LoRA-based Efficient 3D Style Transfer

Jixuan He
jh2926@cornell.edu
Cornell Tech
New York, New York, USA

Andy Chang
ac2976@cornell.edu
Cornell Tech
New York, New York, USA

Toby Io
mi284@cornell.edu
Cornell Tech
New York, New York, USA

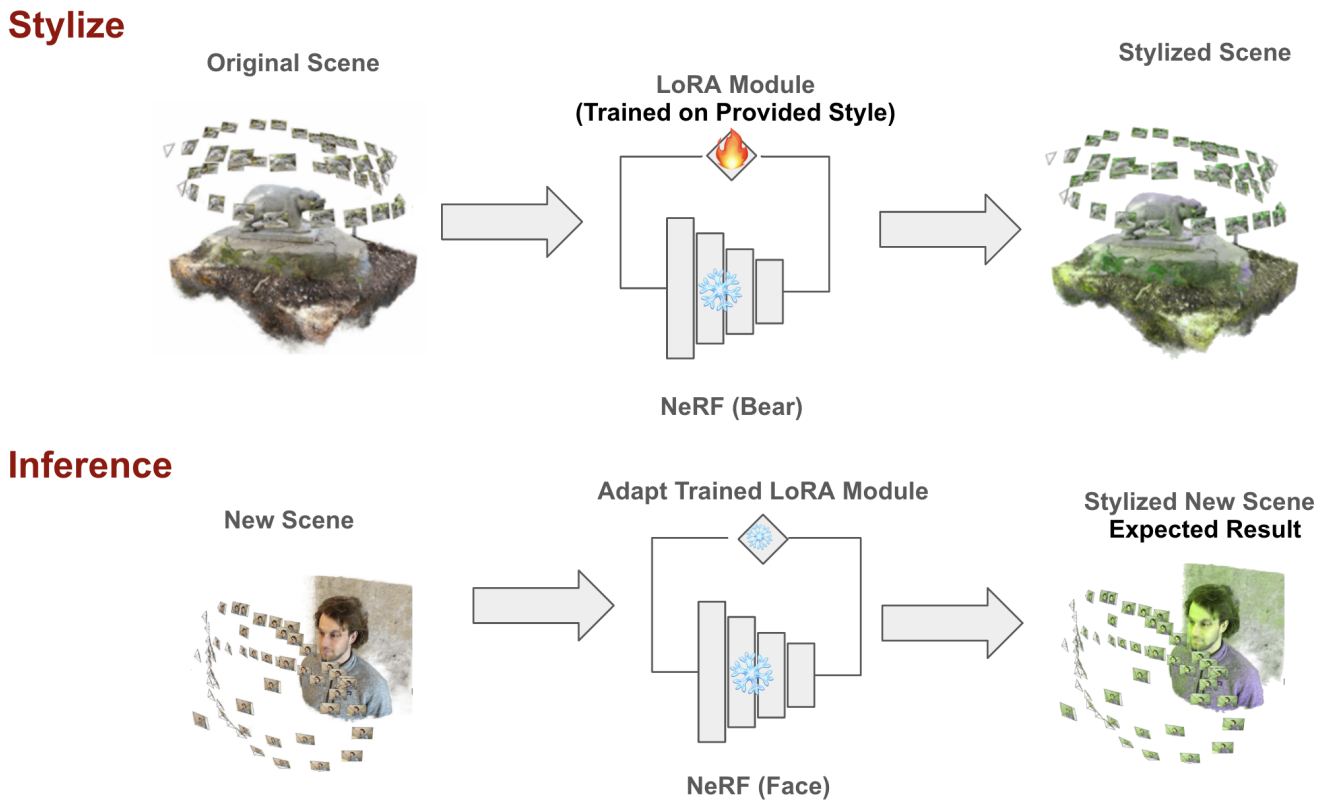

Figure 1: Overview of the proposed LoRA-based 3D style transfer. The LoRA module learns the style representation from a reference NeRF scene during training. At inference, the trained LoRA module is applied to a new NeRF scene, producing a stylized output without requiring retraining.

Abstract

3D style transfer enables the transformation of three-dimensional scenes into specific artistic styles, offering significant applications in gaming, design, and visualization. Traditional approaches often rely on computationally expensive iterative retraining or complex multi-stage zero-shot methods, both of which lack scalability and adaptability across diverse scenes. To address these limitations, our project introduces an efficient and reusable solution by leveraging Low-Rank Adaptation (LoRA) within Neural Radiance Fields (NeRFs).

Inspired by the success of network adapters in neural fine-tuning, we propose integrating a transferable LoRA module into NeRF pipelines. This LoRA module captures the stylistic representation of a scene and can be seamlessly applied to new scenes without additional retraining. Our experiments focus on three key areas: identifying the optimal placement of LoRA within NeRF (RGB head, density head, or both), exploring various style transformations (linear, non-linear, and complex), and incorporating CLIP-based semantic features to improve style understanding and generalization.

Preliminary results demonstrate the promise of our method for efficient 3D style transfer but reveal challenges in generalizing styles across scenes. Issues such as local minima in vanilla NeRF optimization and sensitivity to network initialization limit direct style transfer. As future work, we propose exploring hypernetwork-based meta-learning techniques or generalized-NeRFs to improve adaptability and robustness. The source code is available at https://github.com/KaKituken/efficient-3d-transfer.

1 Introduction

A style is more than just a set of visual patterns or techniques; it also embodies an artist's unique worldview. In the digital age, transferring stylistic elements from existing art to new scenes allows

users to explore and reinterpret the world through the "lens of the artist." Translating these stylistic elements into 3D scenes opens up transformative possibilities in gaming, design, and visualization. While 2D style transfer has been extensively studied, applying similar methods to 3D settings presents unique challenges. Unlike 2D images, which benefit from sufficient semantic priors, 3D scene representations such as neural radiance fields (NeRFs) [15] often lack semantic information, making 3D scene editing less flexible.

Previous works have achieved 3D style transfer and editing mainly in two ways. The first type of methods [14] learns a universal style transformation module for 3D-aware view features without optimizing the initial neural field. However, these approaches require massive pre-training on external datasets and don't allow regular neural rendering, thus making them difficult to be integrated seamlessly in related tools and software. The other type of methods either introduces diffusion priors [21, 5] or involves employing multi-view diffusion models for consistent image editing across multiple views, followed by reconstructing the 3D scene from the edited images [2, 23]. While these approaches are able to produce high-quality and consistent results, the intermediate techniques such as Score Distillation Sampling (SDS [17]) or Iterative Dataset Update (IDU) all require going through several passes of the diffusion model, making it time consuming. Furthermore, each desired style requires optimizing the model for each scene, making it highly inefficient.

We hypothesize that the same style across different scenes shares common latent semantic information, such as global stylistic features like color, tone, and texture, which are often independent of scene-specific details. This observation motivates our hypothesis that styles can be learned in a scene-agnostic manner, enabling consistent style transfer across 3D models. Therefore, we aim to develop an optimization method that **efficiently stylizes 3D models without requiring retraining for each new scene**. Inspired by 2D style transfer techniques—where learnable adapters like LoRA [7] capture and apply styles to diffusion models—we explore the potential of learning NeRF adapters that encapsulate the latent semantic information of a style.

Figure 1 provides an overview of our proposed LoRA-based 3D style transfer method. Our approach introduces a lightweight and reusable LoRA module to the NeRF pipeline, enabling efficient stylization of 3D scenes. By first training the LoRA module on a reference scene, it can be seamlessly adapted to new scenes during inference without retraining, significantly reducing computational costs. The figure illustrates this process, showing how a style learned from one scene (top row) can be effectively transferred to a new target scene (bottom row), yielding the expected stylized results. As a result, by training once on a single style, the learned adapter network could be applied on other NeRF scenes and construct 3D scenes that are stylized. This differs from current methods as we wouldn't need an additional pass through a diffusion model or incur large overhead to propagate styles and streamlines the inference process.

Our experiment results show that the proposed LoRA-based method effectively transfers certain global stylistic features, such as color and saturation transformations, across 3D scenes. However, challenges persist in handling more complex semantic-aware transformations, such as partial color filters and object-specific recoloring, which require improved disentanglement of style and structure. To address these challenges, we extend LoRA integration into a modified **LeRF** [11] framework, incorporating **CLIP embeddings** [18] for semantic guidance. These embeddings enhance the representation of high-level scene semantics, providing additional context for LoRA modules. While this approach improves semantic richness, achieving finer-grained stylization remains difficult, as LoRA is trained on a single scene and directly applied to novel scenes without fully leveraging broader contextual information. These findings highlight the need for more adaptive mechanisms that can dynamically capture and utilize semantic nuances, paving the way for future improvements in context-aware 3D style transfer.

In summary, the key contributions of this work are as follows:

(1) We propose a lightweight and reusable LoRA module for efficient and scalable 3D style transfer across NeRF scenes, eliminating the need for retraining for each new scene.
(2) We enhance the NeRF pipeline by incorporating CLIP embeddings into a modified LeRF framework, improving the representation of high-level scene semantics and enabling better disentanglement of style and structure.
(3) We conduct extensive experiments across various style transformations, identify challenges in handling complex semantic-aware styles, and propose future directions for adaptive mechanisms to improve context-aware 3D style transfer.

2 Related Work

2.1 Style Transfer in 2D and 3D

Style transfer has been widely explored in 2D images, with methods ranging from iterative optimization [4, 9] to real-time stylization [8, 13] via feed-forward neural networks. However, extending these techniques to 3D scenes is not at all trivial, as adding a new dimension introduces challenges such as multi-view consistency and preserving geometric fidelity across novel viewpoints. Recently, Neural Radiance Fields (NeRFs) [15] have become a primary method for 3D scene reconstruction and view synthesis due to its lightweighness and versatility, but their application to style transfer has revealed limitation in semantic richness and flexibility.

2.2 Existing Methods for 3D Style Transfer

3D style transfer techniques typically address the problem through one of the following approaches: 1) optimization-based, 2) zero-shot or feedforward, and 3) hybrid approaches:

2.2.1 ***Optimization-Based Methods:*** These approaches rely on iterative processes to stylize scenes, often utilizing powerful priors such as diffusion models or Score Distillation Sampling (SDS). Examples include Instruct-NeRF2NeRF [5], which refines training datasets iteratively but suffers from blurry artifacts and high computational costs, and DreamFusion [17], which achieves high-quality stylization through SDS but is impractical for scalable use due to its computational intensity. Additionally, learned styles in these methods are not transferable to new scenes, which is what our work tries to achieve.

2.2.2 ***Zero-shot and Feed-forward Techniques:*** Zero-shot and feed-forward methods have emerged to address the inefficiencies of optimization-based approaches. StyleRF [14] achieves multi-view

consistent stylization using feature grid transformations, but its multi-stage training process rely largely on external datasets and experience sub-optimal scene reconstruction with artifacts. FreeEditor [10] eliminates the need for retraining by leveraging a single-view editing scheme and Edit Transformer, enabling efficient and consistent style transfer across views. However, it introduced a huge overhead during inference due to the complex structure required to propagate the reference style. FPRF [12] takes this a step further by introducing a stylizable radiance field based on adaptive instance normalization (AdaIN) [8] and a style dictionary. These methods each introduce creative ways to stylize 3D scenes, but either incur additional computational overhead or produce sub-optimal scene reconstructions to achieve zero-shot transfer capabilities.

2.2.3 ***Hybrid approaches:*** Hybrid approaches combine elements of optimization and zero-shot techniques to leverage the strengths of both paradigms. Freditor [6] exemplifies this approach by performing frequency decomposition to separate low and high frequency components, enabling high fidelity, transferable stylization while maintaining multi-view consistency.

2.3 LoRA and Parameter-Efficient Fine-Tuning

Low-Rank Adaptation (LoRA)[7] was introduced as a parameter-efficient fine-tuning method[1] for large pre-trained models by injecting low-rank matrices into their weight layers. By freezing the base model and optimizing only the additional LoRA parameters, it achieves significant computational and memory savings without compromising performance. LoRA has been widely adopted across domains such as natural language processing and image generation, where fine-tuning large models for specific tasks or styles would otherwise be prohibitively expensive. In vision tasks, LoRA has also proven effective for fine-tuning diffusion models for image generation and segmentation, further underscoring its modularity and versatility.

In our work, we draw inspiration from the success of LoRA as a lightweight, modular fine-tuning approach. Rather than directly fine-tuning NeRF weights, we introduce LoRA modules into the NeRF pipeline as adapters. These modules capture and apply the latent semantic information of a style while leaving the base NeRF untouched. By doing so, a LoRA module trained on one scene could be directly used as a **plug-in** on new scenes without the need to retrain the underlying NeRF model. We enable efficient and reusable 3D style transfer without scene-specific retraining, significantly reducing computational overhead.

2.4 Semantic Embeddings in NeRFs

Recent works have explored integrating semantic embeddings into neural representations to enhance context-awareness. CLIP [18], a vision-language model, aligns image and text embeddings in a shared latent space, providing a versatile foundation for semantic tasks. Language Embedded NeRFs (LeRFs) [11] extend this idea by incorporating CLIP embeddings into NeRF pipelines to enable semantically informed 3D reconstructions. These methods demonstrate that semantic guidance can significantly improve tasks requiring high-level contextual understanding, such as object recognition and semantic segmentation.

Building upon these ideas, our work incorporates CLIP embeddings into the NeRF pipeline to enrich the semantic representation of 3D scenes. This integration enables the LoRA modules to disentangle style and structure more effectively, addressing the limitations of vanilla NeRFs in capturing semantic priors.

Our proposed method addresses key limitations of existing approaches by **eliminating iterative optimization** and **scene specific retraining**, reducing computational costs compared to Instruct-NeRF2NeRF and DreamFusion. Unlike StyleRF and FPRF, which use grid-based transformations or complex style dictionaries, our lightweight NeRF adapters, inspired by Low-Rank Adaptation (LoRA) [7], efficiently integrate into NeRF pipelines. By leveraging compact, trainable modules, our method captures latent semantic style information, enabling generalization across scenes without fine-tuning while maintaining multi-view consistency and photorealistic quality. If successful, these advantages would position our method as a scalable and practical alternative for 3D style transfer in applications like real-time rendering and virtual reality.

3 Methodology

3.1 Method Overview

Our method leverages the NeRFacto architecture and integrates Low-Rank Adaptation (LoRA) to achieve efficient and scalable style transfer. We incorporate CLIP embeddings into the architecture to provide the model with rich semantic information. The training process consists of three key stages: training a base NeRF to reconstruct 3D scenes, learning style transformations using LoRA modules, and adapting these modules to new scenes without fine-tuning. This workflow ensures efficiency, scalability, and real-time rendering support.

3.2 NeRF Architecture

The base architecture builds upon NeRFacto, which processes spatial coordinates through hash encoding and view directions through spherical harmonics encoding to predict density and RGB values.

The base architecture could be seen in Figure 3 and consists of several key components:

- **Input Encoding**
  - Spatial coordinates (x, y, z) are processed through hash encoding (Φ)
  - View direction (dir) is encoded using spherical harmonics encoding (SH)
- **Neural Network Architecture**
  - First MLP (θ) processes hash-encoded spatial coordinates to predict density
  - Second MLP (θ) combines:
    - Encoded view direction (SH)
    - Intermediate features from spatial coordinates
    - Appearance embeddings to predict final RGB colors
- **Output**
  - Density: Direct output from the first MLP
  - RGB: Color prediction from the second MLP

The NeRFacto architecture provides the foundation for our method, enabling reconstruction of 3D scenes. In the following section, we

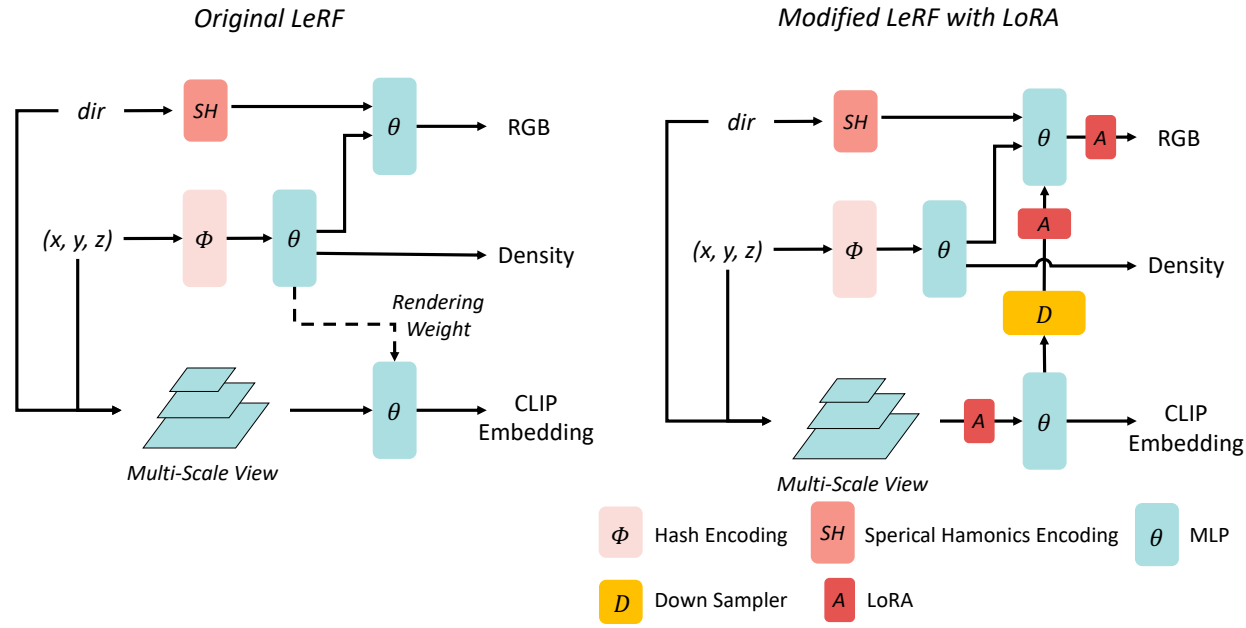

Figure 2: A visual explanation of our proposed method.

describe the training process that builds on this architecture to achieve style transfer.

3.3 Training Process

Our training workflow consists of three sequential stages: training the base NeRF scene, learning style transformations using LoRA, and adapting the trained LoRA modules to new scenes.

In the first stage, we train a NeRFacto model to accurately reconstruct the geometry and photometric properties of a target 3D scene. This involves optimizing the base NeRF using standard reconstruction losses, such as density-based photometric loss and view-consistency loss, which ensure that the model produces high-quality 3D representations. The resulting NeRF captures the structural and visual details of the scene, forming a robust foundation for subsequent style transfer. We also trained a modified LeRF model that incorporates CLIP embeddings. This enables the possibility of performing semantic-aware style transfer, leveraging the rich semantic information provided by the CLIP embeddings.

The second stage introduces LoRA modules into the RGB prediction layers and appearance embeddings of the NeRFacto model as well as the CLIP head of LeRF model. During this phase, the base NeRF weights remain frozen, and the LoRA parameters are updated to learn the target style. Style-consistency loss aligns the rendered RGB outputs with the target style, while perceptual losses further enhance feature alignment for stylistic fidelity.

In the final stage, the trained LoRA modules are applied to new NeRF scenes without additional fine-tuning. Thanks to the modular nature of LoRA, the style learned in the second stage generalizes effectively across diverse scenes, significantly reducing computational costs.

This training process emphasizes efficiency and generalization. Decoupling style learning from scene-specific training minimizes redundancy, while the reusability of LoRA modules across scenes enhances scalability. Moreover, the modularity of LoRA supports dynamic adjustments, making the method suitable for real-time rendering applications.

3.4 LoRA Implementation

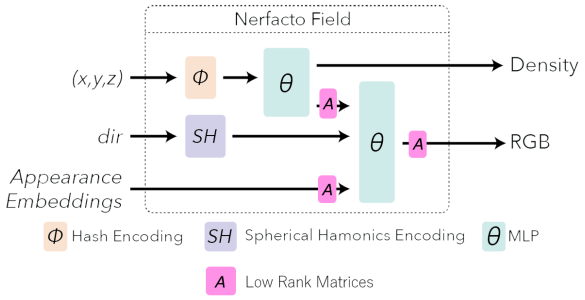

Figure 3: NeRFacto architecture and LoRA modules used at various heads.

We extended the NeRFacto framework by integrating LoRA modules at specific points in the network. As seen in Figure 3, we mainly focused on adding LoRAs before or behind the MLPs that predict RGB colors and process appearance embeddings.

These modules add low-rank matrices A and B to the linear layers, enabling the network to learn style-specific representations without altering the base NeRF weights.

In detail, LoRA modules augment the MLPs in the NeRFacto field by replacing each linear layer with a LoRA-augmented version. These modules use low-rank matrices ($r = 4$) and a scaling factor ($\alpha = 1.0$), initialized with Kaiming uniform for matrix A and zeros for matrix B. The original weights of the NeRF model are frozen during style training, ensuring that only the LoRA parameters (A, B) are updated. Each LoRA layer implements the transformation:

$$\text{output} = W(x) + \alpha(AB)x \tag{1}$$

where W represents the original layer weights, $A \in \mathbb{R}^{d\times r}$, and $B \in \mathbb{R}^{r\times k}$. This lightweight modification enables the network to learn style-specific representations efficiently while preserving the base NeRF's structural integrity.

We then created a module in NerfStudio that enables real-time rendering of the NeRF representation with the trained LoRA modifications. While the integration of LoRA modules enables efficient style transfer, vanilla NeRFs lack the semantic context required to disentangle style and structure effectively. To address this, we proposed to incorporate semantic embeddings from CLIP into our pipeline, as described in the next section.

3.5 CLIP-Enhanced Semantic Embeddings

To address the challenge of disentangling style and structure in 3D scenes, we incorporate semantic embeddings, specifically CLIP embeddings, into our NeRF pipeline. These embeddings not only provide the necessary context to NeRFs for understanding the semantic structure of 3D scenes, but also allow LoRA to learn and adapt to the underlying semantics of the input. This ensures that LoRA can effectively identify what initial semantics the input must contain when transferring it to a specific style.

To achieve this, we extend the language embedded NeRF (LeRF) framework [11], which integrates pre-trained CLIP embeddings into the NeRF pipeline to leverage their rich semantic features. By building upon LeRF, we aim to harness its semantic richness while adding lightweight LoRA modules for efficient and reusable style transfer. As shown in Figure 2, RGB prediction procedure is independent from the CLIP embeddings prediction in the original LeRF. Since we hope the LoRA on RGB head could learn the semantic information in CLIP embeddings, we modified the architecture to introduce CLIP embeddings into the RGB prediction head. These embeddings are concatenated with directional encodings (dir), appearance embeddings, and intermediate spatial features, enriching the RGB prediction process with high-level semantic context.

This integration ensures that the model can utilize semantic guidance to better align style transfer operations with the structural and contextual information of the scene. Simultaneously, the modular nature of LoRA enables computationally efficient training by fine-tuning only the lightweight adapter parameters while keeping the base NeRF and CLIP model weights frozen.

By incorporating CLIP embeddings into the RGB head, our method enhances the understanding of both style-specific transformations and the initial semantic content of the scene.

3.6 Implementation Details

To support real-time rendering of stylized NeRF scenes, we implemented a custom module in NerfStudio that dynamically applies the trained LoRA parameters during inference. This integration allows for interactive visualization of the stylized outputs without requiring additional computational passes.

4 Experiments

4.1 Dataset

Our experiments utilize the Instruct-NeRF2NeRF (in2n) NeRF dataset, which provides multi-view images of various scenes. This dataset will serve as the foundation for our style transfer experiments once to optimize our LoRA model. The in2n dataset is particularly suitable for our purposes due to its diverse scene types and high-quality multi-view captures.

We also curated a controlled set of style transformations on the in2n dataset to validate our proposed idea using standard image processing techniques. These transformations include:

- **Linear Transformations**
  - Color Filters: Uniform RGB shifts
  - Desaturation: Color to grayscale mapping
  - Negative: Inverts the RGB value
- **Non-Linear Transformations**
  - Saturation: Complex color intensity amplification
  - Color Restoration: Full colorization from grayscale
  - Partial Color Filters: Spatial-based RGB shifts

These synthetic transformations serve as sanity checks for our implementation, providing clear validation targets with known ground truth. Once the model demonstrates successful learning of these controlled transformations, we proceed to more complex artistic style transfers using in2n to generate NeRF models of various style.

4.2 Evaluation

We evaluate our style transfer results using both quantitative and qualitative metrics. For quantitative evaluation, we used Peak Signal-to-Noise Ratio (PSNR) to measure the reconstruction quality between our stylized outputs and the target style model. Additionally, we conduct human evaluation studies where we manually assess the visual quality and style similarity of our results compared to the target styles. This dual approach allows us to assess both the technical accuracy and perceptual quality of our style transfer method.

Upon the experiments, we focus on validating our idea on the control set of style transformations as mentioned in 4.1. In order to validate our idea of transferrable stylization, we first trained two Nerfacto models on a reference scene (the scene with a bear) and target scene (the scene with a face) respectively, and fine-tuned the model on the stylized reference scene using LoRA. We then applied the trained LoRA to the target scene to verify whether the LoRA module could transfer the style or not.

We conducted our experiments on the following settings:

- Linear Transformation
  - Color Filters
  - Gray Scale
  - Negative
- Non-Linear Transformation
  - Saturation
  - Partial Color Filter
- Complex Styles

We also tested the influence of using LoRA in different positions as well as the weight of the LoRA applied.

4.3 Linear Transformation

To evaluate the capacity of LoRA, we first ran experiments on linear transformations such as color filters, grayscale and negative. These linear transformations are easy to learn, and therefore are ideal settings to assess the effectiveness of using LoRA.

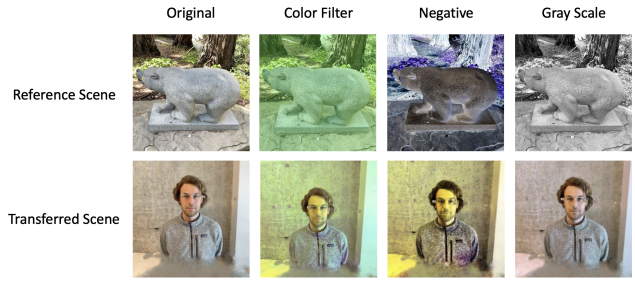

Figure 4: Linear Transformations

Figure 4 shows the results under three different linear transformation settings. We trained two Nerfacto models on the original reference scene and transferred scene respectively, as shown in the first column. We then fine-tuned the model using LoRA on different stylized reference scenes, then applied that LoRA to stylize the transferred scene, as shown in the last three columns. It is seen that LoRA is better at handling global atmosphere-like styles such as color filters, but it struggles to process some more complicated transformations such as negative and grayscale.

4.4 Non-linear Transformation

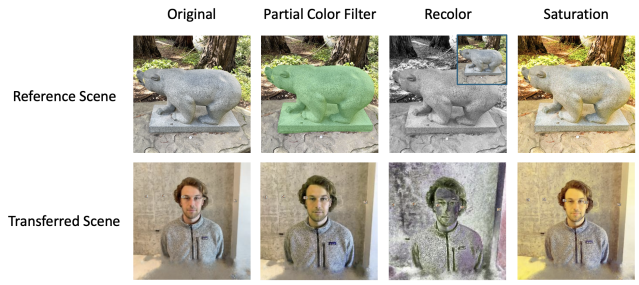

Figure 5: Non-linear Transformations

We then evaluate the capacities of LoRA as a stylization on non-linear transformations. Throughout this subset of transformations, a similar phenomenon can be observed as in Section 4.3. In in Figure 6, we can see that our introduced method is capable of capturing global-wise transformations such as saturation adjustment. However, for rather complex styles such as partial color filters (the second column) and recoloring, the results are rather suboptimal. The LoRA module is less likely to learn semantic information using this vanilla structure, thus resulting in the styles less transferrable to new scenes.

4.5 Complex Transformation

Figure 6: Complex Transformations

We also evaluated the model's performance on more complex artistic styles, such as Van Gogh's style. We utilized the Instruct-NeRF2NeRF pipeline for text-prompt driven style editing and incorporated a LoRA adapter into the model to capture the prompted style. We then attempted to transfer this LoRA to novel scenes.

Our results show that the method did not perform well on these more complex transformations. We hypothesize that LoRA adapters alone may be insufficient to capture complex style transformations due to the local optimization problem - different scenes may converge to different local minima, making style transfer inconsistent. Future work should focus on developing techniques to help LoRA learn more semantically meaningful representations that could enable better generalization across different scenes for complex style transformations.

4.6 Incorporating CLIP Semantic Embeddings

Figure 7: Complex Transformations

We extended the baseline experiments by incorporating CLIP semantic embeddings into the NeRF pipeline as mentioned in Section 3.5. We run the style transfer of color filter to illustrate the effect. The results, as shown in Figure 7, demonstrate that CLIP feature improves the awareness of content for LoRA. In this case, LoRA only makes the grass greener while preserving the color of the person and the tent. This selective transformation shows our method's

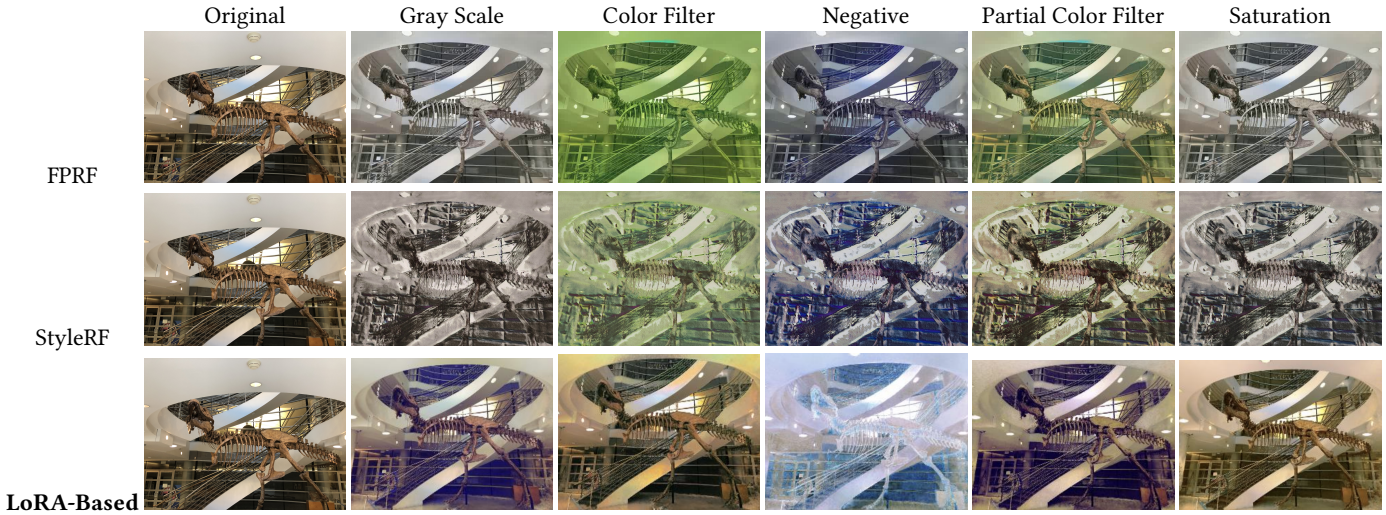

Figure 8: Qualitative comparisons. The proposed method is compared with the most related state-of-the-art methods, FPRF [12] and StyleRF [14].

potential to make content-aware style decisions, distinguishing between various scene elements. However, it still struggles to transfer complex and fine-grained style, indicating that current network architecture might be suboptimal for capturing sophisticated stylistic transformations.

We also observe that during the style transfer, LoRA on the RGB head dominates the output while LoRA on the CLIP branch contributes less to the final result. It suggests that the modified LeRF doesn't fully utilize the semantic embeddings and potential improvements could be made in better integrating and balancing the contributions from both branches during style transfer.

4.7 Qualitative Comparisons

To evaluate the effectiveness of our proposed method, we conducted qualitative comparisons with two state-of-the-art methods, FPRF[12] and StyleRF[14], as shown in Figure 8. All experiments were performed on an amazon EC2 instance with an NVIDIA A10G Tensor Core GPU with 24GB VRAM and 32GB RAM using the LLFF NeRF dataset, which is compatible with all models.

Computational Efficiency: FPRF exhibits significant advantages in computational efficiency, requiring only 1 hour for training and 3 minutes for inference. In contrast, StyleRF demands 5.5 hours for multi-step training (relying on the WikiArt dataset) and 30 minutes for inference. These differences highlight the trade-offs between processing speed and stylistic richness among the evaluated methods. Our method requires only 0.4 hours to train and 15 minutes for inference, while also demonstrating significantly lower GPU and CPU memory requirements compared to other proposed methods, highlighting the efficiency of our architectural approach to style transfer.

Visual Quality: As illustrated in Figure 8, the evaluated methods demonstrate distinct strengths and limitations: While StyleRF adapts scenes to provided styles in a zero-shot manner, its outputs often exhibit substantial artifacts, which degrade scene clarity and visual fidelity. This limitation stems from its reliance on external datasets and complex style propagation, which may not generalize well to diverse scenes. FPRF produces cleaner, artifact-free outputs but operates as a view-consistent color filter. While it preserves scene geometry and consistency across views, its stylization capabilities are relatively limited, primarily applying global transformations such as color adjustments rather than detailed style adaptation. TThe last row shows the results from our LoRA-based method. While the model struggles to capture fine-grained content-dependent styles such as localized color filters, it demonstrates strong capability in transferring global style attributes like image negation and saturation adjustments. These results suggest that the current LoRA architecture is effective at capturing and transferring scene-agnostic styles.

Table 1: Efficiency Comparison of various methods

Metric	StyleRF	FPRF	LoRA
Final Model PSNR	26.22	29.86	**32.88**
Total Trainable Parameters	42.8M	94.48M	**2.6M**
GPU Memory (Training)	21.0GB	5.6GB	**3.8GB**
CPU Memory (Training)	12.7GB	10.6GB	**1.5GB**
Training Time (hrs)	5.5	1.2	**0.4**
Inference Time (mins)	30.6	**3.5**	15

4.8 Ablation Studies

We conducted ablation studies on the influence of using different hyper-parameters. Figure 9 shows the transferred stylized results for various training steps and where to add the LoRA module. It's indicated that training converges around 6K steps, and transfer of the best quality can be achieved if we add LoRA to both density and RGB heads.

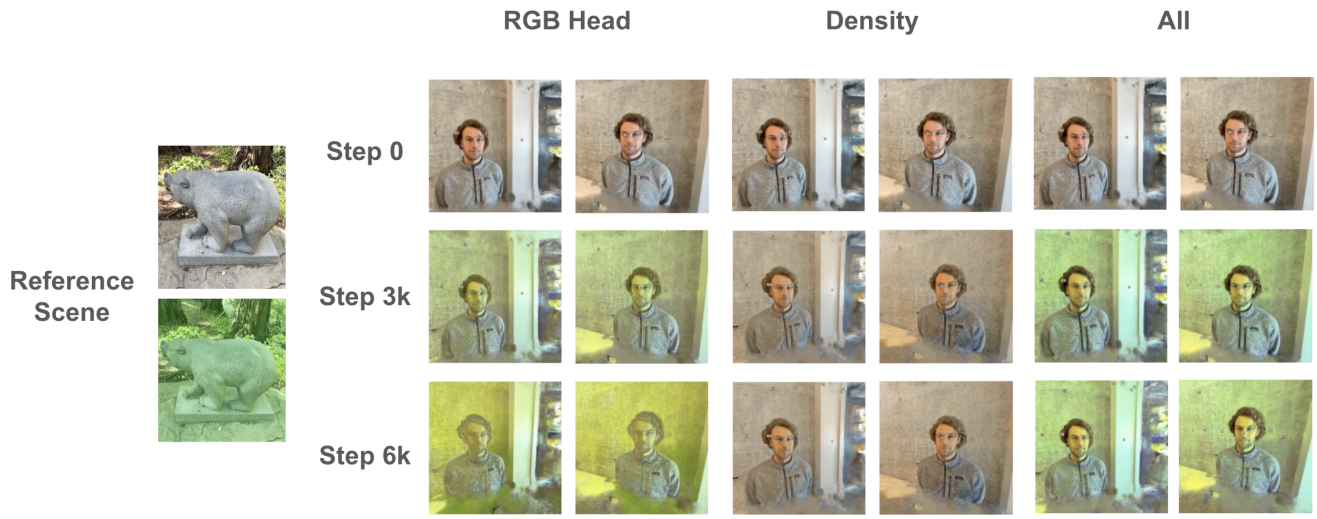

Figure 9: Low Rank Adapters used at various positions

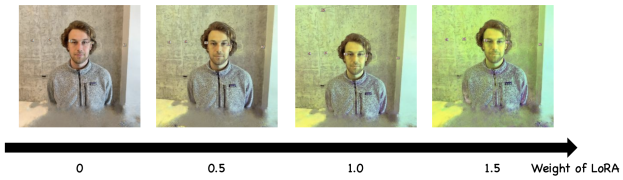

Figure 10: Low Rank Adapters used at various weights

Following the community's consensus on using LoRA, we also added additional control to the weight of LoRA modules. By modifying the weights, users can control how much style to blend into the original image. Figure 10 shows results of using different weights under the color filter setting. The filter intensity increases when the user increases the weight.

4.9 Discussions

After conducting the experiments, we realized that the results were not as good as expected even when incorporating CLIP semantic embeddings. In this section, we discuss possible reasons for these challenges and outline potential directions to address them.

4.9.1 ***Instability of LoRA.*** Currently, our LoRA is only dependent on the original model and the target style, and it lies in a feature space associated with the model's weights. Since it doesn't rely on any semantic features, LoRA is sensitive to changes in the model's weights, which makes transferring between different scenes challenging.

To illustrate this instability, we designed a self-transfer experiment to test LoRA's performance. We first trained two Nerfacto models on the same scene but using different random seeds. This led to a slight difference in the trained model weights. Then, we trained a LoRA on the first model to learn the color filter and transferred this LoRA to the second model. Ideally, we would get similar outputs because the scenes are identical, but as Figure 11 shows, there is a difference in color saturation in the learned style when compared to the original model. This experiment demonstrates that even minor variations in model weights can significantly impact LoRA's style transfer capabilities.

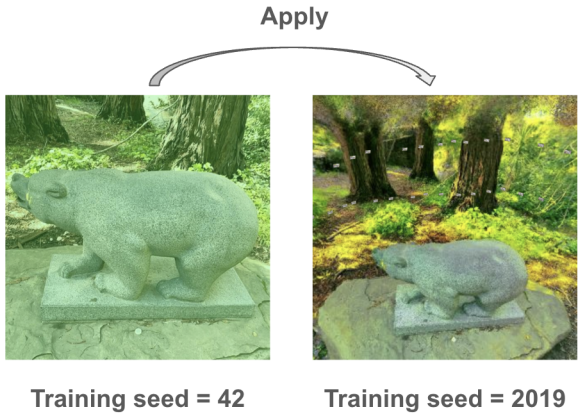

Figure 11: LoRA's instability demonstrated using same scene style transfer.

4.9.2 ***Local Minimum of vanilla NeRFs.*** While integrating CLIP semantic embeddings and LoRA modules improves the ability to generalize style across scenes, our experiments revealed a limitation tied to the optimization properties of vanilla NeRFs. Specifically, NeRF models are inherently optimized to fit a **specific scene**, converging to a local minimum that best reconstructs the geometry and

appearance of that scene. This scene-specific optimization makes it difficult to transfer learned style parameters (LoRA) seamlessly across different scenes.

As illustrated in Figure 12, the challenge can be visualized as follows:

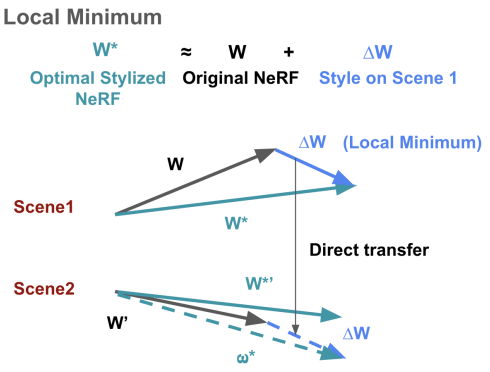

Figure 12: Misalignment of Local Minima in NeRFs During Style Transfer. The style adjustment (ΔW) learned for Scene 1 does not align with the weight distribution of Scene 2, leading to incorrect stylization (dashed arrow) when directly transferred.

For Scene 1: The original NeRF representation W is optimized to a local minimum that reconstructs the specific scene. To stylize this scene, we introduce a style adjustment ΔW (blue arrow) using LoRA, resulting in the optimal stylized representation W^*.

For Scene 2: The NeRF weights W' are independently optimized to a local minimum for Scene 2's geometry and visual properties. Directly transferring the style adjustment ΔW (from Scene 1) to Scene 2 assumes that the weight space alignment (between W and W') is consistent. However, this assumption might not hold because the two local minima are **misaligned due to scene-specific optimization**.

To address this issue, we explored methods to align the weight spaces of different NeRF models using **shared semantic priors**, such as CLIP features at the initialization stage. The intuition was that incorporating CLIP embeddings during training could provide a common semantic representation, improving the alignment of W and W'.

However, our experiments revealed that CLIP embeddings alone are not sufficient. This is because:

1. Optimization in a Single Scene-Specific Space:
While CLIP embeddings enrich the semantic representation, the NeRF is still optimized within a single scene, and the resulting semantic prior remains a **subset of the entire CLIP feature space**. As a result, LoRA fine-tuned on this representation lacks the broader contextual understanding necessary to generalize effectively across scenes.

2. Limited Generalization:
Even with the shared semantic prior, LoRA cannot overcome the misalignment caused by independent scene-specific optimization, as NeRFs still lack a globally shared weight space.

4.9.3 ***Potential Directions for Future Work.*** To resolve this issue, we believe that leveraging generalized NeRFs or alternative approaches for learning shared representations could improve LoRA-based style transfer:

1. Generalized NeRFs:
Methods such as IBRNet [22] or GNT [19] optimize NeRF models that generalizes across multiple scenes. This approach produces a **joint weight space** shared across different scenes, which could provide a better foundation for LoRA modules to generalize.

2. Hypernetworks and Meta-Learning:
Hypernetwork or meta-learning techniques [3] could enable the learning of generalized style parameters. Hypernetworks can adaptively generate NeRF weights conditioned on specific scenes, mitigating the issue of weight space misalignment. Similarly, meta-learning approaches could train NeRF models to quickly adapt to new scenes while maintaining shared style representations.

By addressing the local minimum problem using generalized NeRFs, hypernetworks, or meta-learning, we can create shared weight spaces that improve the transferability of style adjustments. These solutions would allow LoRA modules to operate on consistent, scene-agnostic representations, resulting in more robust and accurate style transfer across diverse 3D scenes.

5 Conclusion

We proposed a LoRA-based approach for efficient and reusable 3D style transfer within Neural Radiance Fields (NeRFs). By integrating CLIP-based semantic embeddings, our method effectively transfers global stylistic features, such as color and saturation, across diverse 3D scenes without requiring scene-specific retraining. However, challenges remain in handling complex semantic-aware transformations due to the scene-specific optimization of vanilla NeRFs.

Our findings suggest that generalized NeRFs or meta-learning techniques could address these limitations by enabling shared semantic weight spaces for better adaptability. This work provides a foundation for lightweight, scalable 3D style transfer with applications in gaming, virtual reality, and real-time rendering.

6 Appendix

Our method is implemented in PyTorch [16] and utilizes the NeRFStudio [20] library for visualization. We build our method on top of NeRFacto [20] and follow the implementation of incorporating CLIP embeddings from the work LeRF [11]. For more details, our source code can be found here: https://github.com/KaKituken/efficient-3d-transfer.

References

[1] Elad Ben Zaken, Yoav Goldberg, and Shauli Ravfogel. 2022. BitFit: simple parameter-efficient fine-tuning for transformer-based masked language-models. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Smaranda Muresan, Preslav Nakov, and Aline Villavicencio, (Eds.) Association for Computational Linguistics, Dublin, Ireland, (May 2022), 1–9. DOI: 10.18653/v1/2022.acl-short.1.

[2] Jun-Kun Chen, Samuel Rota Bulò, Norman Müller, Lorenzo Porzi, Peter Kontschieder, and Yu-Xiong Wang. 2024. Consistdreamer: 3d-consistent 2d diffusion for high-fidelity scene editing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 21071–21080.

[3] Pei-Ze Chiang, Meng-Shiun Tsai, Hung-Yu Tseng, Wei-Sheng Lai, and Wei-Chen Chiu. 2022. Stylizing 3d scene via implicit representation and hypernetwork. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 1475–1484.

[4] Leon A Gatys, Alexander S Ecker, and Matthias Bethge. 2016. Image style transfer using convolutional neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2414–2423.

[5] Ayaan Haque, Matthew Tancik, Alexei A Efros, Aleksander Holynski, and Angjoo Kanazawa. 2023. Instruct-nerf2nerf: editing 3d scenes with instructions. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 19740–19750.

[6] Yisheng He, Weihao Yuan, Siyu Zhu, Zilong Dong, Liefeng Bo, and Qixing Huang. 2025. Freditor: high-fidelity and transferable nerf editing by frequency decomposition. In *European Conference on Computer Vision*. Springer, 73–91.

[7] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*.

[8] Xun Huang and Serge Belongie. 2017. Arbitrary style transfer in real-time with adaptive instance normalization. In *Proceedings of the IEEE international conference on computer vision*, 1501–1510.

[9] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. 2016. Perceptual losses for real-time style transfer and super-resolution. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part II 14*. Springer, 694–711.

[10] Nazmul Karim, Hasan Iqbal, Umar Khalid, Chen Chen, and Jing Hua. 2025. Free-editor: zero-shot text-driven 3d scene editing. In *European Conference on Computer Vision*. Springer, 436–453.

[11] Justin Kerr, Chung Min Kim, Ken Goldberg, Angjoo Kanazawa, and Matthew Tancik. 2023. Lerf: language embedded radiance fields. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 19729–19739.

[12] GeonU Kim, Kim Youwang, and Tae-Hyun Oh. 2024. Fprf: feed-forward photo-realistic style transfer of large-scale 3d neural radiance fields. In *Proceedings of the AAAI Conference on Artificial Intelligence* number 3. Vol. 38, 2750–2758.

[13] Xueting Li, Sifei Liu, Jan Kautz, and Ming-Hsuan Yang. 2019. Learning linear transformations for fast image and video style transfer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 3809–3817.

[14] Kunhao Liu, Fangneng Zhan, Yiwen Chen, Jiahui Zhang, Yingchen Yu, Abdulmotaleb El Saddik, Shijian Lu, and Eric P Xing. 2023. Stylerf: zero-shot 3d style transfer of neural radiance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 8338–8348.

[15] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. 2021. Nerf: representing scenes as neural radiance fields for view synthesis. *Communications of the ACM*, 65, 1, 99–106.

[16] Adam Paszke et al. 2019. Pytorch: an imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32.

[17] Ben Poole, Ajay Jain, Jonathan T Barron, and Ben Mildenhall. 2022. Dreamfusion: text-to-3d using 2d diffusion. *arXiv preprint arXiv:2209.14988*.

[18] Alec Radford et al. 2021. Learning transferable visual models from natural language supervision. In *International conference on machine learning*. PMLR, 8748–8763.

[19] Mukund Varma T, Peihao Wang, Xuxi Chen, Tianlong Chen, Subhashini Venugopalan, and Zhangyang Wang. 2023. Is attention all that neRF needs? In *The Eleventh International Conference on Learning Representations*. https://openreview.net/forum?id=xE-LtsE-xx.

[20] Matthew Tancik et al. 2023. Nerfstudio: a modular framework for neural radiance field development. In *ACM SIGGRAPH 2023 Conference Proceedings* (SIGGRAPH '23).

[21] Can Wang, Ruixiang Jiang, Menglei Chai, Mingming He, Dongdong Chen, and Jing Liao. 2023. Nerf-art: text-driven neural radiance fields stylization. *IEEE Transactions on Visualization and Computer Graphics*.

[22] Qianqian Wang, Zhicheng Wang, Kyle Genova, Pratul P Srinivasan, Howard Zhou, Jonathan T Barron, Ricardo Martin-Brualla, Noah Snavely, and Thomas Funkhouser. 2021. Ibrnet: learning multi-view image-based rendering. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 4690–4699.

[23] Ethan Weber, Aleksander Holynski, Varun Jampani, Saurabh Saxena, Noah Snavely, Abhishek Kar, and Angjoo Kanazawa. 2024. Nerfiller: completing scenes via generative 3d inpainting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 20731–20741.