Generating aesthetic and novel scenes from images using style transfer on NeRF

Caleb Teeter Chirag Totla Jerry Zeng Diptyaroop Maji

University of Massachusetts Amherst

Abstract

A Neural Radiance Field (NeRF) is a technique to synthesize novel 3D views from a set of 2D images. NeRFs have many important use cases, ranging from medical imaging to creating virtual reality. Recently, there has been an increased focus on stylizing synthesized 3D scenes due to the high demand for new visual 3D content in applications like animated movies and Mixed Reality games. In this project, we generated stylized novel views by using style transfer on NeRF-generated images. Since style transfer can be applied both before and after generating novel views, we compared both designs. Our analysis shows that applying style transfer after NeRF produces much better and crisper stylized views than applying it before NeRF. Using the better design, we generated stylized novel views for three NeRF synthetic datasets using three different styles and evaluated the stylized views both qualitatively and quantitatively. We also modified a popular NeRF implementation that uses dense voxel grids to generate novel views to improve its training time at high resolution. Our implementation reduces the training time of NeRF by 70.6% (3.4× speedup) at 256^3 *resolution while only degrading the PSNR by 6.7%. Additionally, we also created a new "bottle" dataset to test our NeRF implementation on new data, and was able to generate good quality novel views on the new data (PSNR 27.39). Our project is available at* **https://github.com/diptyaroop/CS674_Spring24**.

1. Introduction

The demand for stylized 3D content has been increasing in recent years due to surge of applications like AR/VR games, animated 3D movies, photo-realistic image creation, etc. Consequently, there has been numerous works on generating such content recently. To generate stylized 3D content, researchers have mainly leveraged two techniques: (1) Neural Radiance Fields (NeRF) [1], and (2) Style transfer. NeRF is a technology that is capable of generating novel views from a given set of images and is well known for being able to achieve photo-realistic renderings of scenes. Style transfer on the other hand allows for any content image to be stylized to a certain style, allowing for distinct art styles to be applied to images. The intersection between these two techniques usually produces good-quality 3D visuals with distinct art styles.

A prominent paper on generating stylized novel views is StyleRF [12], which integrates style transfer directly into the volumetric rendering process. Inspired by this paper, in this project we apply style transfer on NeRF to generate stylized novel views of a scene given a set of training images and camera poses of the scene, and a specific art style. We implement these two components and the design in our own way. Due to resource constraints, we looked at NeRF implementations that enable training with low GPU resources and less time. DVGO [19] is an enhancement of NeRF that essentially uses dense voxel grids to increase the run speed and decrease computational requirements. Another project is Plenoxels [6], which uses a sparse voxel grid. Both of these projects aim to reduce the training time without reducing the quality of NeRF. These two projects inspired us to create a project that also attempts to reduce the runtime, and we decide to base our NeRF implementation on DVGO [19]. However, DVGO has an inherent limitation in that it does not scale well to higher resolutions due to its use of the dense voxel grid data structure. So, we modify DVGO by replacing its dense voxel grid using multi-resolution hash encoding technique introduced in [16]. Our Style Transfer implementation is based on StyTR2 [4], a transformer-based model that uses two seperate transformer encoders to generate encodings for both the style and content images.

With these NeRF and style transfer architectures we generate stylized novel views. Since style transfer can be applied both before and after NeRF , we propose both the designs and compare the performance of the two, before generating novel views with the better design choice and evaluating the stylized generated views.

Our contributions. Specifically, our contributions are as follows:

1. Proposing and comparing two design choices (NeRF–ST and ST–NeRF) for generating stylized novel views. Our

comparison shows that applying style transfer after genearting new images using NeRF is a better design than appying NeRF on stylized images.
2. Generating stylized novel views given a set of training images, camera poses and a specific style. We applied our NeRF–ST design on three NeRF-synthetic datasets and three styles and evaluated the generated views both qualitatively, and quantitatively using the "effectiveness" metric defined by [20]. We see that our method generates good quality stylized novel views.
3. Modifying the popular DVGO [19] implementation of NeRF and improving its training time for high resolution scenes. Our hash-encoding modification reduces the training time by 70.6% (3.4× speedup) while only degrading image quality by 6.7%.
4. Creating a new dataset to test our modified NeRF implementation (called ***mrhe-NeRF***) on new data.

Our project is available at **https://github.com/diptyaroop/CS674_Spring24**.

2. Background and related work

In this section, we provide a brief background on the NeRF and style transfer components along with a concise review of the prior work in this domain.

2.1. Neural radiance fields (NeRF)

Given a set of camera poses and source images of a scene, NeRF tries to solve the problem of novel view synthesis. More specifically, NeRF uses synthesizes a target image for an arbitrary camera pose for the scene. Mathematically, NeRF takes a 5D input (x, y, z, θ, ϕ) for a scene, where (x, y, z) is the location and (θ, ϕ) is the viewing direction, and generates a 4D output (r, g, b, σ), where (r, g, b) is the emitted color and σ is the volume density at that location.

Mildenhall et al. [14] were the first to solve this problem using a coordinate-based MLP and differential volume rendering to predict (r, g, b, σ) by training on a set of locations and viewing directions. Since their seminal work, there has been a huge influx of NeRF related papers in the field, each trying to address some limitations of the original paper or extending the technique to more diverse scenarios. Since the original NeRF technique takes a long time to train and hence may not be feasible to run in real-time or with low resources, many recent works focus on speeding up NeRF training [3, 6, 16, 19]. Some of the prominent works are DirectVoxGO [19] and Plenoxels [6], which replaced the MLP with a voxel grid to improve the training time significantly. InstantNGP [16] used a multi-resolution hash encoding along with other techniques like fully-fused MLPs to reduce the training time to a few seconds. While [6, 16] require custom CUDA kernels, TensoRF [3] factorized radiance fields to compact components to achieve similar results with standard PyTorch implementation and significantly less memory footprint than [6, 19].

In addition to these papers, there are a lot of papers on improving the inference speed of NeRF [10, 21], applying NeRF on unconstrained images [7, 13], NeRF for 4D view synthesis [8, 9], etc., in recent years. We do not go in-depth about those as it is beyond the scope of this project.

2.2. Style transfer

Style transfer is also a well-studied problem, with numerous works in recent years. Style transfer takes an input style image and an input content image and outputs an image that effectively conveys the content of the content image in the style of the style image. While ST does not have a clear state-of-the-art model because of the lack of standard qualitative metrics, there are some recent models that have notable innovations. StyleFlow [5] uses an invertible model that projects into the style feature space on the forward pass, applying style, then projects back into the content space on the backward pass, resulting in a stylized image. StyTr2 [4] is a transformer based model that uses two seperate transformer encoders to generate encodings for both the style image and the content image. A transformer-decoder then decodes the content embeddings into the style of the style image. The stylized encodings are then upsampled into an output image from the model.

There are numerous other style transfer models, however this paper is more focused on the combination of ST with NeRF. Therefore, we will leave comparing other ST models in NeRF applications to future work.

2.3. Stylized views

Combining a NeRF model with a ST model, the network can output novel views of a scene in a given style. Although much of ST is subjective, the model that seems to be state of the art is StyleRF [12]. StyleRF generates high quality results by applying style transfer directly to the feature space of NeRF. By working directly with the radiance field, StyleRF has access to a better representation of the 3D geometry than the output images allow for. Working directly with the radiance field also promotes consistency across the generated images.

A handful of other past papers apply 2D ST models to the NeRF output. The focus of this paper is on the viability of applying ST before NeRF. Therefore, we aimed to keep the approaches as comparable as possible, by utilizing the techniques of older papers and using the same models regardless of the order of NeRF and ST.

3. Method

Our project has broadly two components — the NeRF component and the ST component. In this section, we describe each component in detail along with our design choices.

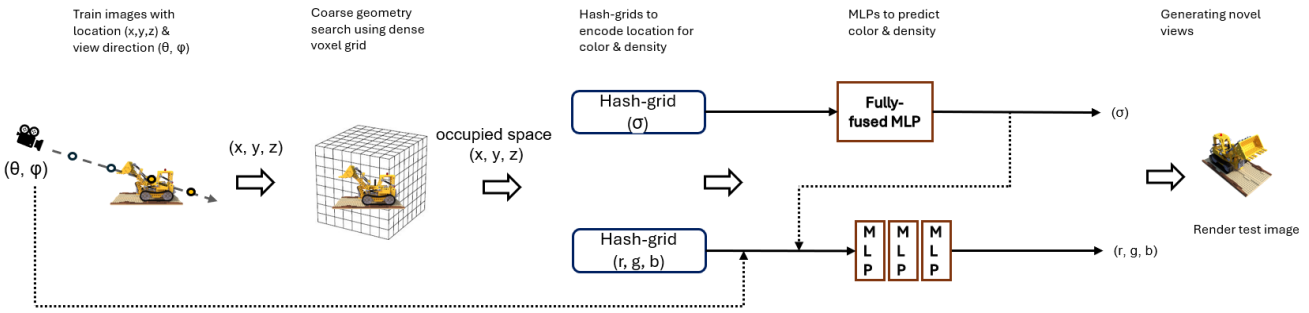

Figure 1. *Our **mrhe-NeRF** architecture. First, we create a coarse geometry using dense voxel grids (similar to DVGO). Then, we use that geometry as a reference to select necessary rays in the fine stage. In the fine stage, we use 2 hash-grids to encode the 3D locations. Outputs from the hash-grids are fed to the MLPs (along with other relevant inputs) to finally predict the 4D color and density output.*

3.1. NeRF component (*mrhe-NeRF*)

Our NeRF component (which we call ***mrhe-NeRF***) is based on the architecture used in DirectVoxGO (DVGO) [19]. DVGO uses a two-stage architecture to speed up the training speed of NeRF . In the first stage, they use a low-resolution dense voxel grid to fix the coarse geometry of the object in the scene. This is done to reduce the number of query rays to use since most areas in a scene are typically unoccupied and hence, querying rays through those parts is redundant. Once the coarse geometry is fixed, DVGO runs selected query rays in the second fine stage to get the NeRF output. In the fine stage, DVGO uses a hybrid of a dense voxel grid followed by a shallow MLP. DVGO also progressively scales up the resolution of the dense voxel grid to improve the quality of the generated images. However, since DVGO uses a dense voxel grid, it is inherently not scalable to higher resolutions since the computation time increases in a cubic manner with an increase in the grid resolution. Consequently, the authors limit the resolution to 160^3 in their paper.

In this project, we overcome this limitation by using multi-resolution hash encoding introduced by Muller et al. [16]. Multi-resolution hash encoding is a type of parametric encoding that enables the algorithm to scale up in resolution arbitrarily, as shown in [16]. Our NeRF implementation is also two-stage, and our coarse stage is exactly the same as in DVGO. However, once the coarse geometry is fixed, we replace the dense voxel grid in DVGO with two hash-grids. One hash-grid encodes the 3D location information and feeds it to a fully-fused MLP that predicts the density. The other hash-grid encodes the 3D location and feeds it to an MLP that predicts the color at that location. The color-MLP also takes in the positionally encoded view direction information and the density output obtained from the density-MLP to predict the (r, g, b) values. The density-MLP has one hidden layer with 64 neurons, while the color MLP has 3 hidden layers, each having 128 neurons. Figure 1 shows the architecture of ***mrhe-NeRF*** .

In the fine stage, we skip the known free space using the coarse geometry obtained and only pass the necessary rays (rays hitting the object) to speed up the training process. We keep the dense voxel grid information obtained in the coarse stage and progressively scale the dense grid to 160^3 (similar to DVGO) to get a more accurate estimation of necessary rays. We found that doing this improves the model performance over not scaling the coarse dense voxel grid. Note that while we limit the upper resolution of the dense voxel grid to avoid longer training times, the resolution of the hash-grids and, hence, the scene, can be scaled arbitrarily to improve the quality of the generated images. We scale up to 256^3 resolution in our experiments.

3.2. Style transfer component

The ST component of our model uses StyTr2 [4]. The authors introduce a transformer-based model designed specifically for image style transfer, highlighting its advantages over conventional CNN architectures commonly employed in feature extraction and style representation tasks. They underscore a critical limitation of CNNs: their struggle to effectively capture long-range dependencies without an extensive layer depth. However, they also address the inherent trade-off associated with deeper networks, namely the potential loss of feature resolution and fine details. This necessitates a nuanced approach to model design and evaluation in the pursuit of optimal image stylization outcomes. An investigation by An et al.[2] elucidated that CNN-based style transfer models tend to exhibit a bias towards content representation, resulting in inadvertent content leakage from the styling images. StyTr2 introduces an innovative content-aware positional encoding scheme (CAPE), aimed at enhancing image style transfer processes. CAPE dynamically learns positional encodings based on semantic features extracted from images, allowing for adaptive expansion to accommodate varying image sizes. The model employs two distinct encoders—one for processing content images and another for style images. The content image encoder incorporates patch embeddings alongside positional encodings from the CAPE module, whereas the style image

encoder solely utilizes patch embeddings. Subsequently, a decoder integrates content and style features in a regressive fashion, followed by a 3-layer CNN decoder for upsampling and generating the final output image. This comprehensive architecture enables effective fusion of content and style while preserving fine-grained details in the stylized output. The loss function is a combined loss function balancing the content loss vs. style loss for the output image. For more specific details on the model and loss function please refer to the original StyTr2 [4] paper.

The model was trained utilizing the MS-COCO [11] dataset, leveraging the Validation split of 2017, comprising 5,000 images for content, and a manually collected WikiArt [18] images dataset. Training was conducted over 25,000 iterations, with a batch size of 6. The training process took place on Kaggle, benefiting from the platform's provision of free access to 32 GB VRAM resources.

3.3. Design choices

In this project, we propose two design choices. Since the final goal is to produce novel stylized images, we posit that this can be done in the following two ways:

1. **NeRF–ST**: Given a set of images and view directions for a scene, we first generate novel views using NeRF. Once these views are generated, we input the set of generated images to our style transfer component to stylize those novel views.
2. **ST–NeRF**: Given a set of images, we stylize the view first by applying style transfer to these images. Then, we use these stylized views as inputs to our NeRF component to generate novel stylized views. It may be noted that the view directions for an image is not affected by stylization. Hence, we can use the same view directions.

4. Evaluation

In this section, we compare our NeRF implementation and the vanilla DVGO and show the comparison between NeRF–ST and ST–NeRF. We evaluate the ST component both qualitatively and quantitatively. We use the "effectiveness" metric defined by [20] for quantitative evaluation.

4.1. Experimental setup and methodology

We evaluated the NeRF implementations on a commodity laptop with NVidia RTX 3050 GPU having 6 GB memory. In this project, we only used the NeRF synthetic dataset. Specifically, we used the lego, chair, and ficus dataset for our experiments. We also created a new "bottle" dataset to test our implementation on new data (refer Sec. 4.5). For all the experiments, there were a total of 400 images, with the train, validation, and test split being 1:1:2. For style transfer, we used three styles — The Starry Night by Vincent van Gogh, Antibes in the Morning by Claude Monet and Don Quixote and Sancho Setting Out by Gustave Dore.

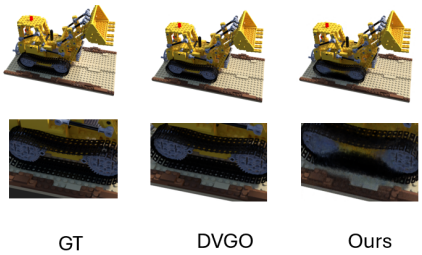

Figure 2. *Qualitative comparison of images produced by **mrhe-NeRF** and DVGO*

4.2. *mrhe-NeRF* results

First, we show the performance of our NeRF implementation and compare it with vanilla DVGO. Figure 2 shows a qualitative comparison of both methods. We also provide the ground truth for reference. ***mrhe-NeRF*** performs slightly worse than DVGO, as evident when zooming into the lego figure. Table 1 shows the quantitative comparison of PSNR values and training time rounded up to the nearest minute, where a higher PSNR means a better image. We see that the PSNR of ***mrhe-NeRF*** is 6.7% less than that on average but achieves an average reduction of 70.6% (3.4× speedup) in the training time. Overall, our method generates slightly worse views but has a much faster turnaround time, and this difference in time will only increase as we scale up the resolution of the scene. Although our image quality is slightly worse, we posit that it can be easily improved with fine-tuning the hyperparameters of the hashgrid and the subsequent MLPs, which we keep as a future work.

	DVGO		***mrhe-NeRF***	
	PSNR	Training time (mins)	PSNR	Training time (mins)
Lego	35.79	134	31.89	36
Chair	35.50	105	33.21	23
Ficus	33.67	101	32.84	41
Avg	**34.98**	**113.33**	**32.64**	**33.33**

Table 1. ***mrhe-NeRF** achieves a slightly worse PSNR but has a much faster training time in comparison.*

4.3. Comparison of NeRF–ST with ST–NeRF

Next, we compare our two design choices. We see that while the training time is comparable in both the designs, generating images using NeRF followed by applying style transfer (NeRF–ST) produces much better results than applying style transfer on images before using NeRF (ST–NeRF), as shown in Figure 3. We visualize the coarse geometry obtained in our first stage to analyse the reason for

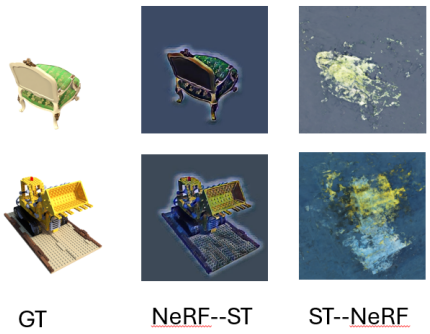

Figure 3. *NeRF–ST generates much clearer novel stylized images than ST–NeRF.*

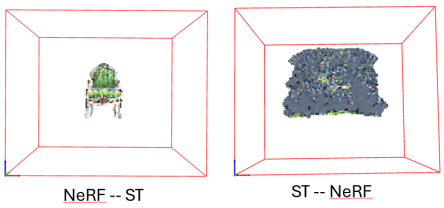

Figure 4. *The dense grid used can obtain precise geometry of the object when trained on vanilla images. However, when trained on stylized images, there is no clear geometry and the dense grid returns a blob as the object.*

this performance degradation. Figure 4 shows the geometry obtained by the dense voxel grids for both designs. We see that while the coarse stage can accurately deduce the geometry of a chair when it is trained with vanilla images, the coarse stage produces a blob instead of a chair if style transfer is applied first. Consequently, our fine stage gets noisy inputs and predicts images with a lot of clutter. We hypothesize that the coarse stage cannot obtain a clear geometry because stylized images may have embedded style-related features or fuzzy details that the dense grid cannot decipher properly. One way to improve the performance would be to replace the grid with a neural network that can capture the more complicated and intricate details in the images. However, more analysis is needed to find the concrete reason for this degradation and improve the performance.

From our comparison, we conclude that NeRF–ST is the better design for generating stylized novel views. Thus, we proceed with this design for the next set of experiments described in the following section.

4.4. NeRF–ST stylized view results

Once we have finalized our design choice, we generate stylized novel views for the lego, chair, and ficus datasets. We evaluate the stylized views both qualitatively by looking at the generated images (Section 4.4.1) and quantitatively using the "effectiveness" metric (Section 4.4.2).

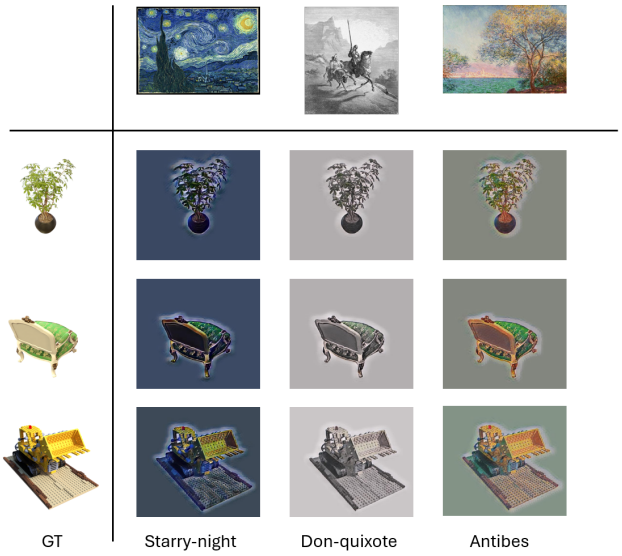

Figure 5. *Qualitative evaluation of stylized novel views. Our NeRF–ST design can effectively transfer the style components while keeping the content image intact.*

Method	Calibrated E-score
NeRF Then ST	-11.805
ST then NeRF	-14.495*
Comparison Images	-5.583

Table 2. NeRF then ST shows slightly improved style output. (* ST then NeRF only evaluated on one dataset and one style due to time constraints.)

4.4.1 Qualitative Evaluation of Output

Figure 5 shows the generated images using the NeRF–ST design for three datasets and three styles. We see that our approach preserves the content while effectively transferring the style onto the image.

4.4.2 Quantitative Evaluation of Output

To evaluate the preserved style of output we used the Effectiveness metric proposed by Yeh et al. [20]. Larger scores (i.e. closer to zero) represent that more of the style was preserved in the final image. The numbers displayed table 2 are gathered by averaging the E-score across all output images before they are stitched into a video. NeRF then ST is the averaged score across 3 styles (Starry Night, Don-Quixote, Antibes in the morning) applied to 3 objects (Ficus, Lego, Chair). Due to time constraints, ST then NeRF was only evaluated on the Lego dataset with Starry Night style, though effectiveness was consistently much better for NeRF then ST across all object-style combinations. The

Figure 6. *Running **mrhe-NeRF** on a created bottle dataset yield good quality images in quick time.*

calculation uses the E-score weights calibrated from user studies proposed in the original paper. The comparison images represent a random selection of images from the NeRF llff data [15], with the average of the scores being displayed in the table above. The score is likely much larger because the NeRF synthetic [15] dataset used as input to our model has its background stripped away while the NeRF llff data does not.

The results reveal that some of the transferred style is lost in the NeRF process. However, despite the intense visual artifacts, most of the style is actually preserved. Qualitatively, the artifacts in the output do appear to be in the target style, just obscuring the content object.

4.5. *mrhe-NeRF* on new dataset

Finally, we created a new "bottle" dataset following the guidelines specified in [17] to test our ***mrhe-NeRF*** implementation on new data. Figure 6 shows the qualitative results. Our implementation performs well on the new dataset, generating novel views with PSNR 27.39 at 256^3 grid resolution. We keep stylizing the novel views as future work.

5. Discussion and future work

Although our initial experiments show that NeRF–ST is a better method than ST–NeRF to generate stylized views, we believe more analysis is needed for conclusive proof. For example, when images are stylized first, style features are embedded into the input. Hence, adding more features as input to the color- and density-MLPs, or replacing the MLPs with more advanced networks like CNNs may be an interesting direction to explore in the future.

Another area to explore in the future would be improving the performance of our ***mrhe-NeRF*** . As shown in Section 4, ***mrhe-NeRF*** performs slightly worse than DVGO. We hypothesize that its performance can be improved in several ways — for example, one way is fine-tuning the hyperparameters of the hash-grids and the MLPs. We keep this as another important future work.

6. Conclusion

In recent years, there has been an increased emphasis on generating stylized novel 3D scenes due to the high demand of such content in various applications like animated movies and AR/VR games. In this project, we generated stylized novel views by applying style transfer on NeRF . In attempting the different orderings of applying NeRF and style transfer, we concluded that applying style transfer after NeRF, instead of vice versa, allows for a more quality output. Using our NeRF–ST design, we then generated stylized novel views on three distinct NeRF syntehtic datasets, that all used three different styles, and evaluated the stylized novel views both qualitatively and quantitatively. We also modified a popular NeRF implementation to improve its training speed at high resolution. Our modification reduced the training time by 70.6% (3.4× speedup) at 256^3 resolution while only degrading the PSNR by 6.7%. Finally, we also created a new dataset using a bottle, aptly named "bottle", to test our new NeRF implementation. We believe there is ample opportunity for extending this project, which we plan to do on a future iteration.

References

[1] Amazon. What is nerf? Available at https://aws.amazon.com/what-is/neural-radiance-fields/. 1

[2] J. An, S. Huang, Y. Song, D. Dou, W. Liu, and J. Luo. Artflow: Unbiased image style transfer via reversible neural flows. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 862–871, Los Alamitos, CA, USA, 2021. IEEE Computer Society. 3

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

[4] Yingying Deng, Fan Tang, Weiming Dong, Chongyang Ma, Xingjia Pan, Lei Wang, and Changsheng Xu. Stytr2: Image style transfer with transformers. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11326–11336, 2022. 1, 2, 3, 4

[5] Weichen Fan, Jinghuan Chen, Jiabin Ma, Jun Hou, and Shuai Yi. Styleflow for content-fixed image to image translation. *arXiv preprint arXiv:2207.01909*, 2022. 2

[6] Sara Fridovich-Keil, Alex Yu, Matthew Tancik, Qinhong Chen, Benjamin Recht, and Angjoo Kanazawa. Plenoxels: Radiance fields without neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5501–5510, 2022. 1, 2

[7] Injae Kim, Minhyuk Choi, and Hyunwoo J Kim. Up-nerf: Unconstrained pose prior-free neural radiance field. *Advances in Neural Information Processing Systems*, 36, 2024. 2

[8] Lingzhi Li, Zhen Shen, Zhongshu Wang, Li Shen, and Ping Tan. Streaming radiance fields for 3d video synthe-

sis. *Advances in Neural Information Processing Systems*, 35: 13485–13498, 2022. 2

[9] Zhengqi Li, Simon Niklaus, Noah Snavely, and Oliver Wang. Neural scene flow fields for space-time view synthesis of dynamic scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6498–6508, 2021. 2

[10] Haotong Lin, Sida Peng, Zhen Xu, Yunzhi Yan, Qing Shuai, Hujun Bao, and Xiaowei Zhou. Efficient neural radiance fields for interactive free-viewpoint video. In *SIGGRAPH Asia 2022 Conference Papers*, pages 1–9, 2022. 2

[11] Tsung-Yi Lin, Michael Maire, Serge J. Belongie, Lubomir D. Bourdev, Ross B. Girshick, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: common objects in context. *CoRR*, abs/1405.0312, 2014. 4

[12] Kunhao Liu, Fangneng Zhan, Yiwen Chen, Jiahui Zhang, Yingchen Yu, Abdulmotaleb El Saddik, Shijian Lu, and Eric P Xing. Stylerf: Zero-shot 3d style transfer of neural radiance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8338–8348, 2023. 1, 2

[13] Ricardo Martin-Brualla, Noha Radwan, Mehdi SM Sajjadi, Jonathan T Barron, Alexey Dosovitskiy, and Daniel Duckworth. Nerf in the wild: Neural radiance fields for unconstrained photo collections. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7210–7219, 2021. 2

[14] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. *Communications of the ACM*, 65(1):99–106, 2021. 2

[15] Mildenhall et al. NeRF dataset. Available at https://drive.google.com/drive/folders/128yBriW1IG_3NJ5Rp7APSTZsJqdJdfc1. 6

[16] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM transactions on graphics (TOG)*, 41(4):1–15, 2022. 1, 2, 3

[17] Muller et al. Tips for training NeRF models with Instant Neural Graphics Primitives. Available at https://github.com/NVlabs/instant-ngp/blob/master/docs/nerf_dataset_tips.md. 6

[18] Fred Phillips and Brandy Mackintosh. Wiki Art Gallery, Inc.: A Case for Critical Thinking. *Issues in Accounting Education*, 26(3):593–608, 2011. 4

[19] Cheng Sun, Min Sun, and Hwann-Tzong Chen. Direct voxel grid optimization: Super-fast convergence for radiance fields reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5459–5469, 2022. 1, 2, 3

[20] Mao-Chuang Yeh, Shuai Tang, Anand Bhattad, Chuhang Zou, and David Forsyth. Improving style transfer with calibrated metrics. *arXiv preprint arXiv:1910.09447*, 2019. 2, 4, 5

[21] Alex Yu, Ruilong Li, Matthew Tancik, Hao Li, Ren Ng, and Angjoo Kanazawa. Plenoctrees for real-time rendering of neural radiance fields. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5752–5761, 2021. 2