Fast 3D Style Transfer using Neural Radiance Fields

Philipp Kutschmann
philipp.kutschmann@tum.de

Jonas Zausinger
jonas.zausinger@tum.de

Abstract

In the past, 3D style transfer with neural radiance fields suffered from slow optimization and rendering speed and poor transfer of finer stylistic details. In this project, we address these issues. We improve the time complexity by encoding the input with trainable multiresolution hash tables. Additionally, we realize an effective pixel subsampling to render image patches for style supervision and show that it significantly improves the detail of style transfer.

1. Introduction

3D scene reconstruction is a challenging task in the field of computer vision. In 2020 Mildenhall *et al.* [2] proposed a method that encodes a scene representation in the weights of a neural network. This lightweight representation with a Neural Radiance Field (NeRF) opened up further research possibilities such as stylization of entire 3D scens. To get a consistent stylization of the rendered output images (e.g. to render a video), the 3D model itself needs to be stylized, as opposed to applying 2D style transfer on rendered images which leads to inconsistencies and flickering artifacts.

Previous work addressing the task of 3D stylization with NeRF is often slow at train and test time and lacks stylistic details [1]. Our method improves runtime and quality compared to our baseline method by Chiang *et al.* [1] by enhancing the architecture with multi-resolution hash encoding [3] and by developing effective usage of GPU memory for detailed style transfer.

Specifically, current methods for NeRF-based 3D style transfer are stil constrained by the memory size of the GPU. Therefore a subsampling method of the image is needed to reduce the space complexity. During our project we implemented different subsampling methods (see Section 3.3.2) and evaluated their influence on 3D stylization in combination with our network architecture.

To summarize, our contributions are:

- Multi-resolution hashencoding for NeRF-based 3D stylization to decrease training and rendering times by multiple orders of magnitude compared to Chiang et al. [1]
- A memory efficient patch subsampling approach for better stylization with finer style details present e.g. brush strokes

2. Related Work

Since the introduction of NeRF a multitude of papers were published that build on and expand the original network architecture. A problem with the original method was the long training and inference time. A speedup for the classic NeRF method was developed by Müller *et al.* [3] in 2022. Encoding the inputs with trainable multiresolution hash tables allowed the use of a smaller network. This reduction of floating point and memory access operations results in faster training times and making novel view synthesis real-time capable on modern graphics cards (∼25 FPS on a NVIDIA Quadro RTX 5000). The existing pytorch implementation [4] is heavily used in our method's codebase and is the core reason for our speedup compared to our baseline method.

In 2022 Chiang *et al.* [1] showed that with a two stage training of NeRFs a style transfer for a specific scene with an arbitrary style image can be achieved. During the first training stage a NeRF++ model [5] gets trained while the stylization is learned in the second stage. For the second stage training, the weights of the geometry branch of the neural network get frozen and only the colour branch is adjusted. Furthermore a hypernetwork gets trained, allowing the use of varying style images at inference time and making it possible to apply other styles without retraining the network. While training the second stage, also a different loss function is used compared to the classic NeRF model, comprising a style and a content loss. Unfortunately this method mostly just transferred the colour of the style image and captured very little of the stylistic details such as brush strokes. Additionally it took multiple hours to train the network and multiple minutes to render novel, stylized views. In our work, we speed up the training process by multiple orders of magnitude so that a universal style transfer is no longer required and the network can be easily retrained for different styles and scenes.

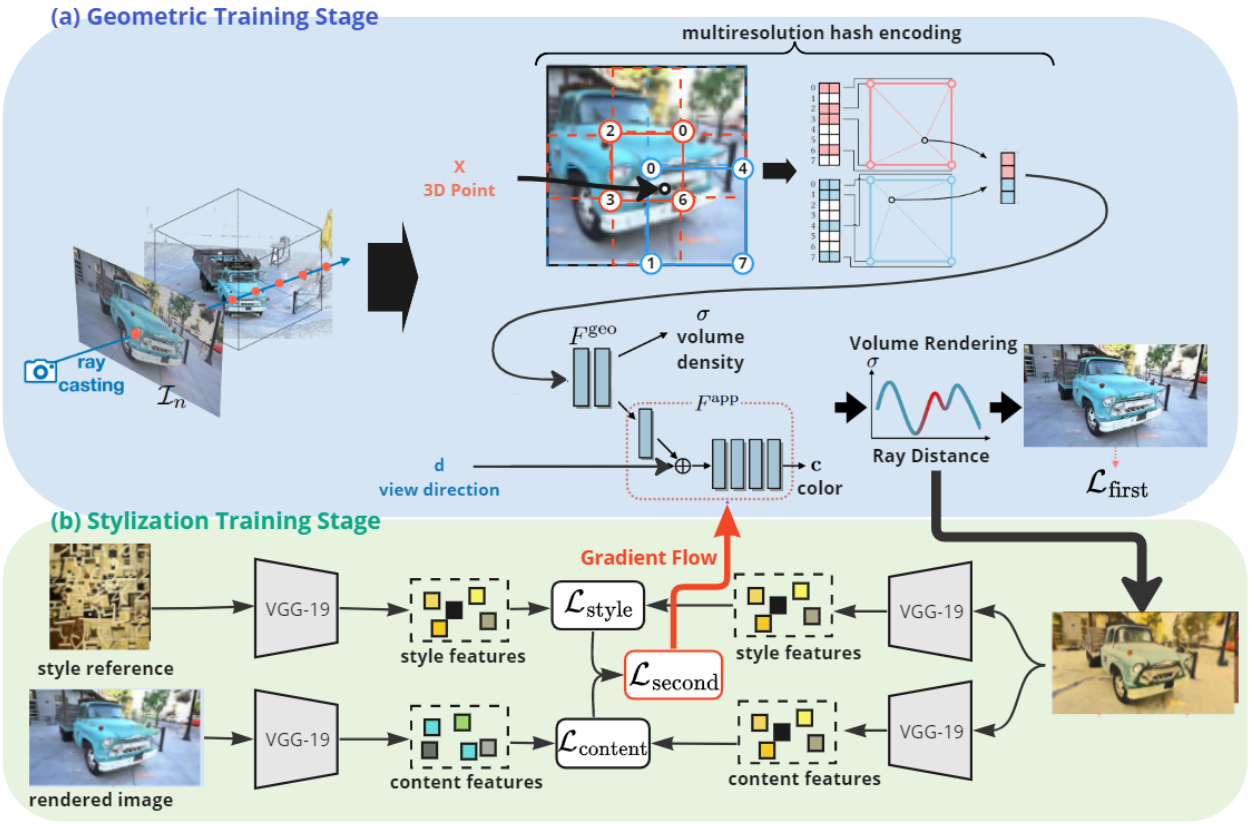

Figure 1. **Overview of our proposed method with two stage training**
(a) The first stage is the geometric training stage, where the NeRF model learns the implicit representation of the 3D scene.
(b) When it has learned the scene reasonable well, we move to the stylization training stage. Here, our model learns to apply the style of the style reference image to the scene. In this stage, a different loss is used to optimize the model, which consists of a style and a content loss. These losses are computed using feature maps obtained from a pretrained VGG-19 network

3. Method

We model a 3D scene with an implicit representation using neural radiance fields (NeRF) and learn to transfer arbitrary style of a 2D image by formulating the stylization of radiance fields as an optimization problem. As input our method expects multi-view RGB images of a 3D scene with the corresponding camera poses and a style image. The output is a stylized 3D representation of the scene from which arbitrary novel views can be rendered. To this end, we split up the training into two different stages as proposed by Chiang *et al.* [1]. First in the geometric training stage the NeRF learns the implicit representation of the 3D scene. Then in the stylization training stage we render images of the radiance fields from different viewpoints in a differentiable manner, and minimize a content loss between the rendered stylized images and the original captured images, and also a style loss between the rendered images and the style image. An overview of our method can be seen in Fig. 1.

3.1. Network Architecture

The Architecture of our stylized NeRF can be seen as consisting of 3 parts. **(i)** A base branch or encoding branch, which processes the input, **(ii)** a geometry branch, which outputs a density value for each pixel, and **(iii)** a appearance branch, which outputs the RGB-value for each pixel. For the encoding branch, we leverage the multiresolution hash encoding proposed by Müller *et al.* [3]. With this special input encoding it is possible to reduce the size of the NeRF model, therefore dramatically reducing training and inference time without sacrificing quality. The input is encoded through multiresolution hash tables of trainable feature vectors whose values are optimized through stochastic gradient descent. For the geometry and appearance branch we use MLPs with 2 and 3 layers respectively.

3.2. Geometric Training Stage

In this training stage the model learns the geometry and radiance of the scene, without stylization yet. We apply volume rendering [2] along camera rays to obtain pixel col-

ors. The loss function to train the model on the scene is the mean square error (MSE) between the color value of each rendered pixel $\hat{c}(r_m)$ and the ground truth color $c(r_m)$ of the corresponding image pixel of ray r_m.

$$\mathcal{L}_{\text{first}} = \frac{1}{M} \sum_{m=1}^{M} ||\hat{c}(r_m) - c(r_m)||_2$$

Because of our input encoding, the training time of this stage takes only a few minutes, unlike our baseline, where this stage needs several hours of training for the model to learn the representation of the 3D scene.

3.3. Stylization Training Stage

When the model has learned the 3D scene sufficiently well in the first stage, we switch to the second training stage. In this stage, we design the optimization process differently from Chiang *et al.* [1] by not only optimizing a hypernetwok, but directly optimizing the NeRF model. This allows us to improve the quality of the style transfer.

The MSE-loss from stage one is replaced with another loss, which consists of a content loss and a style loss.

$$\mathcal{L}_{\text{second}} = \mathcal{L}_{\text{style}} + \lambda_{\text{content}} \mathcal{L}_{\text{content}}$$

3.3.1 Style and Content Loss

Here we use the same style and content losses as they are used for 2D style transfer. The losses will be derived from the feature maps, which are obtained from a Image-Net pretrained VGG-19 network.

$$\mathcal{L}_{\text{content}} = ||\Phi_4(y_c) - \Phi_4(\hat{y})||_2$$

where $\Phi_l(\cdot)$ the denotes the feature representation obtained from the `relu_l-1` layer of an ImageNet-pretrained VGG-19 network. We use a Gram matix-based style loss:

$$\mathcal{L}_{\text{style}} = \sum_l ||G_l^{\Phi}(y) - G_l^{\Phi}(\hat{y})||_2$$

$$G_l^{\Phi}(x)_{c,c'} = \frac{1}{C_l H_l W_l} \sum_{h=1}^{H_l} \sum_{w=1}^{W_l} \Phi_l(x)_{h,w,c} \Phi_l(x)_{h,w,c'}$$

3.3.2 Sub-sampling Techniques

The goal of style transfer is to make the entire scene have a similar style to the reference image. Therefore, the style loss is calculated using global style features. For this reason, we would prefer to render images at full resolution and then calculate the style and content loss based on the full resolution images. However, since rendering a single pixel already requires dense sampling on the camera ray to obtain numerous 3D points for querying the NeRF model, it would be even more costly in terms of GPU memory usage to attempt to render the image at full resolution and perform backpropagation for network optimization [1]. Therefore, we need a technique so that we can compute and backpropagate the losses to a subset of the pixels. To achieve this, we tried two different techniques:

- **Down-sampling**: Here we sample from the whole image, but leave out a certain number of pixels, so we reduce the resolution.
- **Patch wise sub-sampling**: As shown in Fig. 2, we first cut a window of random size, and then down-sample its resolution, so that we get the desired size of pixel subset. This technique was also proposed by Chiang *et al.* [1]

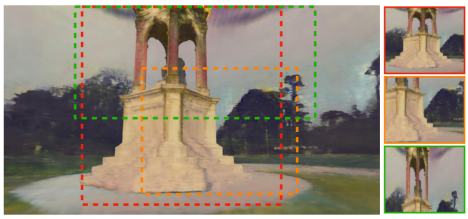

Figure 2. Patch-wise sub-sampling

3.3.3 Freeze Parts of the Model

In the geometric training stage all three branches of the model are trained. However, in the stylization training stage we can now decide if we want to train all three branches to have the best possible style transfer, or if we only train the appearance branch and fix the other weights. The advantage of this is that the geometry of the 3D scene is completely preserved and only the coloring can change, but at the disadvantage that finer style details cannot be transferred as well. We experimentally found that optimizing all three branches gives the best visual pleasing stylization results. By using a content loss and training only for about 15 minutes, the geometry is sufficiently well preserved, while at the same time sufficient network capacity is available to adopt the stylization. These results can be seen in Section 4.1. In appendix C are ablation experiments, where we show the difference in the results when only the appearance branch is retrained.

4. Results and Discussion

To show the quantitative and visual improvements over the baseline paper, we used multiple scenes and tested these with different style images. Please also refer to our reference video for comparison of continuous scene outputs.[1]

[1] https://www.youtube.com/watch?v=-EQwxKx1S4M

Figure 3. Our results and comparison with our baseline (Chiang *et al.* [1])

4.1. Qualitative Results

In comparison to our baseline, our approach manages to transfer not only the color of the style image, but also fine style details such as curved lines, brush strokes and patterns. Looking at the first column in Fig. 3, our method takes over the round shapes from the style image, which is for instance visible in the ear of the fox, as well as in patterns of the wallpaper after stylization. Looking at the second column in Fig. 3, we observe that our method clearly transfers the coarse brush strokes from van Gogh's Starry Night painting and aligns them with the fur of the fox. In contrast, the method by Chiang *et al.* [1] mostly transfers color without clear stylistic patterns. In the appendix, we provide ablation experiments on the sub-sampling techniques and trained branches during stylization and further show that our method produces multi-view consistent results.

4.2. Quantitative Results

One of our main goals was to speed up the training and inference time of stylized NeRFs. We measured the training time it took for our model and the baseline method to achieve visually pleasing results. We additionally profiled inference time for both approaches (Tab 2). Our tests were conducted on a NVIDIA Quadro RTX 5000.

The measurements in table 1 and 2 show that we outperformed the baseline's training and inference time by several orders of magnitude. This allows discarding the approach of training a hypernetwork to achieve a universal style transfer as a new model for a different style image is trained in a much reduced time frame. The reduced inference time made rendering novel views real-time capable. Therefore the stylized 3D model can be visualized in a GUI with the user being able to manipulate the viewing angle and position without much delay.

Training time	First stage	Second stage	Total
Chiang	17h	22h	39h
Ours	5mins	15mins	20mins

Table 1. Training times of our method compared to the baseline

Method	Inference time
Chiang	210 seconds
Ours	0.7 seconds

Table 2. Inference time of our method compared to the baseline

4.3. Limitations

On large-scale scenes with images captured from different distances to objects, our stylization could produce blurry results due to varying scale. Therefore, a depth aware stylization of NeRF scenes is an interesting direction for future work.

5. Conclusion

During this project we have developed and implemented a method that speeds up the training and inference process of stylizing NeRFs. This result was achieved with the use of mutliresolution hash tables as an input encoding. Due to an an alternate subsampling strategy and an adjusted network architecture, enhanced stylistic quality compared to the baseline can be seen. The better quality is noticeable as finer stylistic details got transferred onto the scene. These improvements were verified by our qualitative and quantitative test results.

References

[1] Pei-Ze Chiang, Meng-Shiun Tsai, Hung-Yu Tseng, Wei-Sheng Lai, and Wei-Chen Chiu. Stylizing 3d scene via implicit representation and hypernetwork. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, January 2022. 1, 2, 3, 4, 6

[2] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. In *ECCV*, 2020. 1, 2

[3] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM Trans. Graph.*, 41(4):102:1–102:15, July 2022. 1, 2

[4] Jiaxiang Tang. Torch-ngp: a pytorch implementation of instant-ngp, 2022. https://github.com/ashawkey/torch-ngp. 1

[5] Kai Zhang, Gernot Riegler, Noah Snavely, and Vladlen Koltun. Nerf++: Analyzing and improving neural radiance fields. *arXiv:2010.07492*, 2020. 1

Appendices

A. Qualitative Results

In Fig. 4 one can see our results for different styles and scenes. The last row presents the style output of our baseline method for comparison.

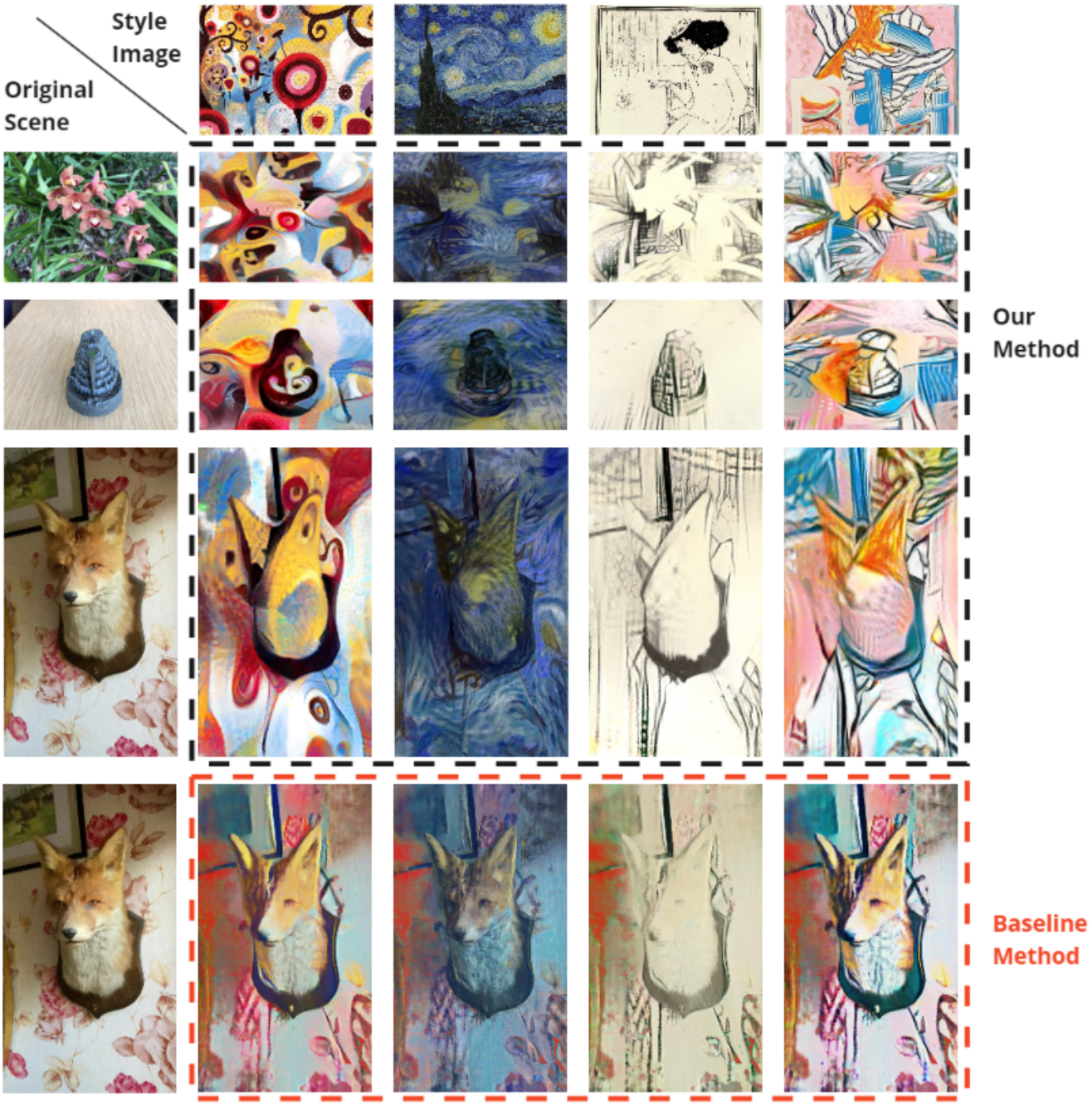

Figure 4. Our results and comparison with our baseline (Chiang *et al.* [1])

B. Multi-View Consistency

In Fig. 5, one can see that our approach results in multi-view consistent stylized output images. For a better impression of the continuous scene outputs we refer to our reference video.[2]

Figure 5. Multi-view consistency

C. Comparison subsampling techniques and trained branches during stylization

In Fig. 6, we show the differences in stylized results when using different pixel sub-sampling techniques and freezing parts of the model in the stylization training stage.

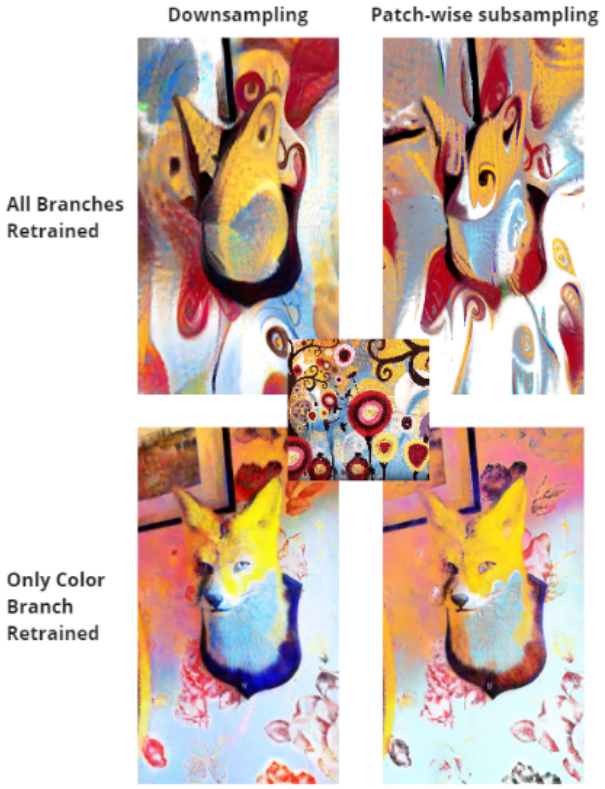

Figure 6. Differences in the stylization output using different sub-sampling techniques and freezing parts of the model.

[2] https://www.youtube.com/watch?v=-EQwxKx1S4M