

Gaussian Splatting in Style

Abhishek Saroha^{1,2}, Mariia Gladkova^{1,2}, Cecilia Curreli^{1,2}, Dominik Muhle^{1,2}, Tarun Yenamandra^{1,2}, and Daniel Cremers^{1,2}

¹ Technical University of Munich

² Munich Center for Machine Learning, Germany

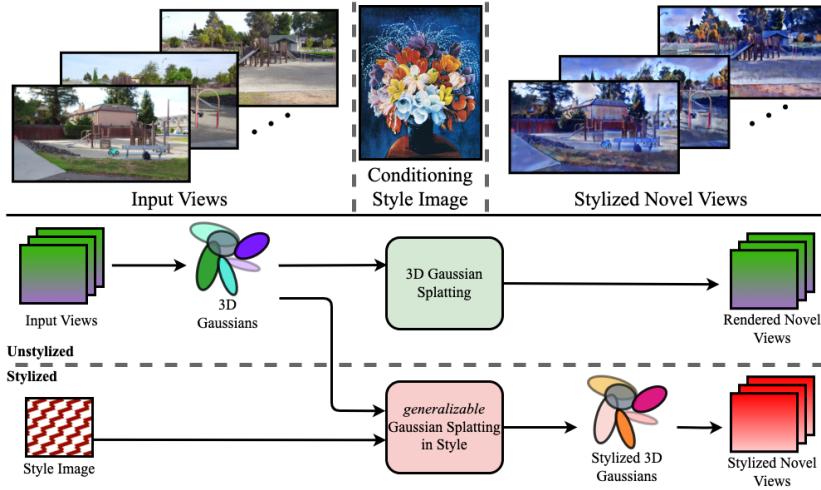


Fig. 1: Given multi-view images of a real-world scene, we perform the task of scene stylization. Unlike previous scene stylization approaches, we do not need to fit a scene to each new style. Employing a neural network, conditioned on a style image, allows us to generalize to a variety of styles. Our method can generate 3D consistent scene stylization at approximately 150 FPS.

Abstract. 3D scene stylization extends the work of neural style transfer to 3D. A vital challenge in this problem is to maintain the uniformity of the stylized appearance across multiple views. A vast majority of the previous works achieve this by training a 3D model for every stylized image and a set of multi-view images. In contrast, we propose a novel architecture trained on a collection of style images that, at test time, produces real time high-quality stylized novel views. We choose the underlying 3D scene representation for our model as 3D Gaussian splatting. We take the 3D Gaussians and process them using a multi-resolution hash grid and a tiny MLP to obtain stylized views. The MLP is conditioned on different style codes for generalization to different styles during test time. The explicit nature of 3D Gaussians gives us inherent advantages over NeRF-based methods, including geometric consistency and a fast training and rendering regime. This enables our method to be useful for various practical use cases, such as augmented or virtual reality. We demonstrate that our method achieves state-of-the-art performance with superior visual quality on various indoor and outdoor real-world data.

Keywords: Scene Stylization · Gaussian Splatting

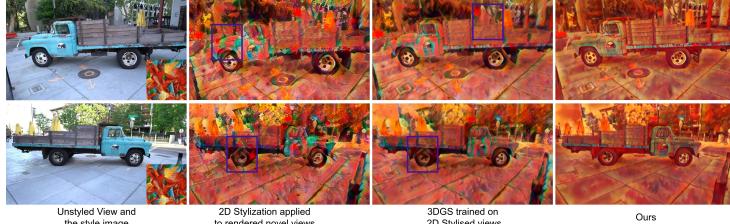


Fig. 2: The motivation from our work stems from the requirement of a specialized method that, while stylizing a scene, considers the spatial information into account. We show that to generate stylized novel views of a scene, it is insufficient to stylize the rendered views or train a scene representation model on stylized 2D images. It leads to loss of information, such as deformity in the solid truck’s body shown above.

1 Introduction

Driven by the curiosity of how a particular image would appear if painted by Van Gogh, Picasso and other artists, style transfer has been at the heart of the image processing community for a long time. It can be seen as a challenge of texture transfer. The core idea of neural style transfer is, given a style image with desired aesthetics and a target image with pursued content, to apply the features of a style image such as colors and texture to the content image while keeping information inside the content image intact. Earlier works [13, 81, 37, 12] devised powerful algorithms for texture synthesis. Further advancements in deep learning gave rise to works [16, 31, 27] that approach the problem by suitably trained neural networks. The ability of neural networks to effectively grasp the features of an image at multiple levels of details made them a preferred choice over non-learning based algorithms[16]. Fuelled by the availability of 3D data, this phenomenon of style transfer was extended to the stylization of scenes[28, 26, 43, 87], and is referred to as *scene stylization*, which is the primary focus of this work.

The ability to control the visual properties of a scene is central to many practical applications such as avatar modeling or scene editing [60, 51, 64, 91] to name a few. With the wide adaptability and increasing interest in various augmented and virtual reality applications, it is not only necessary that these techniques are accurate, but also real-time capable to ensure the best user experience.

Over the past few years, different methods have been proposed to efficiently represent a complex 3D scene. [59, 82, 77, 71] were some methods leveraging an explicit pointcloud representation. The domain witnessed an incredible pace of growth following the pioneering work of Neural Radiance Fields(NeRFs)[50, 49]. NeRFs gained large popularity, primarily due to its ability to represent a complex scene in the weights of a single multi-layer perceptron(MLP), while capturing

fine details during novel view synthesis. Nonetheless, the proposed method required long training time and it was slow during inference. Therefore, many variants of NeRFs emerged such as [66, 86, 1, 54, 57, 3] that improved the seminal work significantly. Recently, 3D Gaussian Splatting(3DGS)[33] further pushed the boundaries of novel view synthesis, especially in terms of consistency and speed. 3DGS is, by design, more explicit in terms of storing scene information in the form of 3D Gaussians, with each Gaussian having certain properties attached to it. Additionally, 3DGS renders views in real-time, thus making it highly suitable for practical applications.

In this paper, we tackle the problem of learning to stylize a set of 3D Gaussians given a set of style images, namely scene stylization. Therefore, we call our approach as Gaussian Splatting in Style(GSS). We condition this representation on a style image to obtain stylized views of the complex 3D scene. Utilizing these Gaussians as the backbone of our approach ensures spatial consistency. Due to the proposed design, our model stylizes a scene given any style image and does not require re-training as some of the previous works on scene stylization such as ARF [87]. Furthermore, our approach does not add any overhead to the existing real-time rendering speed of 3DGS, making it a preferred choice for AR/VR applications. We demonstrate the superiority of our method on well-established consistency metrics, showing the best performance in both short-term and long-term measurements. We further support our claims with the qualitative analysis of stylized rendered views comparing our work to existing scene stylization baselines.

To summarise, our contribution in this paper are as follows:

- We develop a novel state-of-the-art method, GSS, to perform neural scene stylization in real-time based on 3D Gaussian splatting. We are among the first to perform scene stylization using 3D Gaussians.
- We demonstrate the effectiveness of our method by comparing against various types of baselines both quantitatively and qualitatively across various real-world datasets across different settings.

2 Related Work

2.1 3D Scene representation

Complex 3D scenes can be represented in many ways. One of the most common and well known ways is that of using point clouds[26, 2]. Similar to point clouds, some works such as [69, 85] use a voxel grid for representing a 3D scene. Voxel grids are not the most ideal choice for scene representation due to their high memory footprint. Neural implicit scene based methods, led by the works of [62, 46, 63] making use of signed distance fields(SDFs) and occupancies respectively, helped solve this memory issue. In principle, implicit neural representations can generate meshes and surfaces upto an arbitrarily high resolution. Recently, work of NeuralAngelo [42] set a new benchmark as it could recover a highly detailed 3D surface of a large-scale scene from a set of input images. Following a similar

direction neural radiance fields, also known as NeRFs[50] are also an implicit method that is described in further detail below.

2.2 Novel View Synthesis

The task of generating unseen views of an object/scene given a collection of input images is known as novel view synthesis. Classical works, such as [39, 5, 10, 92, 22, 70, 17] aimed at generating novel views directly from the given set of multi-view images. With the onset of deep learning, new approaches such as [21, 74, 73, 67] paved the way for neural network-based novel view synthesis(NVS). All these methods represented the scene in their own ways. A major milestone in the domain of NVS was with the advent of Neural Radiance Fields(NeRFs) [50]. NeRFs represent the scene as the weights of a multi-layer perceptron(MLP). The input to these MLPs are points in space and the viewing direction, which then gives out the density and color for each queried point. Despite its success, NeRF suffered from many drawbacks. A slow training regime, followed by the need to optimize it for each scene made it less viable for many use cases. It was also observed that NeRFs performed extremely well when the input views were dense and covered many angles of the scene. It often failed to get the finer details from views far-away from those used in training. Follow up works such as [57, 9, 18, 86] focused specifically on solving NVS for a sparse set of input views. Meanwhile, [86, 83, 55] focused on a more generalized framework to avoid fitting a MLP to each new scene, while [1, 85, 66, 54] improved on the original work in terms of training and inference speed. Recently, 3D Gaussian Splatting(3DGS) [33] also tackled the task of novel view synthesis in real-time. We explain the idea of 3DGS in Section 3 and can recommend the readers to visit [33] for more details.

2.3 Style Transfer and Scene Stylization

Neural style transfer is the task of transferring the style of a source image onto the target(or content) image[16]. It is a complex task as the transfer needs to preserve the information present in the target image. This problem can also be formulated as that of texture transfer, and was tackled in the works of [13, 12, 81, 37] before the widespread adoption of neural networks. Later, the pioneering work of [16] successfully applied the use of neural networks to the task of style transfer. The work made use of a pretrained VGG[72] to extract the semantic information from an image at different hierarchical levels. [16] optimized a loss function composed of two terms, content loss and style loss, which represented the similarity of the generated image with the content image and the style image respectively. Since [16] worked by running an optimization for each target and style image, it was not suitable for real-time applications. This was further improved by [40, 31, 78, 27] by being real-time and providing the flexibility of general style conditioning. Additionally, the works of [44, 30] added the sophistication of monocular depth estimation for preserving the depth information present in the source image. Furthermore, works of [25, 19, 15, 4, 6, 80] extended the work of 2D style transfer to a video sequence by making use of advancements in methods like optical flow[11]. However, [26] and our experiments show that simply extending 2D style

transfer to 3D scenes result in visual artifacts such as blurriness and inconsistency across different views, as also depicted in Figure 6.

Stylizing scenes given a style image has gained prominence in recent years, especially with the arrival of AR/VR headsets in the mainstream. One way to distinguish between different methods is based on their way of scene representation. Works of [26, 52, 2] work with a point cloud based representation. [26] for instance, project the style features from 2D into 3D and transform them into that of the style image before re-projecting them back into 2D for getting stylized views. Similarly, [23] performs scene stylization of indoor scenes in the form of a mesh. Recently, radiance fields [50] gained huge popularity due to their ability to generate novel views with a very high quality. Due to this, a lot of works made use of a NeRF backbone to perform scene stylization. One direction of approaching the problem is by focusing to optimize a radiance for each given style image. Methods such as [32, 87, 41, 8, 56, 84, 89, 61, 90] follow this outline. Methods such as [7, 8] make use of a hypernetwork to impart the style image features onto the rendered views of a complex 3D scene. On the other hand, works such as [28, 43] provide a generalizable scene stylization framework, i.e a stylized scene can be generated at inference time given a conditioned style image input. Our work also follows in this category. Another direction in this domain lies making use of additional information such as depth maps [32] since their primary focus lies in geometry preservation. In this work, instead of a radiance field, we make use of a Gaussian splatting[33] framework.

Apart from image based conditioning for stylization, one direction is to input the conditioning in the form of text. Leveraging the use of large language models(LLMs), [48, 47, 29, 79, 20, 75, 14] perform scene editing, and stylization in some cases, via text-based inputs. In our work, we do not compare with any text-based input methods and focus entirely on methods that condition the scene via style images only. While textual information can be informative and descriptive, the challenges of image-based conditioning can be far more demanding as it provides information in terms of abstraction, strokes, and so on.

3 Preliminaries

3.1 3D Gaussian Splatting

3D Gaussian Splatting(3DGS)[33] is an explicit method for scene representation, that is especially useful for high quality real-time rendering. We briefly introduce the main concepts for completeness.

Gaussian Splatting can be considered as a form of point cloud representation. Each scene in 3DGS is represented as a bunch of 3D Gaussians, each having certain properties. Specifically, each 3D Gaussian is defined by its mean position $\mu \in \mathbb{R}^3$ and a covariance matrix $\Sigma \in \mathbb{R}^{3 \times 3}$ as

$$G(X) = e^{-\frac{1}{2}\mu^T \Sigma^{-1} \mu}. \quad (1)$$

Since the covariance matrix has to be positive semi-definite for differentiable optimization, it is decomposed into a rotation matrix \mathbf{R} and a scaling matrix $\mathbf{S} \in \mathbb{R}^3$ as

$$\Sigma = \mathbf{R} \mathbf{S} \mathbf{S}^T \mathbf{R}^T. \quad (2)$$

Additionally, each Gaussian also contains the opacity α and view-dependent color values, represented by spherical harmonics coefficients. Once initialised, these Gaussians are projected onto 2D for rendering. This rendering process is done by a differentiable tile rasterizer. During the process of optimization, these Gaussians undergo density control, i.e they are constantly removed and new ones are added to the scene in order to maintain high image quality. This is done in order to ensure that the density of Gaussians is more or less consistent in all areas of the scene, especially ones which were empty during the SfM initialization. The 3D scene is optimized for the following loss function:

$$\mathcal{L} = (1 - \lambda)\mathcal{L}_1 + \lambda\mathcal{L}_{D-SSIM}, \quad (3)$$

where \mathcal{L}_1 and \mathcal{L}_{D-SSIM} are computed between the generated image and the ground truth view. For thorough explanation, we refer the reader to [33].

3.2 Stylized NeRF

StylizedNeRF [28] set up a new benchmark in the field of neural scene stylization. Built upon the framework of NeRF and Nerf-W [50, 45], this work makes use of a mutual learning strategy to combine the 2D stylization method AdaIN [27] and a pre-trained NeRF architecture. The high-level idea is to predict a new stylized color conditioned on the style image. This is achieved by replacing the color component with a new style module while keeping the density prediction from the pre-trained NeRF constant to maintain geometric consistency. To facilitate mutual learning between 2D-3D stylization, StylizedNeRF makes the decoder of AdaIN trainable for fine-tuning the final results. Inspired by Nerf-W, they also employ a pre-trained VAE that provides learnable latent codes for the conditioned style inputs. Motivated by the positive results of StylizedNeRF, GSS also makes use of pre-trained Gaussians to better represent the geometric details of the complex scenes.

4 Gaussian Splatting in Style

Our goal is to stylize a complex 3D scene. Our overall training process is similar to that of 3DGGS[33, 38], where the scene is learnt by minimizing the difference between multiple training views and their corresponding ground truth images. In addition to learning the scene representation, we learn a mapping between the different views of the scene and the queried style image. To this end, we employ an MLP that learns the mapping between the various 3D Gaussians and their stylized color based on the style image. During inference time, we generate novel views conditioned on an input style image. It is worth noting that the input style image at inference time may or may not have been a part of the dataset of

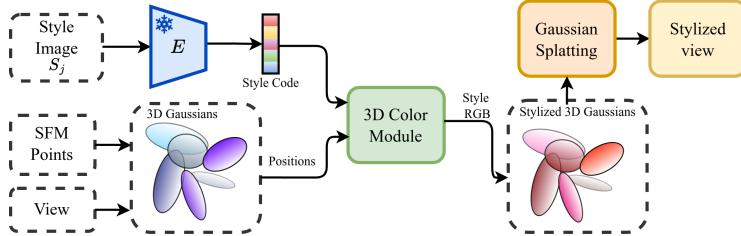


Fig. 3: Here we diagrammatically show the overall architecture of our pipeline. We employ a novel 3D Color module that is jointly trained with the 3D Gaussians, to predict the new colors for each Gaussian based on the querying style image at test time.

style images used in the training process. In the following section, we elaborate on the various constituents of our pipeline.

3D Color Module The first component of our pipeline is a 3D Color module, responsible for predicting the colors of each 3D Gaussian based on the conditioning style image. The input to this module are the mean positions μ of the 3D Gaussians and a latent code representing the style image obtained through the use of a pretrained encoder[65, 28]. The mean input positions are fed into a multi-resolution hash grid(MHG) [54] while the latent codes are fed into an FC layer before concatenating them together. This resulting tensor is then passed through an MLP [53, 38] to obtain a new RGB color for each input Gaussian. The use of MHG and tiny MLP was motivated by the fact that each scene has a large number of Gaussians, and could also be in the order of millions for large scenes such as those in the TnT[35] dataset. The predicted RGB from the 3D color module is fed into the 3DGs pipeline, and the Gaussians, with their new colors are rendered to generate a 3D consistent stylised view \mathcal{I}_{gen} . Since our color module is view independent, it does not affect the runtime during inference time. Furthermore, the geometry is conserved, one forward pass through the proposed pipeline with the conditioned style input allows us to obtain the stylized RGB for any arbitrary viewpoint.

2D Stylization Module The 2D Stylization module is based on AdaIN [27]. It consists of three components, a pretrained VGG [72] as the encoder, a convolutional decoder, and adaptive instance normalisation layer. For details, please refer to [28]. As shown in Figure 3, we pass the style image and the original ground truth view through a pretrained VGG to obtain their features. These features are then passed through the adaptive instance normalization layer and the decoder to obtain the 2D styled image of that particular view. This 2D stylized image acts as a guide for our 3D color module, and gives a rough indication as to how the scene should appear from that specific viewpoint. Therefore, this image, represented by $\mathcal{I}_{style2d}$ forms on what we call it as a Guide Loss \mathcal{L}_g , which is detailed in Section 4.

Training Regime We build on top of vanilla 3DGs[33]. During the training process, we learn the scene in conjunction with the 3D Color module. To this

end, we first start by learning the scene without the color module and train on unstyled training views. The motivation for this "warm-up" phase arises to encourage the network to learn the original geometry, and therefore have an inherent notion of being consistent.

Loss function As shown in Figure 3, we apply the loss functions on the views generated by the 3D color module. Our loss term is therefore made up of the **guide loss** \mathcal{L}_g and is elaborated as

$$\mathcal{L}_g = \|\mathcal{I}_{style2d} - \mathcal{I}_{gen}\|, \quad (4)$$

where $\mathcal{I}_{style2d}, \mathcal{I}_{gen}$ are the 2D stylized image from the 2D stylization module, and the stylized view from our pipeline respectively.

5 Experiments

Table 1: Here, we demonstrate the superior performance of our method using the short-term and long-term consistency metric. We compute the metric using a VGG backbone for both, RMSE and LPIPS[88], elaborated in Section 5.2 . We provide a qualitative evaluation to support visual superiority of our method against the mentioned baselines in Figure 4, and also provide videos for a comprehensive evaluation in the supplementary materials.

Method	LLFF[49]				TnT[36]			
	Short-Term		Long-Term		Short-Term		Long-Term	
	RMSE	LPIPS	RMSE	LPIPS	RMSE	LPIPS	RMSE	LPIPS
GS → AdaIN	0.122	0.132	0.153	0.190	0.050	0.014	0.107	0.054
AdaIN → GS	0.106	0.109	0.138	0.161	0.015	0.002	0.100	0.055
LSNV [26]	0.069	0.032	0.070	0.040	0.030	0.006	0.100	0.045
ARF [87]	0.039	0.019	0.095	0.083	0.012	0.001	0.101	0.048
Stylised-NeRF [28]	0.028	0.009	0.040	0.021	—	—	—	—
StyleRF [43]	0.039	0.017	0.095	0.067	0.024	0.006	0.130	0.070
Ours	0.019	0.005	0.032	0.014	0.015	0.002	0.083	0.040

In this section, we first present the implementation details of our method, followed by comparisons with existing methods and ablation studies.

5.1 Implementation Details

Our implementation is based on the 3DGS [33] framework. We closely follow the implementation of the method. We employ a warmup strategy, similar to Lee et al. [38], wherein we train 3DGS without our 2D stylization and the 3D Color modules. This is so that consistency is established in the initial iterations of the training and initialized Gaussians. We do this for the first 3000 iterations. Similar to 3DGS training, we allow the Gaussians to densify until 15000 iterations,

after which we fix the number of Gaussians in the scene. For the 2D stylization module, we use a pre-trained VGG model [72, 16] and AdaIN decoder [27]. We encode the input style images using a pre-trained image encoder [65]. We train an MLP after the warmup as described in “training regime” for a total of 150k iterations. The MLP is composed of 2 fully connected layers with 64 neurons each. For the MHG, we use a resolution of 16 levels, with each level having a feature vector of length 2. For optimization, we use the Adam optimizer [34] with a learning rate of $1e^{-4}$. We run all our experiments on a Nvidia A4500 GPU.

5.2 3D Stylization with Gaussian Splatting

We compare our method on stylizing 3D scenes against StylizedNerf[28, 24], LSNV[26], StyleRF[43], and artistic radiance fields(ARF)[87]. We would like to emphasize that StylizedNerf, StyleRF, and LSNV are more relevant to our method as the methods focus on generalizing to multiple styles. ARF is optimized on a given style per scene. Further, we compare with two additional baselines that we call Ada-GS and GS-Ada. In Ada-GS, we train 3DGS on 2D stylized images for each style. This setting resembles that of ARF. In GS-Ada, we apply AdaIN[27] on novel views rendered from a pre-trained 3DGS model.

Datasets Following existing methods [28, 43, 26], we stylize various real-world datasets, LLFF dataset [49], and Tanks and Temples (TnT) [36]. LLFF dataset consists of high-resolution forward-facing images of 3D scenes. We stylize *trex*, *fern*, and *room* from the LLFF dataset. We learn to stylize each scene over a dataset of approximately 20000 diverse style images sampled from the WikiArt dataset [68] consisting of paintings and popular artworks of various artists.

Quantitative Results 2D or 3D Stylization, in general, is a subjective task. Hence, no metrics exist that can effectively measure stylization. However, we our goal is to stylize images in a 3D consistent way. Therefore, similar to existing works [26, 28, 43], we measure short-term and long-term consistency. We use adjacent views for short-term and far-away views for long-term consistency. Similar to the evaluation in [43], we first take two stylized views. By using optical flow based methods[76, 58], we then warp one view into another before computing the masked LPIPS[88] and RMSE score. Many works such as [26] also refer to it as warped LPIPS and warped RMSE. Mathematically, they can be summarised as

$$\mathbb{E}_{wlips}(\mathcal{O}_v, \mathcal{O}_{v'}) = LPIPS(\mathcal{O}_v, \mathcal{M}_v(\mathcal{W}(\mathcal{O}_{v'}))) \quad (5)$$

and

$$\mathbb{E}_{wrmse}(\mathcal{O}_v, \mathcal{O}_{v'}) = RMSE(\mathcal{O}_v, \mathcal{M}_v(\mathcal{W}(\mathcal{O}_{v'}))), \quad (6)$$

where \mathcal{M}_v , \mathcal{O}_v , and \mathcal{W} are the masking, the rendered view, and the warping function respectively for two views v and v' .

We quantitatively compare our method with the baselines as shown in Table 1. It can be clearly seen that our method outperforms all other baselines

in both short-term and long-term consistency. We can see that simply applying 2D stylization to rendered novel views is not enough to ensure consistency, in accordance with [26], and hence Ada-GS is almost always more consistent than GS-ADA for both short-term and long-term views. Furthermore, simply adding 2D stylization to the input or outputs of existing scene representation approaches is not sufficient, as all other dedicated baselines perform better on average than GS-ADA and ADA-GS. We also find that methods utilising explicit scene representations, such as GSS and LSNV[26] perform better than radiance field based baselines. It is also important to observe that for StylizedNerf, the generated novel views suffer from high blur, over-smoothing and loss of details. Due to this, they obtain a very high consistency despite not being of reasonable quality as compared to all other methods.

Table 2: We perform a runtime analysis of our method vs other baselines. Due to our 3DGS backbone, we are able to generate stylized novel views approximately 4x faster than the second method, achieving an average of 157 FPS(Frames per second). Radiance fields in general have a high rendering time and suffer from slow training and testing time.

	LSNV	ARF	StylizedNerf	StyleRF	GSS
Rendered FPS	40	1	0.004	0.04	157

On running on the same hardware, GSS is also the fastest. Whereas all other methods take in the order of a few seconds to minutes to generate a single view, we obtain a rendering speed of roughly averaging **157 FPS**. Our method is also efficient in training and it requires roughly 4 hours to train for each scene on a standard Nvidia A4500 GPU.

Qualitative Results Along with the quantitative results, we demonstrate the performance of our method with the baselines with a qualitative evaluation as shown in Figure 4. It can be seen that not only our method preserves content details better, it also is more faithful to the style images. For instance, while StyleRF[43] is great at preserving geometric details, it does not adequately transfer the style image features onto the novel views. On the other hand, ARF[87] is aggressive at transferring the style image features onto the scene, and hence often leads to a loss in the information. It also has an affinity to focus more on the central object and does not transfer the style to far-away objects in the scene. It is worth noting that ARF optimizes each scene for every style image. We also observe that StylisedNerf[28] produces blurry views while LSNV also struggles to retain fine details in the stylized views.

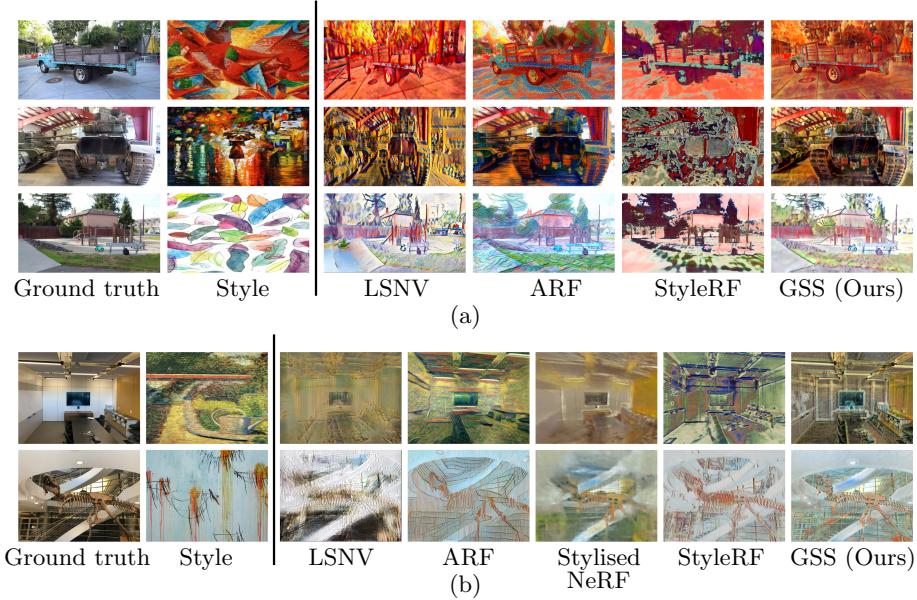


Fig. 4: We provide a detailed qualitative comparison of our method against the baselines detailed in Section 5 on the (a) TnT and (b) LLFF datasets. Our method achieves a highly accurate stylization based on the input style. While methods such as ARF obtain a better texture, it is attributed to the fact that it is optimized separately for each style. StylizedNerf[28] produces images that suffer from over smoothing and blurriness, while StyleRF fails to grasp the accurate style color. On the other hand, our proposed method is able to retain high details present in the unstyled view while transferring the adequate texture and colors of the style image for both, indoor and outdoor real-world datasets.

6 Ablation Studies

To study the effects of various components of our method, we perform several experiments and report their findings below.

Effect of Joint Training Contrary to previous NeRF-based baselines such as StylizedNerf[28], we employ a joint-training regime where the 3D Gaussians of each scene are trained in conjunction with the 3D color module. To study the effect of this regime, we perform an ablation where we employ pre-trained 3D Gaussians. These Gaussians were trained on unstylized images, and were fixed throughout the training process while the 3D Color module was trained from scratch. The stylization results from pretrained Gaussians appear blurry while our method retains significantly higher amount of details. Furthermore, since the geometric parameters such as opacity are fixed in the pretrained model, the final style imparted onto the rendered view lacks coherence.

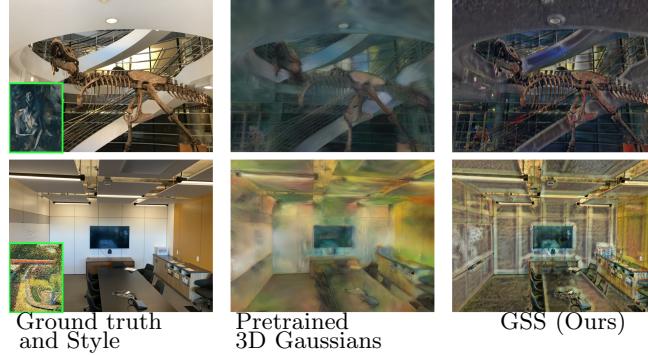


Fig. 5: We show the effect of deploying a joint-training regime of training the 3D Gaussians in conjunction with the 3D Color module. Having a joint training in an end-to-end fashion helps to preserve key details and geometry in the rendered stylized novel view.

Train on stylized images vs stylizing novel renderings Additionally, we quantitatively and qualitatively evaluate the performance of our synthetic baselines, namely GS-Ada and Ada-GS in Figure 6 and table 1. We elaborate on these two baselines in Section 5.2. We observe that GS-Ada is the least consistent as the rendered novel views are stylized independently of each other using AdaIN [27], therefore, have no notion of consistency. While Ada-GS performs better, it has a major drawback, i.e. the method needs to be trained from scratch for every new style image, thus limiting its usability, especially for practical applications.

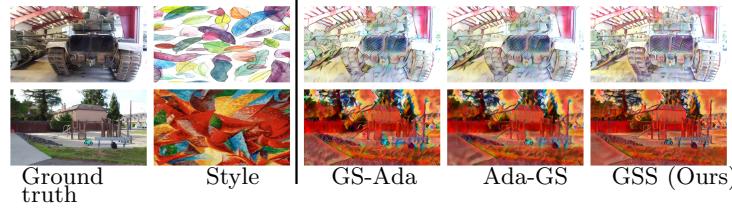


Fig. 6: In addition to providing quantitative results in Table 1, we provide the qualitative results showing the distorted geometry when we apply 2D stylization on rendered novel views(GS-Ada). We also observe that training vanilla 3DGS on 2D Stylized views(Ada-GS) preserves better details than GS-Ada. However, our method provides the most geometrically accurate renderings while being truthful to the queried style.

Style Interpolation We perform style interpolation by taking four different style images at test time, and obtaining the resulting style latent by linearly interpolating between the different styles. We show the results in Section 6. The ablation shows the capability of our model to generate results for not only the



Fig. 7: We interpolate between the latent vectors of the style images. The four style images are shown at the four corners of the image above and were chosen so they're highly diverse and accurately depict the differences in the predicted RGBs for each Gaussian from our model.

given style images, but also a mixture of such styles. It demonstrates that our model learns the mapping from style inputs to predicted RGB values for each 3D Gaussian in a meaningful way, and thus provides increased generalisability.

7 Conclusion

In this paper, we presented a novel method to stylize complex 3D scenes that are spatially consistent. Contrary to a majority of existing works, once trained, our method is capable to take unseen input scenes at inference time and produce novel views in real-time. By leveraging a multi-resolution hash grid and a tiny MLP, we are able to accurately generate the stylized colors of each 3D Gaussian present in a scene. Since we only do one forward pass through the 3D color module, we are able to generate novel views at around **150FPS**. We exhibit that GSS produces superior results by the use of quantitative and qualitative results, thus making GSS suitable for many practical applications.

Bibliography

- [1] Barron, J.T., Mildenhall, B., Tancik, M., Hedman, P., Martin-Brualla, R., Srinivasan, P.P.: Mip-nerf: A multiscale representation for anti-aliasing neural radiance fields (2021)
- [2] Cao, X., Wang, W., Nagao, K., Nakamura, R.: Psnet: A style transfer network for point cloud stylization on geometry and color. In: Proceedings of the IEEE/CVF Winter Conference on Applications of Computer vision. pp. 3337–3345 (2020)
- [3] Chen, A., Xu, Z., Geiger, A., Yu, J., Su, H.: Tensorf: Tensorial radiance fields. In: European Conference on Computer Vision (ECCV) (2022)
- [4] Chen, D., Liao, J., Yuan, L., Yu, N., Hua, G.: Coherent online video style transfer. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 1105–1114 (2017)
- [5] Chen, S.E., Williams, L.: View Interpolation for Image Synthesis. In: SIGGRAPH (1993)
- [6] Chen, X., Zhang, Y., Wang, Y., Shu, H., Xu, C., Xu, C.: Optical flow distillation: Towards efficient and stable video style transfer. In: Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part VI 16. pp. 614–630. Springer (2020)
- [7] Chen, Y., Yuan, Q., Li, Z., Liu, Y., Wang, W., Xie, C., Wen, X., Yu, Q.: Upst-nerf: Universal photorealistic style transfer of neural radiance fields for 3d scene (2022)
- [8] Chiang, P.Z., Tsai, M.S., Tseng, H.Y., sheng Lai, W., Chiu, W.C.: Styling 3d scene via implicit representation and hypernetwork (2022)
- [9] Chibane, J., Bansal, A., Lazova, V., Pons-Moll, G.: Stereo radiance fields (srf): Learning view synthesis for sparse views of novel scenes. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 7911–7920 (2021)
- [10] Debevec, P.E., Taylor, C.J., Malik, J.: Modeling and rendering architecture from photographs: A hybrid geometry-and image-based approach. In: SIGGRAPH (1996)
- [11] Dosovitskiy, A., Fischer, P., Ilg, E., Hausser, P., Hazirbas, C., Golkov, V., Van Der Smagt, P., Cremers, D., Brox, T.: Flownet: Learning optical flow with convolutional networks. In: Proceedings of the IEEE international conference on computer vision. pp. 2758–2766 (2015)
- [12] Efros, A.A., Freeman, W.T.: Image quilting for texture synthesis and transfer. In: Seminal Graphics Papers: Pushing the Boundaries, Volume 2. pp. 571–576 (2023)
- [13] Efros, A.A., Leung, T.K.: Texture synthesis by non-parametric sampling. In: Proceedings of the seventh IEEE international conference on computer vision. vol. 2, pp. 1033–1038. IEEE (1999)
- [14] Fang, J., Wang, J., Zhang, X., Xie, L., Tian, Q.: Gaussianeditor: Editing 3d gaussians delicately with text instructions (2023)

- [15] Gao, C., Gu, D., Zhang, F., Yu, Y.: Reconet: Real-time coherent video style transfer network. In: Computer Vision–ACCV 2018: 14th Asian Conference on Computer Vision, Perth, Australia, December 2–6, 2018, Revised Selected Papers, Part VI 14. pp. 637–653. Springer (2019)
- [16] Gatys, L.A., Ecker, A.S., Bethge, M.: Image style transfer using convolutional neural networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 2414–2423 (2016)
- [17] Gortler, S.J., Grzeszczuk, R., Szeliski, R., Cohen, M.F.: The Lumigraph. SIGGRAPH **96**(30) (1996)
- [18] Guangcong, Chen, Z., Loy, C.C., Liu, Z.: Sparsenerf: Distilling depth ranking for few-shot novel view synthesis. IEEE/CVF International Conference on Computer Vision (ICCV) (2023)
- [19] Gupta, A., Johnson, J., Alahi, A., Fei-Fei, L.: Characterizing and improving stability in neural style transfer. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 4067–4076 (2017)
- [20] Haque, A., Tancik, M., Efros, A.A., Holynski, A., Kanazawa, A.: Instruct-nerf2nerf: Editing 3d scenes with instructions. arXiv preprint arXiv:2303.12789 (2023)
- [21] Hedman, P., Philip, J., Price, T., Frahm, J.M., Drettakis, G., Brostow, G.: Deep Blending for Free-Viewpoint Image-Based Rendering. In: SIGGRAPH Asia (2018)
- [22] Heigl, B., Koch, R., Pollefeys, M., Denzler, J., Van Gool, L.: Plenoptic Modeling and Rendering from Image Sequences taken by a Hand Held Camera. In: GCPR (1999)
- [23] Höllerin, L., Johnson, J., Nießner, M.: Stylemesh: Style transfer for indoor 3d scene reconstructions. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 6198–6208 (2022)
- [24] Hu, S.M., Liang, D., Yang, G.Y., Yang, G.W., Zhou, W.Y.: Jittor: a novel deep learning framework with meta-operators and unified graph execution. Science China Information Sciences **63**(222103), 1–21 (2020)
- [25] Huang, H., Wang, H., Luo, W., Ma, L., Jiang, W., Zhu, X., Li, Z., Liu, W.: Real-time neural style transfer for videos. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 783–791 (2017)
- [26] Huang, H.P., Tseng, H.Y., Saini, S., Singh, M., Yang, M.H.: Learning to stylize novel views. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 13869–13878 (2021)
- [27] Huang, X., Belongie, S.: Arbitrary style transfer in real-time with adaptive instance normalization. In: Proceedings of the IEEE international conference on computer vision. pp. 1501–1510 (2017)
- [28] Huang, Y.H., He, Y., Yuan, Y.J., Lai, Y.K., Gao, L.: Stylizednerf: consistent 3d scene stylization as stylized nerf via 2d-3d mutual learning. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 18342–18352 (2022)
- [29] Hwang, I., Kim, H., Kim, Y.M.: Text2scene: Text-driven indoor scene stylization with part-aware details (2023)

- [30] Ioannou, E., Maddock, S.: Depth-aware neural style transfer using instance normalization. arXiv preprint arXiv:2203.09242 (2022)
- [31] Johnson, J., Alahi, A., Fei-Fei, L.: 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. pp. 694–711. Springer (2016)
- [32] Jung, H., Nam, S., Sarafianos, N., Yoo, S., Sorkine-Hornung, A., Ranjan, R.: Geometry transfer for stylizing radiance fields (2024)
- [33] Kerbl, B., Kopanas, G., Leimkühler, T., Drettakis, G.: 3d gaussian splatting for real-time radiance field rendering. ACM Transactions on Graphics **42**(4) (July 2023), <https://repo-sam.inria.fr/fungraph/3d-gaussian-splatting/>
- [34] Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization (2017)
- [35] Knapitsch, A., Park, J., Zhou, Q.Y., Koltun, V.: Tanks and temples: Benchmarking large-scale scene reconstruction. ACM Transactions on Graphics **36**(4) (2017)
- [36] Knapitsch, A., Park, J., Zhou, Q.Y., Koltun, V.: Tanks and temples: Benchmarking large-scale scene reconstruction. ACM Transactions on Graphics (ToG) **36**(4), 1–13 (2017)
- [37] Kwatra, V., Schödl, A., Essa, I., Turk, G., Bobick, A.: Graphcut textures: Image and video synthesis using graph cuts. ACM transactions on graphics (tog) **22**(3), 277–286 (2003)
- [38] Lee, J.C., Rho, D., Sun, X., Ko, J.H., Park, E.: Compact 3d gaussian representation for radiance field. arXiv preprint arXiv:2311.13681 (2023)
- [39] Levoy, M., Hanrahan, P.: Light field rendering. In: SIGGRAPH (1996)
- [40] Li, C., Wand, M.: Precomputed real-time texture synthesis with markovian generative adversarial networks. In: Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part III 14. pp. 702–716. Springer (2016)
- [41] Li, W., Wu, T., Zhong, F., Oztireli, C.: Arf-plus: Controlling perceptual factors in artistic radiance fields for 3d scene stylization (2023)
- [42] Li, Z., Müller, T., Evans, A., Taylor, R.H., Unberath, M., Liu, M.Y., Lin, C.H.: Neuralangelo: High-fidelity neural surface reconstruction. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2023)
- [43] Liu, K., Zhan, F., Chen, Y., Zhang, J., Yu, Y., Saddik, A.E., Lu, S., Xing, E.: Stylerf: Zero-shot 3d style transfer of neural radiance fields (2023)
- [44] Liu, X.C., Cheng, M.M., Lai, Y.K., Rosin, P.L.: Depth-aware neural style transfer. In: Proceedings of the symposium on non-photorealistic animation and rendering. pp. 1–10 (2017)
- [45] Martin-Brualla, R., Radwan, N., Sajjadi, M.S., Barron, J.T., Dosovitskiy, A., Duckworth, D.: Nerf in the wild: Neural radiance fields for unconstrained photo collections. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 7210–7219 (2021)
- [46] Mescheder, L., Oechsle, M., Niemeyer, M., Nowozin, S., Geiger, A.: Occupancy networks: Learning 3d reconstruction in function space. In: Proceedings IEEE Conf. on Computer Vision and Pattern Recognition (CVPR) (2019)

- [47] Metzer, G., Richardson, E., Patashnik, O., Giryes, R., Cohen-Or, D.: Latent-nerf for shape-guided generation of 3d shapes and textures (2022)
- [48] Michel, O., Bar-On, R., Liu, R., Benaim, S., Hanocka, R.: Text2mesh: Text-driven neural stylization for meshes (2021)
- [49] Mildenhall, B., Srinivasan, P.P., Ortiz-Cayon, R., Kalantari, N.K., Ramamoorthi, R., Ng, R., Kar, A.: Local light field fusion: Practical view synthesis with prescriptive sampling guidelines. ACM Transactions on Graphics (TOG) **38**(4), 1–14 (2019)
- [50] Mildenhall, B., Srinivasan, P.P., Tancik, M., Barron, J.T., Ramamoorthi, R., Ng, R.: Nerf: Representing scenes as neural radiance fields for view synthesis. Communications of the ACM **65**(1), 99–106 (2021)
- [51] Moreau, A., Song, J., Dhamo, H., Shaw, R., Zhou, Y., Pérez-Pellitero, E.: Human gaussian splatting: Real-time rendering of animatable avatars (2023)
- [52] Mu, F., Wang, J., Wu, Y., Li, Y.: 3d photo stylization: Learning to generate stylized novel views from a single image. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 16273–16282 (2022)
- [53] Müller, T.: tiny-cuda-nn (4 2021), <https://github.com/NVlabs/tiny-cuda-nn>
- [54] Müller, T., Evans, A., Schied, C., Keller, A.: Instant neural graphics primitives with a multiresolution hash encoding. ACM Transactions on Graphics (ToG) **41**(4), 1–15 (2022)
- [55] Nguyen-Ha, P., Huynh, L., Rahtu, E., Matas, J., Heikkil, J., et al.: Cascaded and generalizable neural radiance fields for fast view synthesis. IEEE Transactions on Pattern Analysis and Machine Intelligence (2023)
- [56] Nguyen-Phuoc, T., Liu, F., Xiao, L.: Snerf: Stylized neural implicit representations for 3d scenes (2022)
- [57] Niemeyer, M., Barron, J.T., Mildenhall, B., Sajjadi, M.S.M., Geiger, A., Radwan, N.: Regnerf: Regularizing neural radiance fields for view synthesis from sparse inputs (2021)
- [58] Niklaus, S., Liu, F.: Softmax splatting for video frame interpolation (2020)
- [59] Niklaus, S., Mai, L., Yang, J., Liu, F.: 3d ken burns effect from a single image (2019)
- [60] Pang, H., Zhu, H., Kortylewski, A., Theobalt, C., Habermann, M.: Ash: Animatable gaussian splats for efficient and photoreal human rendering (2023)
- [61] Pang, H.W., Hua, B.S., Yeung, S.K.: Locally stylized neural radiance fields (2023)
- [62] Park, J.J., Florence, P., Straub, J., Newcombe, R., Lovegrove, S.: Deepsdf: Learning continuous signed distance functions for shape representation (2019)
- [63] Peng, S., Niemeyer, M., Mescheder, L., Pollefeys, M., Geiger, A.: Convolutional occupancy networks. In: Computer Vision – ECCV 2020. Lecture Notes in Computer Science, 12348, vol. 3, pp. 523–540. Springer, Cham (Aug 2020)
- [64] Qian, S., Kirschstein, T., Schoneveld, L., Davoli, D., Giebenhain, S., Nießner, M.: Gaussianavatars: Photorealistic head avatars with rigged 3d gaussians (2023)

- [65] Radford, A., Kim, J.W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., et al.: Learning transferable visual models from natural language supervision. In: International conference on machine learning. pp. 8748–8763. PMLR (2021)
- [66] Reiser, C., Peng, S., Liao, Y., Geiger, A.: Kilonerf: Speeding up neural radiance fields with thousands of tiny mlps (2021)
- [67] Riegler, G., Koltun, V.: Free view synthesis. In: Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XIX 16. pp. 623–640. Springer (2020)
- [68] Saleh, B., Elgammal, A.: Large-scale classification of fine-art paintings: Learning the right metric on the right feature (2015)
- [69] Schwarz, K., Sauer, A., Niemeyer, M., Liao, Y., Geiger, A.: Voxgraf: Fast 3d-aware image synthesis with sparse voxel grids (2022)
- [70] Seitz, S.M., Dyer, C.R.: View Morphing. SIGGRAPH (1996)
- [71] Shin, D., Ren, Z., Suderth, E.B., Fowlkes, C.C.: 3d scene reconstruction with multi-layer depth and epipolar transformers (2019)
- [72] Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556 (2014)
- [73] Sitzmann, V., Thies, J., Heide, F., Nießner, M., Wetzstein, G., Zollhöfer, M.: Deepvoxels: Learning persistent 3d feature embeddings. In: CVPR (2019)
- [74] Sitzmann, V., Zollhöfer, M., Wetzstein, G.: Scene Representation Networks: Continuous 3D-Structure-Aware Neural Scene Representations. In: NeurIPS (2019)
- [75] Song, H., Choi, S., Do, H., Lee, C., Kim, T.: Blending-nerf: Text-driven localized editing in neural radiance fields (2023)
- [76] Teed, Z., Deng, J.: Raft: Recurrent all-pairs field transforms for optical flow (2020)
- [77] Tulsiani, S., Gupta, S., Fouhey, D., Efros, A.A., Malik, J.: Factoring shape, pose, and layout from the 2d image of a 3d scene (2018)
- [78] Ulyanov, D., Vedaldi, A., Lempitsky, V.: Improved texture networks: Maximizing quality and diversity in feed-forward stylization and texture synthesis. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 6924–6932 (2017)
- [79] Wang, C., Jiang, R., Chai, M., He, M., Chen, D., Liao, J.: Nerf-art: Text-driven neural radiance fields stylization (2022)
- [80] Wang, W., Yang, S., Xu, J., Liu, J.: Consistent video style transfer via relaxation and regularization. IEEE Transactions on Image Processing **29**, 9125–9139 (2020)
- [81] Wei, L.Y., Levoy, M.: Fast texture synthesis using tree-structured vector quantization. In: Proceedings of the 27th annual conference on Computer graphics and interactive techniques. pp. 479–488 (2000)
- [82] Wiles, O., Gkioxari, G., Szeliski, R., Johnson, J.: Synsin: End-to-end view synthesis from a single image (2020)
- [83] Wimbauer, F., Yang, N., Rupprecht, C., Cremers, D.: Behind the scenes: Density fields for single view reconstruction (2023)

- [84] Xu, S., Li, L., Shen, L., Lian, Z.: Desrf: Deformable stylized radiance field. In: 2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW) pp. 709–718 (2023), <https://api.semanticscholar.org/CorpusID:259237546>
- [85] Yu, A., Fridovich-Keil, S., Tancik, M., Chen, Q., Recht, B., Kanazawa, A.: Plenoxels: Radiance fields without neural networks (2021)
- [86] Yu, A., Ye, V., Tancik, M., Kanazawa, A.: pixelnerf: Neural radiance fields from one or few images (2021)
- [87] Zhang, K., Kolkin, N., Bi, S., Luan, F., Xu, Z., Shechtman, E., Snavely, N.: Arf: Artistic radiance fields (2022)
- [88] Zhang, R., Isola, P., Efros, A.A., Shechtman, E., Wang, O.: The unreasonable effectiveness of deep features as a perceptual metric. In: CVPR (2018)
- [89] Zhang, Y., He, Z., Xing, J., Yao, X., Jia, J.: Ref-npr: Reference-based non-photorealistic radiance fields for controllable scene stylization (2023)
- [90] Zhang, Z., Liu, Y., Han, C., Pan, Y., Guo, T., Yao, T.: Transforming radiance field with lipschitz network for photorealistic 3d scene stylization (2023)
- [91] Zheng, S., Zhou, B., Shao, R., Liu, B., Zhang, S., Nie, L., Liu, Y.: Gps-gaussian: Generalizable pixel-wise 3d gaussian splatting for real-time human novel view synthesis (2023)
- [92] Zitnick, C.L., Kang, S.B., Uyttendaele, M., Winder, S., Szeliski, R.: High-quality video view interpolation using a layered representation. SIGGRAPH **23**(3) (2004)

A Qualitative Scene Video Results

We provide videos of the stylized scenes to demonstrate the superior performance of our proposed method against the baselines. It can be observed that our method, not only maintains geometric consistency, but is also able to successfully able to transfer the required style information to the rendered views.

B Additional Comparisons with ARF

We further qualitatively compare our method against ARF[?]. As explored in the main text, ARF has great ability to perform texture transfer, however it suffers from stylizing in the entire space, especially in open outdoor scenes such as the truck scene explored in Figure 1. The stylization is inconsistent for instance, on the floor or non existing on far away objects such as the background trees or the sky. On the contrary, our method is capable of stylizing far distant objects including the sky while maintaining consistency on large areas such as the ground, while being faithful to the original geometry. It is also worth acknowledging that unlike ARF, we do not need to retrain the scene for each new style, thus making our method more versatile.



Fig. 1: Here we show additional qualitative ablations with ARF[?]. We find that despite its aggressive texture transfer abilities, ARF fails to create consistent texture across the entire surface and totally misses stylizing on far away objects and backgrounds. GSS on the other hand, is able to perform stylization uniformly and consistently for distant objects and background such as the floor or buildings. We refer the reader to Appendix A for more evidence to support our comparison.