yan, Brian Budge, Yajie Yan, Xiaqing Pan, June Yon, Yuyang Zou, Kimberly Leon, Nigel Carter, Jesus Briales, Tyler Gillingham, Elias Mueggler, Luis Pesqueira, Manolis Savva, Dhruv Batra, Hauke M. Strasdat, Renzo De Nardi, Michael Goesele, Steven Lovegrove, and Richard Newcombe. The Replica dataset: A digital replica of indoor spaces. *arXiv preprint arXiv:1906.05797*, 2019. 7, 10
- [26] Ondřej Texler, Jakub Fišer, Michal Lukáč, Jingwan Lu, Eli Shechtman, and Daniel Šykora. Enhancing neural style transfer using patch-based synthesis. In *Proceedings of the 8th ACM/EG Expressive Symposium*, 2019. 2
- [27] Ondřej Texler, David Futschik, Jakub Fišer, Michal Lukáč, Jingwan Lu, Eli Shechtman, and Daniel Šykora. Arbitrary style transfer using neurally-guided patch-based synthesis. 2020. 2
- [28] Ondřej Texler, David Futschik, Michal Kučera, Ondřej Jamriška, Šárka Sochorová, Menglei Chai, Sergey Tulyakov, and Daniel Šykora. Interactive video stylization using few-shot patch-based training. *ACM Transactions on Graphics*, 39(4):73, 2020. 2
- [29] Dmitry Ulyanov, Andrea Vedaldi, and Victor S. Lempitsky. Instance normalization: The missing ingredient for fast stylization. *arXiv:1607.08022*, 2016. 2
- [30] Can Wang, Ruixiang Jiang, Menglei Chai, Mingming He, Dongdong Chen, and Jing Liao. Nerf-art: Text-driven neural radiance fields stylization. *arXiv preprint arXiv:2212.08070*, 2022. 8
- [31] 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. In *NeurIPS*, 2021. 1, 2
- [32] Kai Zhang, Nick Kolkin, Sai Bi, Fujun Luan, Zexiang Xu, Eli Shechtman, and Noah Snavely. Arf: Artistic radiance fields. In *European Conference on Computer Vision*, pages 717–733. Springer, 2022. 1, 2, 4, 5, 6, 7, 10
- [33] Kai Zhang, Gernot Riegler, Noah Snavely, and Vladlen Koltun. Nerf++: Analyzing and improving neural radiance fields. *arXiv:2010.07492*, 2020. 8