Editing Neural Radiance Fields

A thesis submitted in partial fulfillment
of the requirements for the degree of

Master of Science
in
Computer Science and Engineering by Research

by

Rahul Goel
2019111034
rahul.goel@research.iiit.ac.in

Advisor: Dr. P. J. Narayanan

International Institute of Information Technology Hyderabad
500 032, India

April 2024

International Institute of Information Technology Hyderabad
Hyderabad, India

CERTIFICATE

This is to certify that work presented in this thesis proposal titled ***Editing Neural Radiance Fields*** by *Rahul Goel* has been carried out under my supervision and is not submitted elsewhere for a degree.

_____ _____

Date Advisor: Dr. P. J. Narayanan

To my parents

Acknowledgements

The research work done as a part of this thesis has been possible due to the support from my family, friends and my advisor Prof. P.J. Narayanan.

I am and will forever be grateful to my parents, Siddharth Goel and Satbeer Goel, for believing in me. Ever since I can remember, they have prioritized me (and my sister) over themselves. They have worked tremendously hard for over two decades to ensure our right upbrining for which I cannot thank them enough. I hope they are proud of the work done in this thesis and the work that will be done beyond it. I also want to thank my elder sister and the rest of my extended family with whom I've spent most of my life.

I would like to thank my friends here at IIIT Hyderabad who have played an important role in the learnings and memories that I will take from this college. The highest honor by far goes to Ishaan Shah, my first roommate, batchmate and then labmate at CVIT. We have learnt, enjoyed and grown a lot together which I'll always appreciate and remember. I would also thank Dhawal for helping me take my first steps in the uncertain world of research. A big thanks to (almost) Dr. Saurabh Saini for being the north-star. I would also thank the CVIT gang: Chandradeep, Astitva, Aakash, Amogh, Shanthika, Pranav and several others. They have played a big role in making CVIT a second home.

Finally, I would like to thank Prof. PJN for believing in me. He has played a very important role in the research work done as a part of the thesis. I deeply appreciate the amount of hours he puts into the research work of his students despite being busy with administrative duties. He has always provided me with great research and career advice which it seems I often forget to follow. Lately, he has been incredibly supportive of a new exploratory phase of my research. I am incredibly happy to have had him as a first research advisor.

I'd also like to give a bit of appreciation to myself. In the past five years, I've found myself in many unhappy situations. I've gone both with the flow and against it to recover. I hope that my future self uses them as motivation when in difficult times.

Abstract

Neural Radiance Fields (NeRFs) have emerged as a pivotal advancement in computer graphics and vision. They provide a framework for rendering highly detailed novel view images from sparse multi-view input data. NeRFs use a continuous function to represent scenes that can be estimated using neural networks. This approach enables the generation of photorealistic images for static scenes.

Outside the domain of image synthesis, NeRFs have been widely adopted as a representation of several downstream including but not limited to scene understanding, augemented reality, scene navigation, segmentation, and 3D asset generation. In this thesis, we explore upon the segmentation and editing capabilities in radiance fields.

We propose a fast style transfer method that leverages multi-view consistent generation of stylized priors to change the appearance vectors in a Tensorial Radiance Field. Our method promises a speed-up of several orders of magnitude in applying style transfer and adheres to the colorscheme from the style image better than previous works.

Next, we tackle the task of segmentation in radiance fields. Our method uses a grid-based feature field which allows extremely fast feature querying and searching. Combined with our stroke-based segmentation, this allws the user to interactively segment objects in a captured radiance field. We improve the state-of-the-art in terms of segmentation quality by a huge margin and in terms of segmentation time by orders of magnitude. Our method enables basic editing capabilities like translation, appearance editing, removal, and composition for which we show preliminary results.

We further explore the problem of composition of radiance fields. Composition of two radiance fields using ray marching requires twice the amount of memory and compute. We use distillation to fuse multiple radiance fields into one to circumvent this problem. Our distillation process is roughly thrice as fast as re-training and produces a unified representation for radiance fields.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Rendering

In 3D computer graphics, the world is often represented as a 3D scene using one of many 3D representations. This 3D scene is then rendered onto an 2D image using a virtual camera; the process is termed rendering. Traditionally, the contents of the scene are known and hand-crafted by artist. This realistic hand-modelling combined with physically accurate light transport results in photorealistic renderings. However, this process is difficult and tedious for the artist and may require hours to days of work depending upon the complexity of the 3D object being designed.

1.2 Novel View Synthesis and Image Based Rendering

However, photorealism is abundantly present in real-world photographs. Researchers have explored the problem of novel view synthesis i.e. capturing images of a real-world scene and rendering the scene from new views. This can allow photorealism and avoid the tedious hand-crafting required to model the scenes. Another major advantage of these approaches is that the cost of rendering an image is independent of the complexity of the scene. The originally proposed light-field rendering [15, 27] methods take densely captured images and interpolates them to take a reconstruct a slice of the underlying light-field. However, for a faithful reconstruction, the sampled images have to be significantly dense. Additionally, the lack of a proxy geometry creates two problems:

- The novel views must lie outside the convex hull of the underlying scene geometry [27].
- The underlying scene cannot be edited or manipulated easily due to a complete implicit representation.

Recent novel-view synthesis methods like NeRF [33] utilize radiance fields which estimate geometry close to the surface. In this thesis, we focus and use such methods and introduce editing capabilities to newer representations like NeRF and it's follow-up works.

1.3 3D Representations in the Era of Machine Learning

Choosing the correct 3D representation is an important and at times a difficult task depending upon the application. It is often observed in computer graphics, computer vision, robotics and medical imaging. For inverse problems (like 3D reconstruction), things get even more difficult since the choice determines how well the inverse problem is solved.

For the purpose of 3D reconstruction, point clouds have been the go-to choice [41] which are later refined, and converted into a different representation (like a mesh) for the required application.

3D representations can broadly be divided into two categories: (i) implicit representations: the form of representation where the geometry in not known directly but satisfies some relationship, (ii) explicit representations: the form of representation where the geometry is directly known (often unprocessed or processed samples).

Explicit representations involve points, voxels, meshes, etc. They are a great choice for applications that requires fast processing since processing of such representations can be parallelized. However, a major drawback is the discretization in the represenation. Opposed to this, implicit represenations like SDFs and UDFs are continuous. They may or may not be fast to evaluate depending upon the operation. An important advantage of implicit representations are that they are often mathematical functions and neural networks being universal approximatoin functions can fit them. This unlocks a great potential of using them as a proxy for learning 3D shapes. Explicit representations can also be obtained through differentiable optimization. Similarly, a set of features can be regressed in a pre-determined voxel lattice. Meshes on the other hand are extremely difficult to directly obtain through *learning-based* methods due to their complex connectivity. Recent developments show that hybrid representations like *voxel grid + MLP* or *point cloud + mlp* provide great trade-offs between implicit and explicit shape representations [52].

1.4 Radiance Fields

A Radiance Field maps the radiance present in a 3D scene as view-dependent color values, approximated using RGB. Formally, for a given 3D point $x \in \mathcal{R}^3$ and a view-direction described using polar angles $d \in \mathcal{S}^2$, a radiance field maps it $\mathcal{F}$ maps it to an RGB vector $c \in \mathcal{R}^3$ s.t. $0 \leq c_i \leq 1$:

$$F(x, d) : \mathcal{R}^3 \times \mathcal{S}^2 \rightarrow \mathcal{R}^3$$

A radiance field represents the light being transmitted at any point in any direction in a scene. Therefore, it can be used to capture the appearance of objects in a given scene at a given moment in time. Therefore, they are a good choice for solving the inverse problem of reconstruction of appearance of 3D objects.

1.4.1 Volumetric Density

We have been successful in modeling the appearance of objects using a radiance field. However, we have not modelled the geometry yet. Radiance fields are coupled with volumetric density to represent the entire scene. The volumetric density is a function that takes in the 3D location $x \in \mathcal{R}^3$ and outputs a density value $\sigma \in [0, 1]$.

Intutively, the density at a point represents the amount of opaque content present at a position. 0 represents completely transparency i.e. light completely passes through while 1 represents complete opaqueness i.e. no light passes through. Another intution is that the density at a location represents the probability of a ray terminating at that location.

It is important to note that this property, unlike radiance/colour, does not depend on the direction of viewing the 3D point.

1.4.2 Volumetric Rendering Equation

Given a camera ray $r(t) = o + td$, with near and far bounds to be t_n and t_f respectively, the expected color of the ray $C(r)$ is given by:

$$C(r) = \int_{t_n}^{t_f} T(t)\sigma(r(t))c(r(t), d)dt \text{ where } T(t) = \exp\left(-\int_{t_n}^{t} \sigma(r(s))ds\right)$$

Firstly, the function $T(t)$ denotes the accumulated transmittance along the ray from t_n to t. Intutively, it is the probability that during it's travel from t_n to t, the ray has not terminated. A higher transmittance means that the ray has not terminated while a lower transmittance means that the ray has terminated.

The product of transmittance $T(t)$ and volumetric density $\sigma(t)$ at a point indicates the contribution of that point in the final color of the ray. Hence, the final colour can be determined by the integrating the color-contributions along the ray where the color-contribution is simply the contribution of the point multiplied by the color of the point.

The continuous integral is converted to a discrete estimation using numerical quadrature:

$$\hat{C}(r) = \sum_{i=1}^{N} T_i(1 - \exp(\sigma_i\delta_i))c_i \text{ where } T_i = \exp\left(-\sum_{j=1}^{i-1} \sigma_j\delta_j\right)$$

This function that calculates $\hat{C}(r)$ from sampled set of (c_i, σ_i) also reduces to the traditional α-compositing formula upon substituting $\alpha = 1 - \exp(-\sigma_i\delta_i)$.

As mentioned briefly previously, this equation can be used to render a camera ray. To render an entire image, multiple such camera rays can be shot to render an entire array of pixels.

The beautiful thing about this formulation is that it is differentiable enabling us to calculate gradients and perform gradient descent to learn the radiance field along with a volumetric density field. NeRF, as the name suggests, uses a neural network (an MLP to be more specific) to represent these fields. The weights of the MLP are learnt using gradient descent. For further details, please refer to Chapter 2.

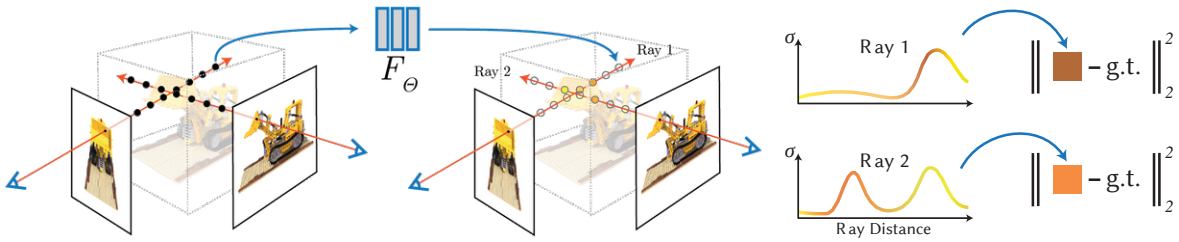


Figure 1.1: *NeRF:* NeRF uses a neural network to estimate the volumetric density and the view-directional radiance for a given co-ordinate and view-direction. It blends these values according to the volumetric rendering equation. The photometric loss is calculated between the rendered images and the ground truth to calculate the gradients for back-propagation. (Figure taken from NeRF [33])

1.5 Our contributions

We first explore editing Neural Radiance Fields through style transfer. Existing style transfer methods firstly rely on the original method of Neural Radiance Fields. Moreover, they attempt changing style codes directly in the latent space or rely on repeated optimization from inconsistent images. We realize another method that relies on multi-view consistent generation of stylized priors which can assist in rapid adaption of style to an already trained radiance field. Our work shows speed-up in style adaption by several orders of magnitude attributed to both grid-based representation of NeRFs and our stylization strategy. With our method, the resultant stylized radiance field adheres better to the colorscheme provided by the reference style image. Our work shows short-term and long-term view consistency similar to previous methods.

We next explore fast, interactive and accurate segmentation in the domain of grid-based radiance fields. There has been previous work in object based selection/segmentation in the space of radiance fields. However, there have been several drawbacks to them which we address. Although previous works allow patch-based or stroke-based selection of objects in radiance fields, they are non-interactive and excruciatingly slow. Moreover, the segmentation produced by them is not accurate. Our method of segmentation is more accurate and orders of magnitude faster than the previous approaches. It allows users to select and remove objects in the scene iteratively and hence it is truly interactive. Furthermore, we showcase how segmentation is an important step which enables editing operations for individual objects.

Finally, we tackle the problem of *fusing* multiple radiance fields. Many works in the NeRF space show preliminary results of composition. We observe that to render to radiance fields simultaneously, all the methods require twice the amount of storage and time. We propose to distill the information from multiple radiance fields to a single one to circumvent this issue. This *fused* representation allows us to render multiple composited radiance fields using the memory and time complexities of a single one.

Concretely, the contributions of this thesis are:

- A methodology to rapidly apply multi-view consistent style transfer on a pre-trained radiance field, albeit already having a style network trained. (Chapter 3)

- A method to quickly and interactively segment grid-based radiance fields which can be used to generate multi-view segmentation masks or editing in radiance fields. (Chapter 4)
- A distillation method to fuse multiple composed radiance fields into a single one improving storage and runtime performance. (Chapter 5)

This thesis resulted in a full paper in ICVGIP 2022 [12], a full paper in CVPR 2023 [13] and a short paper at XR-NeRF Workshop at CVPR 2023 [14].

In the next chapter, we explain the related work relevant to our contributions.

Chapter 2

Related Work

2.1 NeRF: Neural Radiance Fields

The formulation of radiance fields and the volumetric rendering equation described in Chapter 1 is differentiable. Neural Radiance Fields (NeRF) [33] utilize a coordinate-based neural network to learn the volumetric density and view-directional radiance. The input to the network is the coordinate of a 3D point and the view-direction of the ray. The output of the network is the volumetric density and a view-direction dependent color represented using a 3-dimensional vector RGB. The authors show that fully connected deep networks are biased towards learning low frequencies which they mitigate by applying positional encoding: mapping the inputs co-ordinates and view-directions to their sinusoids at multiple frequency levels. Positional encoding [45] *"lifts the inputs to a higher dimensional space"* making it easier for the neural network to learn high frequency information across space.

NeRF showed state-of-the-art results for novel view synthesis but due to an MLP being evaluated hundreds of times per pixel, training and inference is excruciatingly slow.

2.2 Grid-Based Radiance Fields

Due to an 8-layer neural network being queried hundreds of times to render every pixel in an image, NeRFs [33] take close to 24 hours to train on a small scale scene. Due to the same reasons, at inference time, it takes several seconds to render as single image.

To tackle this, there has been a shift towards grid-based or hybrid radiance fields that use a lattice structure to store local features which can be interpreted by a function (or MLP) to get radiance. DVGO [42, 43] employ a 3D voxel grid for feature storage which is followed by a small MLP. For a point x, the features in the nearby 8 voxel vertices are tri-linearly interpolated. These features are concatenated with the view-direction of the ray and passed through the MLP to obtain color that can change with a change in view-direction. Plenoxels [10] follows a similar grid-based approach. However, they store the coefficients for Spherical Harmonics in the grid which are decoded using the SH function of a chosen degree. It is observed that the SH-based representation is not as faithful as MLP is representing high-frequency specular information.

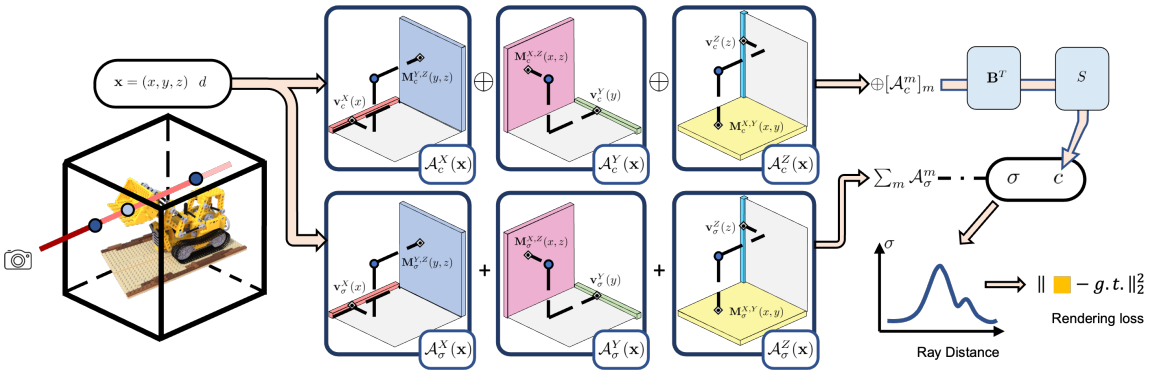


Figure 2.1: *TensoRF:* Tensorial Radiance Fields (TensoRF) represent the 3D scene as decomposed tensors *(VM decomposition)*. The decomposed/projected features undergo outer product to obtain the feature at a 3D coordinate which is decoded by an MLP and rendered using the volumetric rendering equation. We use TensoRF extensively in our works. (Figure taken from TensoRF [6])

Grid-based radiance fields significantly speed up the training and inference time. Training takes around 15 minutes while rendering an image takes around a second. The reason for this massive speed-up is that now only a few parameters are optimized per sample on the ray compared to an entire 8-layer neural network in NeRF. Grids, being spatial, offer faster look-ups. However, they are bulky and require close to 1GB of storage for a good resolution.

An important work that tackles this problem is TensoRF [6]. They propose a low-rank approximation of the 3D grid through tensor decomposition. The feature vector at a 3D point can be calculated by taking the tensor product / outer product of a 1D-vector and a 2D-matrix. Since, this process is differentiable, the vector and matrix (VM decomposition) can be optimized through gradient descent. This new representation takes the same training time and inference time as other grid-based radiance fields but reduces the memory requirement from $O(N^3)$ to $O(N^2)$ for N^3 voxel resolution.

Until recently, the hash-grid representation proposed by Instant-NGP [34] offered the fastest implementation of radiance fields by leveraging a learnable hash-grid, an extremely fast on-chip neural network, and a complete CUDA implementation.

In our works, we extensively used grid-based radiance fields: DVGO [42, 43] and TensoRF [6] to represent volumetric density, color and semantic features.

2.3 Feature Fields

Radiance fields are being rapidly adopted for several downstream applications. This is due to their capability of learning large-scale scenes from sparse views and produce excellent novel views for static scenes. Researchers are exploring an important branch: learning other features alongside the view-dependent colors.

SemanticNeRF [55] explores incorporating semantic segmentation logits alongside appearance and geometry in an MLP-based radiance field. They show its advantages in novel semantic view synthesis, label denoising, and label interpolation.

Works like N3F [48] and DFF [24] distill semantic features from general purpose feature predictors like DINO [5] and LSeg [28]. They show that distilling these 2D features from multiple-views into 3D produces a more powerful feature-rich 3D representation. They show that these distilled features can be used to select and carve-out objects in 3D by user's choice. DFF and LeRF [22] show that language features can be embedded into radiance fields leading to text-based selection and editing.

In our works, we use similar ideas and distill semantic features like DINO [5] from multiple views into grid-based radiance fields. Like N3F [48], we observe an big jump in the quality of features after distillation. Alongside distilled featuers, we leverage the 3D spatiality coming from grids to improve the segmentation and editing capabilities in radiance fields.

2.4 Appearance Editing in Radiance Fields

A simple, although less controllable, form of editing is global appearance editing of the entire scene. Global appearance editing is a good approach to test the editing possiblities in a 3D representation. We explore style transfer as a form of global editing in the representation of radiance fields.

Stylizing images has been a well-studied problem in the vision community. The method proposed by Gatys et al. [11] optimizes white noise to match the content of one image while transferring the style from the other. Johnson et al. [20] proposed to use simple feed-forward architecture, which produces stylized content in real-time. While being quick at producing results, they require a separate neural network for each style. These works provide a strong basis for stylizing 2D content.

Recently, style transfer has been extended to Neural Radiance Fields. Some work [7] use a hyper-network [16] trained on the style to predict the weights of the appearance MLP. StylizedNeRF [19] appraoches this problem by mutually optimizing 2D-3D consistency but suffer from inaccurate style transfer. SNeRF [35] devises a new iterative re-training strategy. Theses methods and especially SNeRF suffer from a very high training cost.

We take inspiration from SNeRF and train a radiance field representation on stylized images themselves. We realize the limitations in their approach and propose solutions that enable faster and accurate adaptation of style as described in Chapter 3.

2.5 Structural Editing and Segmenting Radiance Fields

Since its initial release, the popularity of radiance fields has only risen. Radiance fields are getting better on two ends simultaneously: (i) improving in quality [2, 3], (ii) improving in speed [10, 21, 34].

Naturally, researchers are investigating methods to manipulate and edit an already captured radiance fields in a more controllable fashion compared to Sec. 2.4. Seamless editing can enable us to capture and

augment real world scenes and visualize them in real-time. Improvements in quality further improves photo-realism.

In our works, we show preliminary results in editing and manipulating already captured radiance fields under some constraints. Our work acts as a proof-of-concept of editing of neural radiance fields and ideas from it can be taken and applied to achieve the ultimate goal of seamless editing.

Style transfer is a good proxy to test global appearance editing in radiance fields. It can help in realizing multiple problems that can pop-up in editing attempt on neural radiance fields: multi-view consistency in appearance, manipulation of existing radiance, changes in geometry and how to decouple changes in geometry and appearance, etc. Style transfer has been explored in NeRFs since it's advent [19, 35]. In our work StyleTRF [12], we explore the choices previous works made for style transfer and offer better subtitutes. As a result, we obtain style transfer which can be obtained orders of magnitude faster.

Object selection/segmentation from an already learnt radiance fields is an extremely challenging and important task. Due the novel view synthesis capabilities of radiance fields, segmenting them can provide masks from unseen views. Moreover, 3D segmentation in this representation can enable localizing objects which makes manipulating them for editing a much easier task since it restricts the control to the desired object. Initial methods like Semantic NeRF [55] tried directly regressing semantic labels in the novel views from sparse priors. A few leveraged deep image features like DINO [5] and LSeg [28] to attribute semantics to the 3D scene points. N3F [48] and DFF [24] demonstrate object-specific segmentation using deep semantics. Though these methods support segmentation, interactive content addition and removal are not supported by them, as the underlying scene representation is an implicit neural function that prohibits easy alterations and extensions into other application scenarios. NVOS [38] follows a 3D variant of GrabCut using the positive and negative strokes for segmentation of scenes represented as RFs and MPI [49] but struggles to produce faithful segmentation while incurring significant performance overhead.

In our work ISRF [13], we use a 2D-3D distillation-based approach similar to N3F and DFF, but focus on fine-grained interactive segmentation. We build upon the proven methods like semantic features, nearest neighbor matching, and voxel carving techniques [17, 25] by extending them to radiance fields. We experimentally prove how such simple techniques combined with the appropriate scene representation can improve result quality and fine details while simultaneously being quite intuitive, interpretable, and efficient (80$\times$ faster than NVOS).

2.6 Composition and Fusion of Radiance Fields

A simple composition using affine transformation has been attempted by works like D^{2}NeRF [51] where compact Neural-RFs were employed to obtain the desired composition. The desired composition increases memory footprints and render times in proportion to the number of scenes involved. In the case of explicit volumetric lattice, though the partial content retrieval is more accurate, the memory

requirement grows rapidly, leading to infeasibility when more than a few scenes are being composited. Hence, a fused representation of composition is desired owing to memory footprints and render times as that of a single RF.

Works like PVD [9] provide methodologies for faster distillation of one RF to another in a supervised setting. But the work is limited to a single RF representation. We draw ideas from this work and build a FusedRF [14] representation that is compact and easy to render.

Chapter 3

StyleTRF: Stylizing Tensorial Radiance Fields

Figure 3.1: *Stylization:* We show results of stylization using our technique presented in Sec. 3.1. We stylize the PLAY-GROUND scene using two different styles *mosaic* and *mudbath*. Our Pipeline adapts style in a nominal time of 40 sec on top of a pre-optimized TensoRF scene representation. Once the style is adapted in accordance, stylized novel views can be generated with traditional volumetric rendering techniques.

Stylized view generation of scenes captured casually using a camera has received much attention recently. The geometry and appearance of the scene are typically captured as neural point sets or neural radiance fields in the previous work. An image stylization method is used to stylize the captured appearance by training its network jointly or iteratively with the structure capture network. The state-of-the-art SNeRF [35] method trains the NeRF and stylization network in an alternating manner. These methods have high training time and require joint optimization. In this work, we present StyleTRF, a compact, quick-to-optimize strategy for stylized view generation using TensoRF [6]. The appearance part is fine-tuned using sparse stylized priors of a few views rendered using the TensoRF representation for a few iterations. Our method thus effectively decouples style-adaption from view capture and is much faster than the previous methods. We show state-of-the-art results on several scenes used for this purpose.

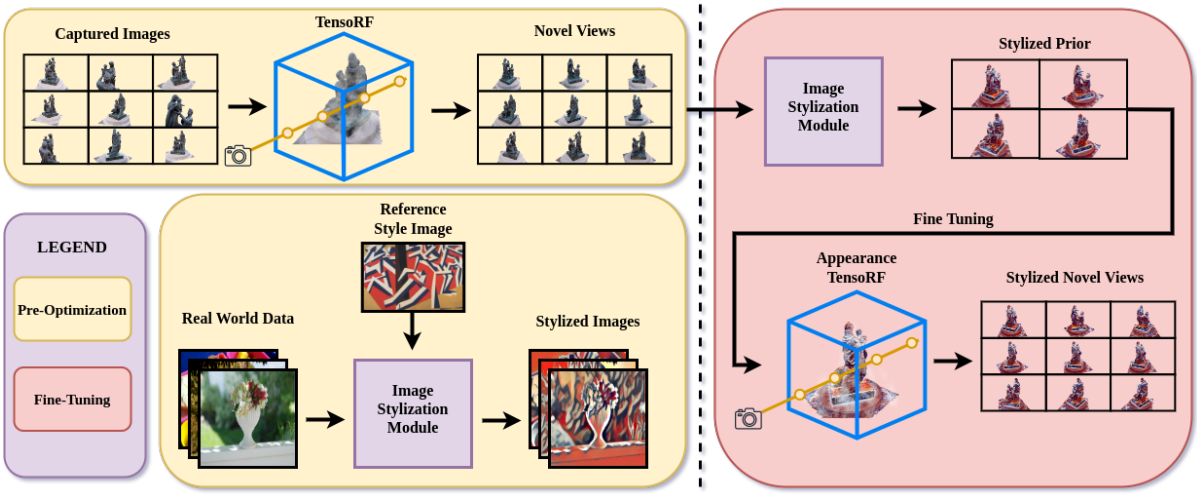


Figure 3.2: *System overview*: The pipeline diagram presents an overview of the strategy employed by our work. We first optimize Tensorial Radiance Fields for representation of the scene as proposed by [6]. Concurrently we optimize stylization module utilizing [20] method on COCO14-dataset [30]. The stylization module is then used to stylize a sparse set of novel views generated by the pre-optimized TensoRF. These stylized views act as sparse style-priors and are used to fine-tune the appearance of the previously optimized scene representation. It is to be noted that we freeze the density terms of the TensoRF and only alter the appearance vectors which retains geometric while adapting the novel style.

3.1 Method

Given a set of posed images of a scene and a reference style image, our aim is to render stylized novel views of the scene, which are consistent in appearance and geometry across different frames. We achieve this by fine-tuning the appearance of a TensoRF [6] using stylized images.

3.1.1 Preprocessing Stylization Module

In our method, we use the stylization method presented by [20], which produces stylized content in real time. The approach requires a per-style training of a CNN, for which it utilizes COCO2014 dataset [30]. The per-style optimization takes an approximate time of 20-minutes and at the inference, it produces 30 images/sec. Since training per-style takes only 20 minutes, we can simultaneously train multiple [20] models for each desirable style independently and infer based on the preference.

We choose [20] as our underlying stylization module over [11], unlike others [35] because it gives stable stylization for two closeby viewpoints. In the case of [11], stylizing two closeby viewpoints that share a large amount of image content usually results in drastically different stylization as shown in Fig. 3.5. This is because the image developed using [11] start with white-noise and try to converge the distribution of *reference-style* and *content* images. This results in unstable stylization across nearby viewpoints as the optimization depends on various factors such as the learning rate, initialization, and

type of optimizer. One of the reasons for the slow multi-iter stylization in SNeRF was to alleviate the inconsistencies caused by [11] based stylization.

On the other hand, once [20] is trained, the output image is deterministic with respect to the input image. Furthermore, two close-by camera viewpoints share a large amount of spatial information. Since CNN's are spatially invariant in the local region, our image transformation network gives fairly stable stylization across closeby viewpoints Fig. 3.5.

Though [20] provide stable stylization across adjacent views, it is *not temporally consistent*. Hence it is to be noted that, we do not rely on temporally-consistent stylization while only relying on the nominally-stable stylization of nearby viewpoints. The per-style training of our Stylization Module is depicted in Fig. 4.2.

3.1.2 Scene Representation using TensoRF

In order to generate a novel view, it is necessary to have a geometric representation and appearance understanding. In order to address this issue, we use Radiance fields($\mathcal{L}$). The radiance fields [56] ($\mathcal{L}$) is given by:

$$\mathcal{L} : R^3 \times S^2 \rightarrow R^3 \tag{3.1}$$

where R^3 on the left is the scene's world space; S^2 is the sphere of directions about each point, and the R^3 on the right is radiance at the point. Radiance fields in a way encode geometry and radiance in their mapped representation. Though there exist various recent adaptions of the mapping function presented in Eq.(3.1) like NeRF [33], PlenOxel [10], we chose TensoRF [6] which is a compact, accurate, and fast-to-optimize representation of Radiance Fields. We use a specifically *VM-48* variant of TensoRF which optimizes a scene within a timeframe of 15-20 minutes while maintaining a memory footprint of 10-15MB.

The TensoRF representation utilizes a Tensor decomposition known as VM-decomposition which in itself is a special case of BT-decomposition. This reduces the voxel grid memory by order of $\mathcal{O}(n)$. This scene optimization can be done independently for every scene irrespective of style. In a later phase, we alter the appearance to adapt to the reference style in a short period of 40-50 seconds.

Similar to TensoRF [6], we utilize $L1$ norm loss and total variation (TV) loss (Eq. 3.2) for regularization. This helps our process to avoid getting stuck in local minima and prevents overfitting. For scenes with less number of captured images, TV loss is a better choice to obtain good results. The equation for TV loss is given by:

$$\mathcal{L}_{TV} = \frac{1}{N} \sum \left(\sqrt{\Delta^2 T_\sigma} + 0.1 \sqrt{\Delta^2 T_C} \right) \tag{3.2}$$

Here, Δ^2 is the squared difference between the neighboring values in our tensors, N is the total number of parameters across our TensoRF representation T. T_σ represents the density value and T_C represent the appearance value in the TensoRF representation. They are weighted in the ratio of $10 : 1$ respectively. More details about TV Loss can be found in the work by [6].

3.1.3 Stylizing TensoRF representation

3.1.3.1 Novel View stylization

Upon optimizing the radiance fields which encode the geometry and radiance of the scene as discussed in the Sec.3.1.2, we render a sparse set of $20 - 30$ novel views in a simple trajectory (spiral). We stylize these novel renders using the pre-trained Stylization Module discussed in 3.1.1. Fig. 4.2 (top-right) shows the generation stylization of these renders utilizing the Stylization Module.

3.1.3.2 Stylizing Appearance of TensoRF

We utilize the sparse set of stylized novel views generated using the per-style optimized [20] module and optimize the appearance vectors of the TensoRF. During the process of optimization, we ensure that the density terms are frozen, and only the appearance is altered. We explicitly chose to freeze density as we have observed that stylizing looks pleasing and free from artifacts when density is kept frozen. This fine-tuning only takes a nominal time of downwards of 40 secs. Once the fine-tuning is done, we obtain a geometric scene represented as Tensorial Radiance fields, which can be used to render *stylized novel views* with consistent appearance across the viewpoints. The rendering of each image having a resolution of 800×800 takes an approximate time of 4-5 seconds. Fig. 4.2 (bottom-right) shows the appearance modification from a sparse set of inputs and novel view generation.

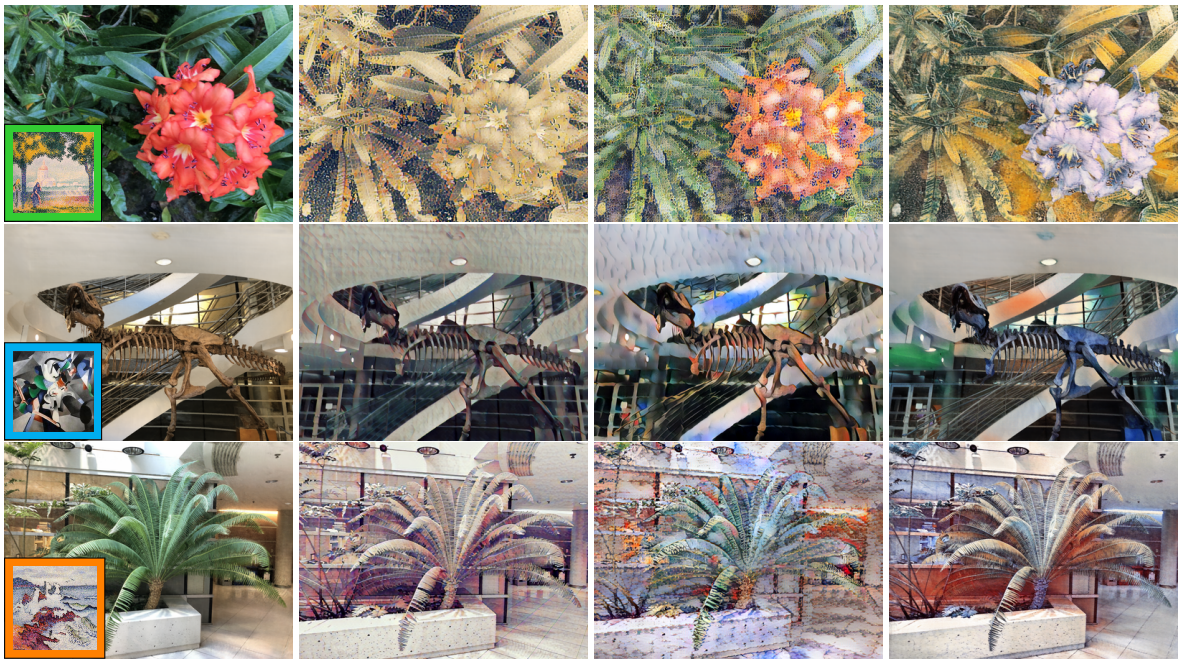
3.2 Implementation Details

3.2.1 Optimizing TensoRF

The training/optimization of radiance fields requires information of the camera poses from which an image is captured. In the case of real scenes, we rely on COLMAP [40]to obtain this information and in the case of synthetic scenes, we use the data obtained from Blender. We optimize TensoRF (VM-Split-48) on the input images for $15k$ iterations. In each iteration, we shoot 4096 rays into the voxel grid. We obtain the radiance using volumetric rendering (Eq. 3.3) and optimize the grid iteratively.

$$\mathcal{C} = \sum_{q=1}^{Q} \tau_q \left(1 - exp\left(-\sigma_q \Delta_q\right)\right) c_q,\ \tau_q = exp\left(-\sum_{p=1}^{q-1} \sigma_p \Delta_p\right) \tag{3.3}$$

We use Adam optimizer [23] which is initialized to a learning rate of 0.02 and is re-initialized to 0.02 after upsampling. The voxel grid is initialized with an effective resolution of 128^3 and iteratively upsampled every 1000 iterations, first upsampling starting 2000 until 5000 iterations are reached. We finally reach an effective voxel-grid resolution of 640^3 for real-world scenes and 300^3 for synthetic scenes. It is to be noted that the resolution mentioned here is *effective* but not exact, as the VM-decomposition provided by TensoRF presents a compact representation of voxel grid. Similar to TensoRF, we bilinearly interpolate the matrix and linearly interpolate the vector in the VM decomposed representation



(a) GT Unstylized (b) ReReVST (c) StylizedNeRF (d) StyleTRF (Ours)

Figure 3.3: *Qualitative Comparison*: [Row 1: *santamaria*, Row 2: *udnie*, Row 3: *mediterranean*] Here we show the comparisons Un-stylized frame in column-(a), temporally consistent stylization of ReReVST [50]in column-(b), StylizedNeRF [19] in column-(c) and stylization using our pipeline in column-(d). It can be observed that due to the usage of combined neural representation density and radiance, the style adaptation is affecting in the case of StylizedNeRF. Observe noisy geometric structures in *mediter* applied onto FERN .

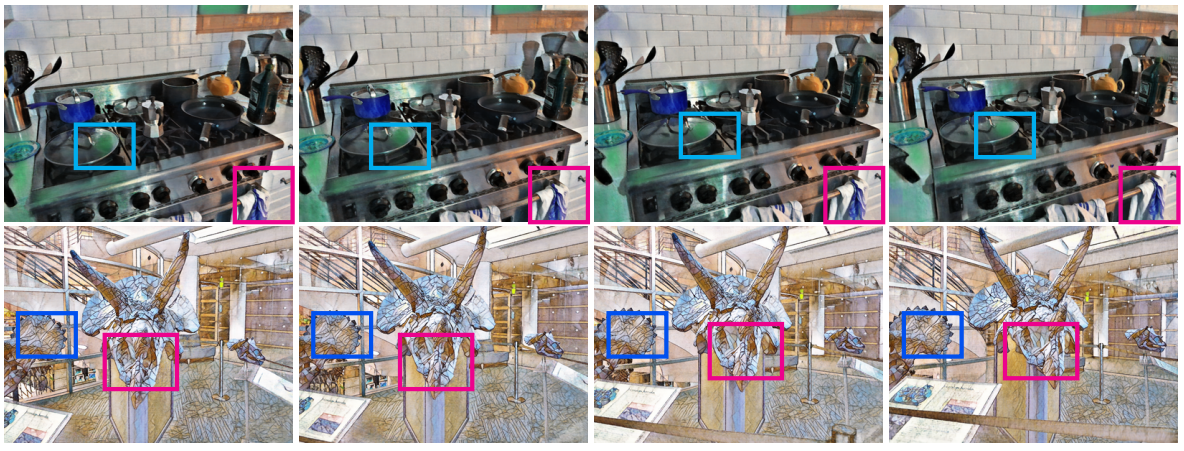
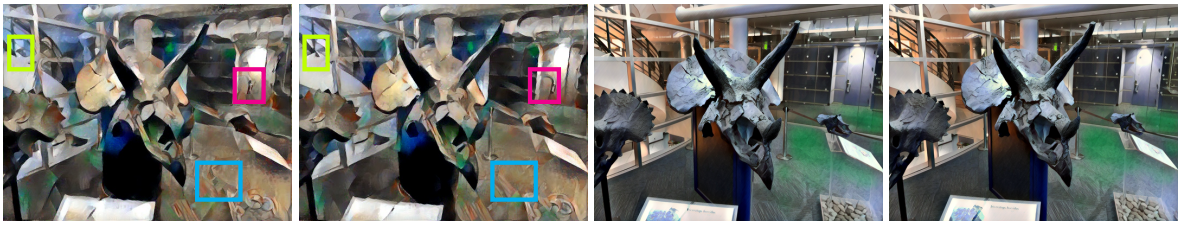


Figure 3.4: *View Consistency Across Frames*: The figure shows stylized novel views of a simple trajectory. To keep it simple we named the frames in t_i with increasing order of i from left to right. It can be observed clearly that our stylization is multi-view consistent both in the case of STOVE and HORNS .



(a) *Gatys* View 1 (b) *Gatys* View 2 (c) *Johnson* View 1 (d) *Johnson* View 2

Figure 3.5: *Gatys vs Johnson Stylized Priors*: The figures shows the output of the stylization modules proposed by [11] and [20] on HORNS stylized based on *udnie*. [11] produces inconsistencies in the style generated across near-by views which provide a poor prior to optimize appearance for our module. [20] provides stable stylization across close-by views as seen in the figure.

during upsampling. This (bi-linear + linear) interpolation is similar to the tri-linear interpolation of the *full-voxel* grid. We perform such interpolations with neighboring voxels during the evaluation of a ray query. This enables us to obtain continuity in our rendered images. This pre-optimization phase requires 10-15 minutes. Once optimized it can be used to generate new views with various camera poses which post-stylization act as priors to our fine-tuning phase.

3.2.2 Stylization

Independently, we train the stylization module [20] with the various reference style image on the COCO-14 Dataset [30] which takes 20 minutes. We train multiple such [20] models for each desired style as the time required to train is quite less. We use this to create stylized priors from the views generated in the previous phase.

While optimizing for the style we freeze the density parameters in the TensoRF representation and optimize only the appearance parameters for a small number of iterations($1k$ iterations) with the stylized prior. This style adaption only takes a nominal time of 40-50 seconds. The style-adapted TensoRF representation obtained in the previous step can be thus used to generate novel stylized views using traditional voxel rendering techniques. The generation of each view takes around 4-5 seconds.

3.3 Experiments

In this section, we present the various experiments with the proposed StyleTRF. We conduct all the experiments mentioned throughout the paper including the comparisons, on a workstation PC equipped with an AMD Ryzen-5800x and an NVidia RTX-3090 GPU. In Sec. 3.3.1 we discuss the choice of stylization module used to stylize the sparse prior used in our approach. We also experimentally compare and contrast between temporal-stylization of smooth trajectories of ground truth 3D content vs Actual 3D-Stylization in Sec. 3.3.2. Finally, in Sec. 3.3.3 we show the effect of different optimization strategies used to stylize the 3D content.

3.3.1 Stylization Module

For stylizing the sparse prior required by our method we use [20]. The concurrent work SNeRF [35] uses [11] method to iteratively stylize the radiance fields. We choose [20] over [11] because the latter depends on several factors such as the initialization, learning rate, and the optimization method which make it unreliable to obtain stable stylization across close-by views let alone temporal consistency. This can be observed in Fig. 3.5. This is one of the reasons, [35] requires multiple epochs to reflect the style in appearance.

On the other hand, [20] use a fixed CNN-based architecture to infer the stylized image. Due to the spatial consistency of CNNs, two close-by views sharing significant spatial content lead to stable stylized close-by views.

3.3.2 Video Stylization v/s 3D Stylization

It can be argued that instead of stylizing 3D content, one can generate novel views on a camera trajectory and use temporally consistent stylization frameworks like ReReVST [50] to obtain stylized novel views.

However, we have observed that though temporal stylization is maintained, the work of ReReVST fails to fully capture the style information. The same can not be said for the stylization of 3D content. In Fig. 4.3 it can be seen that both StylizedNeRF and our method capture better style compared to ReReVST.

3.3.3 Optimization Strategies for Style Adaptation

For the stylization of Radiance fields we present three strategies:

1. *S1:* Optimizing a TensoRF from scratch directly using the stylized priors.

2. *S2:* Pre-optimizing a TensoRF on the original ground truth and adapting for style using sparse stylized priors *without freezing any parameters (both density and appearance)*.

3. *S3:* Pre-optimizing a TensoRF on the original ground truth data and adapting for style using sparse stylized priors while freezing the density.

When following the *S1* we observed geometric artifacts. This is because the stylized priors generated may not share the exact geometry with the ground truth. As stated above in Sec. 3.1.1, [20] does not provide temporally consistent stylization which might also affect the geometry so-optimized in the stylized views in the case of *S1*.

Another strategy *S2* produces considerably better results compared to *S1* as seen in Fig. 3.7. This is because most of the geometric prior is learned from the pre-optimization phase which uses ground-truth images to obtain scene properties.

Scene \ Method	*ReReVST [50]*		*StylizedNeRF [19]*		*Ours*	
	short-term	*long-term*	*short-term*	*long-term*	*short-term*	*long-term*
HORNS	0.0046	0.0137	0.0229	0.0239	**0.0040**	**0.0120**
FERN	0.0028	0.0080	0.0100	0.0168	**0.0020**	**0.0069**
FLOWER	0.0039	0.0106	**0.0020**	0.0277	0.0030	**0.0089**

Table 3.1: *Consistency Metrics:* We show *short-term* and *long-term* consistency metrics across a smooth trajectory generated using our appearance stylized scene representation. We have found that we obtain better *short & long-term* consistency compared to the SOTA 3D-Stylization technique StylizedNeRF [19], while maintaining better style transfer compared to temporal stylizing methods like ReReVST [50].

Though *S2* has produced good stylization at a micro-level, fuzzy geometry can be observed which reduces the appeal of the stylization. Specifically, thin geometric structures suffer from these undesirable fuzzy geometric changes. The limit-free optimization strategy of *S2* fiddles with the geometry components and leads to these artifacts. This can be observed in the insets provided in Fig. 3.7.

Our StyleTRF (*S3*) approach on the other hand alleviates these issues by freezing the geometric components of scene representation. Our approach generates crispier results compared to the rest of the aforementioned strategies *S1, S2*. This behavior is consistent across all the scenes.

We experiment with a different number of stylized priors to stylize the underlying scene. As shown in Fig. 3.6, we find that good stylization is obtained when all the priors cover the entirety of the scene. In most cases, 30-40 randomly sampled camera positions around the object suffice.

3.4 Results

In this section, we compare our results against both 3D stylization techniques and temporally-consistent stylization techniques both quantitatively and qualitatively.

3.4.1 Qualitative Results

For comparing qualitative results, we can use smooth trajectories which are similar to a video. This similarity enables us to compare with temporally-consistent stylization techniques alongside 3D-

(a) 5 Priors (b) 10 Priors (c) 20 Priors (d) 40 Priors

Figure 3.6: *Number of Stylized Priors*: The images in this figure are obtained by fine-tuning StyleTRF using a different number of priors on the kitchen scene from the MipNeRF360 [3] dataset which is an **unbounded 360 degree** scene. Depending on priors quality improves and saturates at 35-40 priors.

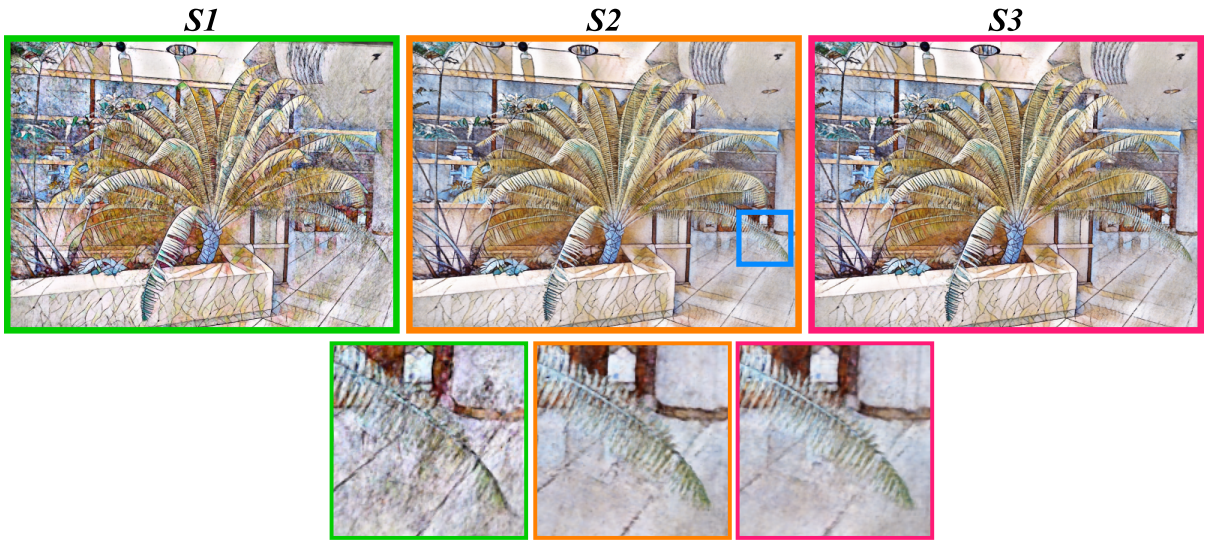


Figure 3.7: *Comparison of the different optimization strategies:* Strategy *S1* optimized directly on stylized images produces massive artifacts due to the loss in geometry. Unconstrained optimization *S2* on the other hand achieves results compared to *S1* while struggling to capture micro-details as seen in the insets. Our StyleTRF on the other hand freezes density while only optimizing for appearance, capturing the style while maintaining the geometric detail of the original scene.

Stylization. For 3D stylized novel view synthesis, though there exists [18], they do not hold an accurate representation of geometry and rely on explicit geometry as input. Although other methods like [7] do not rely on explicit geometry input, they suffer from patch border artifacts as discussed in [19] and Sec. ??. Hence, we compare our results with the latest 3D radiance field-based stylization method StylizedNeRF [19] and temporally-consistent stylization technique of ReReVST [50], on similar lines as [19].

Comparison amongst different stylization Techniques: We have observed that though the novel views generated using StylizedNeRF are reasonably consistent, the geometry after stylization in the case of StylizedNeRF has been greatly affected as seen in Fig. 4.3(c). It produces blurry geometry in the case of TREX stylized using *udnie* along with the missing greenish tints present in the *udnie*, and produces extremely noisy results for *mediterranean* applied on FERN . Contrary to this, our method distinctively transfers different colors to different parts of the scene as seen in TREX stylized using *udnie* and captures the complete color palette in *mediterranean* adapted onto FERN . Our method also consistently preserves geometry across all scenes and styles. The geometric noise in the StylizedNeRF can partly be attributed to the combined density and appearance encoded of a single MLP, which hinders the disentanglement of geometry and radiance as observed by [35].

In the case of the temporal-consistent style transfer technique ReReVST, we have observed that although the method is robust to adapt to new styles on the fly and produces results in real-time, it lacks proper capture of style information. This can be observed Fig. 4.3(b), ReReVST fails to capture the prevalent blue tint in *santamaria* in the case of FLOWER , this could also be observed in the case of

	GT Training	Style Training	Style Adaptation
StyleImp	$\approx$ 12 hrs	NA	$\geq$ 5 hrs
SNeRF	$\approx$ 12 hrs	NA	3-4 days
StylizedNeRF	$\approx$ 12 hrs	NA	$\geq$ 3 hrs
Ours	20 mins	20 mins	45 secs

Table 3.2: *Training Times:* We compare our training times with StyleImp [7], SNeRF [35], StylizedNeRF [19]. Due to our compact representation, we get a training time of $\approx$ 20 mins compared to at least 12 hrs for others. The GT training and style training can be performed in parallel in our method. We disentangle the style training process from the style adaptation process which enables us to quickly adapt to any scene in under a minute.

StylizedNeRF and vaguely captures the color palette of *mediterranean* in FERN . For *udnie* applied on TREX , it does not account for distinct colors present in the style and the resultant image contains an unappealing blend of colors throughout the image.

View Consistency: To qualitatively show view consistency, we render views across a smooth trajectory and show different views across it. In Fig. 3.4 we show our renders for *udnie* and *mosiac* styles adapted onto STOVE and HORNS respectively. It can be observed from the insets of Fig. 3.4 that stylized radiance across the frame is consistent, validating our claim that 3D-stylization-based novel view generation can produce multi-view consistent stylized content.

3.4.2 Quantitative Results

In order to check the consistency of our stylized content, we generate a smooth trajectory similar to SNeRF [35]. The rendered views along the trajectory are used to evaluate the consistency. Since the smooth trajectory replicates the behavior of a video we can comfortably compare our method with the temporal-consistent stylization techniques like ReReVST [50] alongside 3D-stylization methods like StylizedNeRF [19]. For this comparison, we chose FERN , FLOWER , and HORNS scenes and report the respective metrics in Tbl. 3.1. We obtained better metrics compared to both temporal stylization [50] and implicit geometric stylization [19]. We estimated the consistency by calculating the optical flow $\mathcal{O}$ between the non-stylized frames $\mathcal{F}_i^{real}$ and $\mathcal{F}_{i+\delta}^{real}$ rendered using TensoRF.

$$\mathcal{O} : optical flow \left(\mathcal{F}_i^{real}, \mathcal{F}_{i+\delta}^{real} \right) \tag{3.4}$$

Using the so-obtained optical-flow O, we warp the stylized frame $\mathcal{F}_i^{style}$ to $\hat{\mathcal{F}}_{i+\delta}^{style}$.

$$\mathcal{W} : \hat{\mathcal{F}}_{i+\delta}^{style} \leftarrow Warp(\mathcal{F}_i^{style}, \mathcal{O}) \tag{3.5}$$

For the calculation of optical flow, we use RAFT [46] similar to SNeRF.

We then calculate the pixel-averaged L_2 loss between the warped frame $\hat{\mathcal{F}}_{i+\delta}^{style}$ and the actual stylized frame $\mathcal{F}_{i+\delta}^{style}$. We aggregate this loss across the frame-combinations and report the metrics in Tbl. 3.1. For calculation of *short-term* consistency we chose $\delta = 1$ and for *long-term* $\delta = 5$. It is to be noted that

δ here represents the change in the camera position along the trajectory. We have observed that though the *temporal-cosistent* stylization techniques like [50] come close to our results, they struggle to capture the reference-style (as seen in Fig. 4.3(b)).

3.5 Conclusion

In this paper, we presented StyleTRF, a compact and quick-to-optimize stylization technique which can generate stylized novel views of a scene. We have shown that our method can efficiently and faithfully incorporate style into a radiance field representation of a casually captured scene. We have qualitatively and quantitatively compared our method with the previous stylization methods. Our qualitative results and quantitative metrics demonstrate that StyleTRF is consistent across the views, having stylized the underlying 3D representation. We also reported the *short and long-term* consistency metrics which are better in most cases compared to the present 3D stylization methods.

Concurrently, Zhang et al. [53] presented stylized novel view synthesis using voxel-based grids, partly similar to our method. But they chose PlenOxels [10] to represent the radiance fields. They focussed more on obtaining brush strokes while ours concentrates on geometric preserved fast-style adaptation.

Chapter 4

Interactive Segmentation of Radiance Fields

Figure 4.1: We present ISRF, an interactive method to segment objects in radiance fields. Users can draw positive strokes to segment multiple objects at a time in 3D and negative strokes to remove unwanted regions repeatedly. In the figure, the WOODEN TABLE TOP is segmented using one positive and one negative stroke as shown.

Radiance Fields (RF) are popular to represent casually-captured scenes for new view synthesis and several applications beyond it. Mixed reality on personal spaces needs understanding and manipulating scenes represented as RFs, with semantic segmentation of objects as an important step. Prior segmentation efforts show promise but don't scale to complex objects with diverse appearance. We present the ISRF method to interactively segment objects with fine structure and appearance. Nearest neighbor feature matching using distilled semantic features identifies high-confidence seed regions. Bilateral search in a joint spatio-semantic space grows the region to recover accurate segmentation. We show state-of-the-art results of segmenting objects from RFs and compositing them to another scene, changing appearance, etc., and an interactive segmentation tool that others can use.

4.1 Method

We first provide the basics on radiance fields and the feature distillation strategy related to our scene representation. We then detail our proposed interactive segmentation workflow comprising 2D-3D feature matching, region growing, and manipulation techniques on this learned representation.

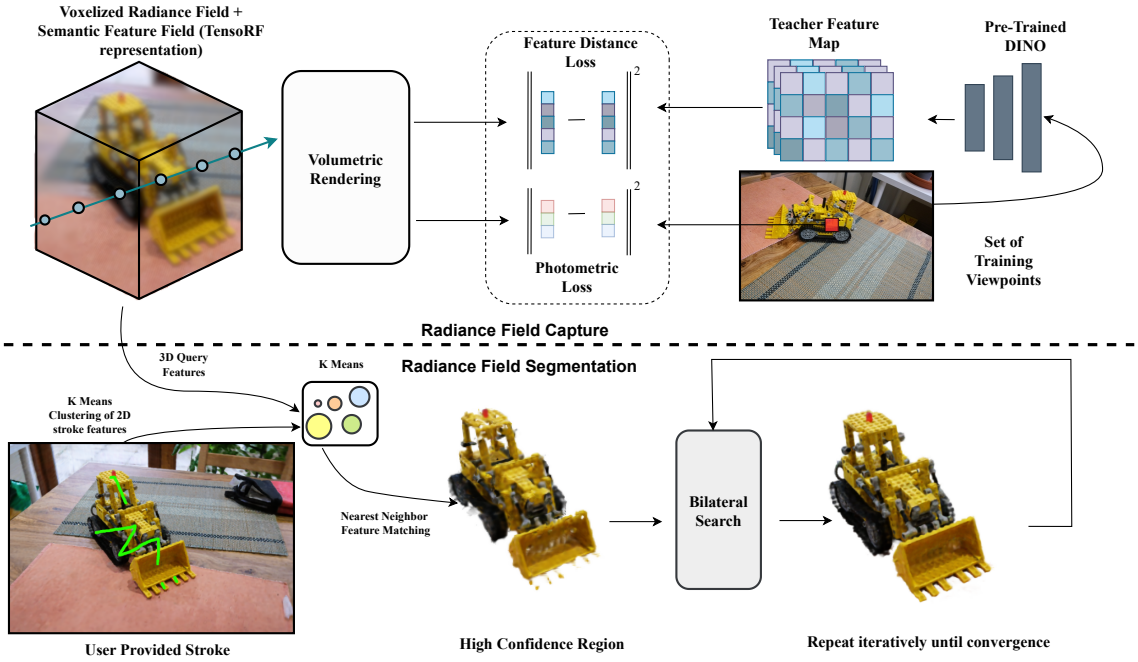


Figure 4.2: *ISRF System overview*: We capture a 3D scene of voxelized radiance field and distill the semantic feature into it. Once captured, the user can easily mark regions using a brush tool on a reference view (green[■] stroke). The features are collected corresponding to the marked pixels and clustered using K-Means. The voxel-grid is then matched using NNFM (nearest neighbor feature matching) to obtain a high confidence seed using a tight threshold. The seed is then grown using bilateral search to smoothly cover the boundaries of the object, conditioning the growth in the spatio-semantic domain.

4.1.1 Radiance Field Representation

A radiance field [56] $\mathcal{F}$ maps the scene radiance values as view dependent RGB color $c \in \mathbb{R}^3$, given a continuous point $x \in \mathbb{R}^3$ and viewing direction $d \in \mathbb{S}^2$ in space as inputs: $\mathcal{F}(x, d) : \mathbb{R}^3 \times \mathbb{S}^2 \rightarrow \mathbb{R}^3$.

NeRF [33], and its variants [3, 31, 54] encoded this mapping as the neural function using an MLP, with a low memory footprint but high training and rendering overhead. They also store scalar point density $\sigma \in \mathbb{R}$ which is used for differentiable volumetric rendering to train the network:

$$\hat{C}(r) = \left(\sum_{i=i}^{K} T_i \alpha_i c_i\right) \qquad \text{where} \tag{4.1}$$

$$\alpha_i = 1 - e^{-\sigma_i \delta_i} \qquad \text{and} \qquad T_i = \prod_{j=1}^{i-1}(1 - \alpha_j). \tag{4.2}$$

Here for a given point i along a ray, δ_i is the distance to the sampled point, T_i is the accumulated transmittance, and c_i is the view-dependent color for the point.

Later efforts like Plenoxels [10], and DVGO [42] stored the field variables in a lattice structure akin to a 3D voxel grid, significantly improving the training and rendering times at the cost of high storage requirements. These quantized values are trilinearly interpolated and decoded to render color value at

any point. The grid structure provides easy spatial context and explicit representation leading to higher efficiency. Recently, TensoRF [6] proposed a matrix-vector decomposition representation of this lattice, reducing storage requirements while facilitating efficient training and view generation.

We use TensoRF as the basis of our work. The top part of Fig. 4.2 shows our radiance field capture step, with the volume represented using TensoRF. In the case of the quantized representation of radiance fields, the radiance is obtained as follows:

$$\sigma_i = \psi(V^\sigma, x_i) \quad \text{and} \quad c_i = \mu_\theta^{rbg}(\psi(V^f, x_i), d). \tag{4.3}$$

Here σ is the density of the volumetric space, V^f the radiance feature grid of appearance features f, and ψ indicates trilinear interpolation. While rendering a given sample point $x_i \in \mathbb{R}^3$ along the ray direction d, a small decoding MLP $\mu_\theta^{rbg}(f_i, d) \rightarrow c_i$ is evaluated. The final color of a ray is calculated by combining all sample colors c_i at every point x_i along it using the Eq. (4.1). This is used to reduce the photometric loss $\mathcal{L}^{(rgb)}$ optimizing for both the radiance feature lattice V^f and parameters θ of MLP (μ).

4.1.2 Semantic Features Distillation

Object segmentation requires knowledge of scene semantics. We include an additional feature into the radiance field for this. In order to attribute semantics to the radiance field, we distill contextual knowledge from a large pre-trained teacher model similar to the prior art [24, 48]. Specifically, our teacher is a vision transformer model trained using self-supervision and is shown to pay attention to semantically meaningful objects in the scene in a class-agnostic manner. This knowledge from the teacher is distilled into the student radiance field in addition to the color and density values as point semantic features $\phi \in \mathbb{R}^m$. Thus the mapping now becomes: $\mathcal{F}(x, d) : \mathbb{R}^3 \times \mathbb{S}^2 \rightarrow \mathbb{R}^3 \times \mathbb{R} \times \mathbb{R}^m$.

More concretely, we use 2D semantic features using the DINO ViT-b8 model [5] for each input posed image.

Recent efforts [24, 48] also use DINO; unlike them, we directly optimize for the features on the voxel grid in the TensoRF representation without a neural network. We also do not encode the direction dependence in these semantic features since the object semantics are direction agnostic. We trilinearly interpolate the distilled semantic feature $\phi_i = \psi(V^\phi, x_i)$ for a point x_i from the learned feature lattice V^ϕ. We combine the ϕ_i along the ray using the Eq. (4.1) like color c_i. The TensoRF representation is optimized to minimize the total loss

$$\mathcal{L} = \mathcal{L}_{rgb} + \lambda \mathcal{L}_{feature} \tag{4.4}$$

to obtain the final radiance field with ϕ, V^f, and V^ϕ. Both losses $\mathcal{L}_{rgb}$ and $\mathcal{L}_{feature}$ are calculated using L^2 norm.

High-resolution feature rendering results in high-frequency feature fields similar to N3F [48]. (See the supplementary document for distilled feature field visualizations.)

Explicit semantic features at every point open the way to adapt traditional 3D analysis techniques to radiance fields in a semantically meaningful fashion. Segmenting objects in 3D voxel space and using bilateral filtering inspired search are examples that go beyond what prior neural representations have shown.

4.1.3 2D-3D Feature Matching

For object segmentation, the user picks a(few) reference views and annotates the regions of interest using a brush stroke. Semantic DINO features associated with the marked pixels are collected.

DINO features were shown to fare well using 1-NN feature matching for good 2D semantic segmentation [5]. However, a single DINO feature will not suffice to segment complex objects with diversity. We cluster the input features using K-Means to obtain a fixed-size exemplar set of features for matching in 3D space. We use nearest neighbor feature matching (NNFM) on the exemplar set to label each voxel as foreground or background. The result is stored in a 3D bitmap. In this step, we use a tight threshold to identify a high-confidence seed region, which is processed further. Prior methods [24, 48] used a single averaged semantic feature from the user-specified patch to match 2D to 3D. Their implicit neural representation can only be segmented after ϕ values are rendered. Feature matching methods like NNFM are too *costly* to evaluate at every point on the ray using a neural representation.

The segmentation results can also be precomputed and stored, facilitating downstream tasks like view generation and editing on the fly without repeated processing.

4.1.4 Region Growing

The high confidence seed region (M^0) from the previous step is grown in the volume-space to delineate the complete object volume. We do this in joint spatio-semantic space to include proximate voxels that are also semantically close. We adopt a *Bilateral Filtering* [47] inspired search dubbed as *Bilateral Search* on the voxel grid using the spatial feature x and semantic feature ϕ values as filter's domain and range kernels, respectively.

We iteratively grow the current bitmap region M^r till convergence, as given below:

$$M^{r+1}(x) = \mathcal{T}_\tau \left(\frac{1}{W} \sum_{x_i \in \Omega_x} M^r(x_i)\, g_{\sigma_\phi}(\phi_i^2)\, g_{\sigma_s}(s_i^2) \right)$$

$$\text{where} \quad \phi_i = \|\phi_{x_i} - \phi_x\|, \qquad s_i = \|x_i - x\|$$

$$\text{and} \quad W = \sum_{x_i \in \Omega_x} g_{\sigma_f}(\phi_i^2) g_{\sigma_s}(s_i^2).$$

Here M^r is the r^{th} iteration of filtering; ϕ_x is the distilled semantic feature at point x in the volumetric space; g_σ is the Gaussian smoothing functions with variance σ; $\mathcal{T}_\tau$ is binary thresholding against value τ; and Ω_x is the immediate voxel neighbors of x.

We find that $\tau = 0.2$ works well for our scenes. The seed region expands to the boundaries of the desired object in a few iterations of bilateral filtering.

4.1.5 User Interactivity

Region growing results in a stable voxel content based on the input strokes. The user can add or remove parts interactively if the extracted content misses out on a few details or when some extraneous content floods into the segmented region. We use positive and negative strokes to add and remove the content in the image space, as followed by methods like GrabCut [39]. The mask of the negative segment is subtracted from the mask of the positive segment to get the final segmented objects. We find practically that even complex objects can be segmented well with a few positive and negative strokes, as shown in the results in the paper and in the supplementary material. Additionally, our method provides interactive feedback for every stroke (as can be seen in Tab. 4.3) that allows users to segment interactively unlike methods like NVOS [38]. Implementation details have been reported in the supplementary document.

4.2 Implementation Details

Our system is implemented using PyTorch [36] branching off the code provided by DVGOv2 [43]. All experiments are performed using a commodity hardware equipped with AMD Ryzen 5800x and a NVIDIA RTX 3090.

The feature components of the radiance fields namely radiance latent vectors and the learnt DINO features are stored using VM decomposition proposed by TensoRF [6]. For radiance latent vectors, we use *VM-48* representation of TensoRF and for DINO features, we use *VM-64* variant of TensoRF. The segmentation masks and densities are stored as a full voxel grids.

The DINO *ViT-b8* [5] model provides 768 features for each patch of 8×8 pixels in an image. We reduce DINO features to 64 by doing a principal component analysis. This is consistent with the prior works [24, 48]. For each pixel, The feature of a pixel is the same as the feature of the patch it belongs to.

We first pre-train the model for the volumetric density and radiance for $20,000$ iterations. Once the radiance field is stabilized on the VM-48 TensoRF representation, we introduce distillation using *student-teacher* strategy similar to that of [24, 48] on the VM-64 TensoRF variant. Upon adoption, the resultant VM-48 variant of TensoRF along with its shallow MLP represents the radiance field, and VM-64 consititute the distilled features. It is to be noted that the distilled features are not accompanied by a shallow MLP. The features are store at voxel lattice locations and tri-linearly interpolated to be compared and optimized against the DINO features without involving any non-linearity. The adoption

Figure 4.3: *Our ISRF vs N3F/DFF [24, 48]:* Both N3F and DFF employ a similar strategy for segmentation. We tweak the threshold for their method and bring out the best results and show their respective results in the Row 2. Row 3 shows our results with the same queried patch (highlighted in green[■] in Row 1). Since our method works best on user provided strokes (shown in yellow[■] in Row 1), we show the corresponding results in Row 4. While N3F/DFF are able to recover simpler objects like COLOR FOUNTAIN , they fail to capture other objects. Our method faithfully recovers the queried objects with clear and smooth boundaries. For more details, please refer to Sec. 4.1.4.

is done with $\lambda = 0.001$ for the weighted loss function for $5,000$ iterations. The loss is taken on the features and radiance together to maintain consistency.

We chose $K = 10$ when applying K-Means to the set of features selected from the user's brush stroke. For the bilateral search, the value of σ_ϕ and σ_s are set to 10.0 and the 1.0 respectively while the threshold value τ is 0.1.

4.3 Results

In this section, we discuss the comparisons and results of our proposed method against the existing semantic features distillation-based Radiance fields segmentation approaches. Specifically, we focus

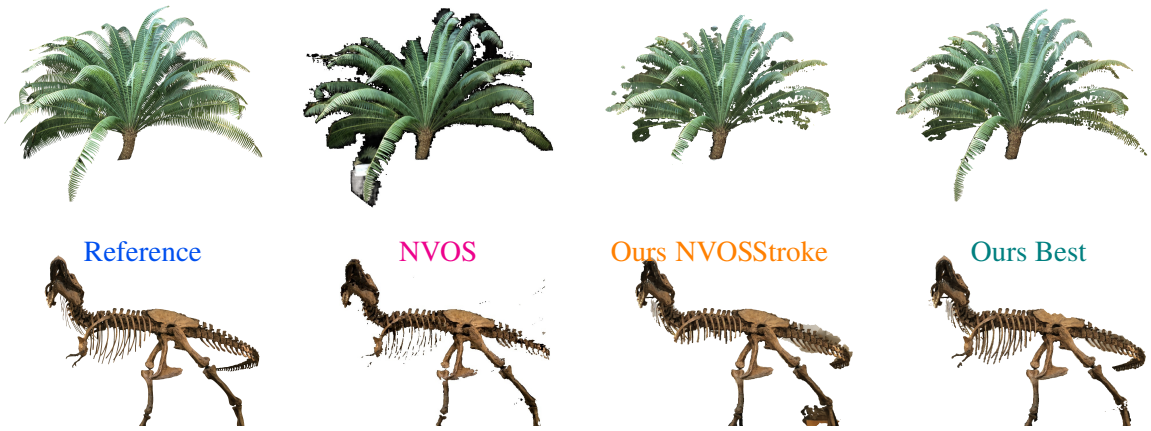


Figure 4.4: *left to right:* Reference segmentation using NVOS professionally segmented mask, Result of NVOS [38], Our result using NVOS stroke, Our result using additional strokes. The quantitative comparisons are mentioned in the main document where our method performs better than NVOS even when using NVOS strokes. Please zoom using *Adobe Acrobat/Okular* reader to see the details.

on the two recent approaches: DFF-DINO [24] and N3F [48]. Both use extracted features from input images and fuse them into the volumetric space. DFF additionally concentrates on the language queries using LSeg [28], but both approaches are similar regarding semantic features. As the code of DFF is not publicly available, we compare our method against N3F, which is similar to DFF for this part.

4.3.1 Comparison

As discussed earlier, our approach supports region selection either by a patch or a hand-drawn brush stroke as shown in Fig. 4.1. To obtain the desired volumetric content, we follow the methods described in the Sections (Secs. 4.1.3 and 4.1.4). Fig. 4.3 shows our segmentation results on a few challenging scenes.

The usage of clustering followed by NNFM clearly outperforms the prior approach of average matching [24,48]. The direct incorporation of nearest neighbor feature matching (NNFM) in these approaches leads to significant rendering delays, while the choice of neural space limits them from using elegant techniques like bilateral search.

In Fig. 4.3 it can be observed that in the case of the COLOR FOUNTAIN , the simple average feature matching technique faithfully recovers the region of interest albeit with some additional noise. However, as the scene's complexity and region of interest grows, the prior art fails to garner pleasing results. This can be observed clearly in the case of the three LLFF [32] scenes (CHESS TABLE , SHOE RACK , STOVE). When only simple averaging is employed, the CHESS TABLE scene suffers due to the erroneous feature matches. The clustered matching mitigates the errors and confines the segmented volume to the TABLE. A similar effect can be observed in the case of STOVE where the object of interest is sparingly covered in the input images but is faithfully recovered with distinct boundaries, unlike N3F. The last scene SHOE RACK is a *classic* example where recovering white-sole might be challenging even with

Scene	Metric	N3F	Ours (Patch)	Ours (Stroke)
CHESS TABLE	Mean IoU ↑	0.344	0.864	**0.912**
	Accuracy ↑	0.820	0.985	**0.990**
	mAP ↑	0.334	0.874	**0.916**
COLOR FOUNTAIN	Mean IoU ↑	0.871	**0.927**	**0.927**
	Accuracy ↑	0.979	**0.989**	**0.989**
	mAP ↑	0.871	**0.927**	**0.927**
STOVE	Mean IoU ↑	0.416	**0.827**	0.819
	Accuracy ↑	0.954	**0.992**	**0.992**
	mAP ↑	0.387	**0.824**	0.817
SHOE RACK	Mean IoU ↑	0.589	0.763	**0.861**
	Accuracy ↑	0.913	0.965	**0.980**
	mAP ↑	0.582	0.773	**0.869**

Table 4.1: This table denotes the Mean IoU (Intersection Over Union), Accuracy and Mean Average Precision measurements for the four LLFF scenes shown in the main paper. The ground truth segmentation masks have been hand-annotated for comparison.

NVOS		**Ours(NVOS Stroke)**		**Our best**	
mIOU	mAcc	mIOU	mAcc	mIOU	mAcc
70.1	92.0	83.75	96.4	90.8	98.2

Table 4.2: Quantitative metrics(*mIOU and mAcc)* of NVOS against Ours using NVOS provided strokes and additional strokes using our interactive feedback tool

the best feature matching scheme. This is where the bilateral search helps in exploiting multi-domain content by conditioning on the spatio-semantics.

To quantitatively compare our method on the LLFF Dataset [32], we hand-annotate the segmentation masks for the prominent objects in the CHESS TABLE , COLOR FOUNTAIN , STOVE and SHOE RACK scenes. Tab. 4.1 reports the segmentation metrics for the four scenes. In our method, to predict the segmentation mask, we threshold α to be greater than 0.1 while rendering. This removes the low volumetric density seeping in that contribute negligibly in the rendered visuals. Quantitative evaluation of *mIOU/mAcc* scores on all the NVOS dataset also reflect similar behavior. Using the input strokes and GT masks of NVOS, we obtain an *mIoU* of **83.75%** (compared to **70.1%** of NVOS) and an *mAcc* of **96.4%** (compared to **92.0%** of NVOS), on the same LLFF dataset. Additionally, our interactive scheme allows for improving the segmentation in subsequent iterations. We achieved an *mIoU* of 90.8% and an *mAcc* of 98.2% on the same dataset using multiple strokes as shown in Tab. 4.2.

4.3.2 Interactive Segmentation with User Strokes

Our method allows both adding and removing content using positive and negative strokes. The cases where the single stroke fails to obtain the desired content in the extracted space, the user can add another

Figure 4.5: *Feature Matching*: This figure shows the high confidence region of the RF (a) obtained using different feature-matching techniques for a particular stroke. While *Average feature matching* (b) fails to cover the entire object due to loss of information during the averaging process, *NNFM* (c) without clustering leads to noise bleeding. Use of *NNFM after clustering* (d) eliminates noisy regions while also considering multiple features at once.

positive stroke to add more content. Fig. 4.8 shows one such example where the excavator ('JCB') has missing teeth in the extracted region. Drawing an additional stroke and bilaterally growing the region again brings out the full desired result. This effectively grows the bit-map M^r by segmenting more desirable regions from the volumetric space.

Similar to adding new content, some scenarios demand the need to remove extraneous content from the extracted region. In such scenarios, we mark the region to be removed and grow it independently of the positive content. Once fully grown, the full extent of the negative/undesirable content is obtained which we subtract from the previously extracted regions obtaining the edited bit-map M^r. Fig. 4.1 shows one such example where the REFLECTIVE GRANITE floods into the TABLE region. We add a negative stroke (red) to remove this undesired region.

Incorporating these functionalities is not trivial in the case of the prior art, as an additional negative match or a positive match calculation at the time of rendering is a tedious task.

4.4 Experiments

In this section, we discuss various feature-matching variants we used to obtain the high-confidence seed region. Additionally, we show some immediate applications of radiance field segmentation.

Step	Time Taken
Pre-training radiance field	7 mins
Training feature field	2.5 mins
K-Means Clustering	2 secs
3D Feature Query	1 secs
Bilateral Region Growing	0.3 secs

Table 4.3: Timings of different steps of the ISRF pipeline

4.4.1 Ablations

In order to obtain a high-confidence region, which acts as a seed for the bilateral filter, a feature-matching technique is required to match the marked features with the distilled semantic features in the volumetric space. To this end, we experimented with three different feature matching techniques, namely (1) Average Feature Matching, (2) Nearest neighbor Feature matching(NNFM) (3) K means + NNFM, which are compared in the Fig. 4.5. It can be easily inferred from Fig. 4.5 (b) that average feature matching performs poorly in this task. In order to improve these results, we resort to the nearest neighbor feature matching. Though this recovers a good high confidence region, it is accompanied by additional noise as seen in Fig. 4.5 (c) Furthermore, as the marked region's size grows, computation also becomes tedious in this case. To address this, we cluster the features using K-means clustering and then do an NNFM that reduces computational overhead and avoids noisy matches, as seen in Fig. 4.5 (d). When $K = 1$, clustering results in mean features of the selected stroke, and as K approaches high values, the search approximates NNFM.

4.4.2 Scene Editing

In this section, we explain the procedures ISRF follows to edit the 3D scenes post segmentation. The segmentation provides a 3D bitmap representing the segmented voxels. The bitmap also assists in faster rendering as the voxels with segmentation mask values of 0 can easily be filtered out. Fig. 4.6 shows the additional results of scene editing.

Object Removal: To remove a segmented object from the scene, we alter the evaluation of the density for a 3D point. We simultaneously evaluate the bit map value b_x at the queried point. The effective density σ'_x while rendering is $\sigma_x * b_x$ for the segmented foreground objects. The effective density σ'_x of background is $\sigma_x * (1.0 - b_x)$.

Translation: If an object is moved to another location, the ray queries lying inside the object's voxel space can be shifted to the desired location. If t is the translation vector for the object to be moved, the

(a) Removal (b) Translation (c) Scene Composition (d) Appearance Editing

Figure 4.6: *Scene manipulation*: Segmented object(s) can be edited in different ways. In (a), we remove the POT from the center of the table. In (b), we translate the POT to the GROUND behind the TABLE. In (c), we replace the POT with the LEGO JCB obtained from a different scene (KITCHEN). We stylize the newly added LEGO JCB using [12] in (d). All scenes are from [3].

(a) Rendered Image

(b) HCR

(c) Iteration 1

(d) Iteration 2

Figure 4.7: *Region Growing*: Image (a) is the reference rendered viewpoint. Image (b) is the high confidence region which misses out frontal region of the dry-leaf when extracting the content. Image (c) shows the result obtained after the first iteration of bilateral filtering, which captures most of the desired region of the leaf. Image (d) is the result of the bilateral filtering applied for the second time to include intricate details such as strands around the dry-leaf.

object's ray-point query values are given by:

$$\sigma'_x, rgb'_x = \sigma_x, rgb_x \; \forall \, b_x = 0$$
$$\sigma'_x, rgb'_x = \sigma_{x+t}, rgb_{x+t} \; \forall \, b_x = 1,$$

where b_x is the bitmap value for the point x.

Scene Composition: To perform scene composition, we follow a similar strategy used by D^{2}NeRF [51]. We alter the volumetric rendering equation to account for density and color from both the scenes as shown below:

$$\hat{C}(r) = \int_{t_n}^{t_f} T(t) \left(\sigma_1(t)c_1(t) + \sigma_2(t)c_2(t)\right) dt$$
$$T(t) = exp\left(-\int_{t_n}^{t} (\sigma_1(s) + \sigma_2(s))\right) ds$$

(a) Stroke 1

(b) Output 1

(c) Stroke 2

(d) Merged Output

Figure 4.8: *Multiple Positive Strokes*: When the method fails to capture some of the details using initial set of strokes, the user can iteratively add more positive strokes to recover the desired object. (a) depicts the initial strokes which lead to missing teeth as shown in (b). Addition of a small stroke on one of the teeth (c) and followed by region grown captures full-details as shown in (d).

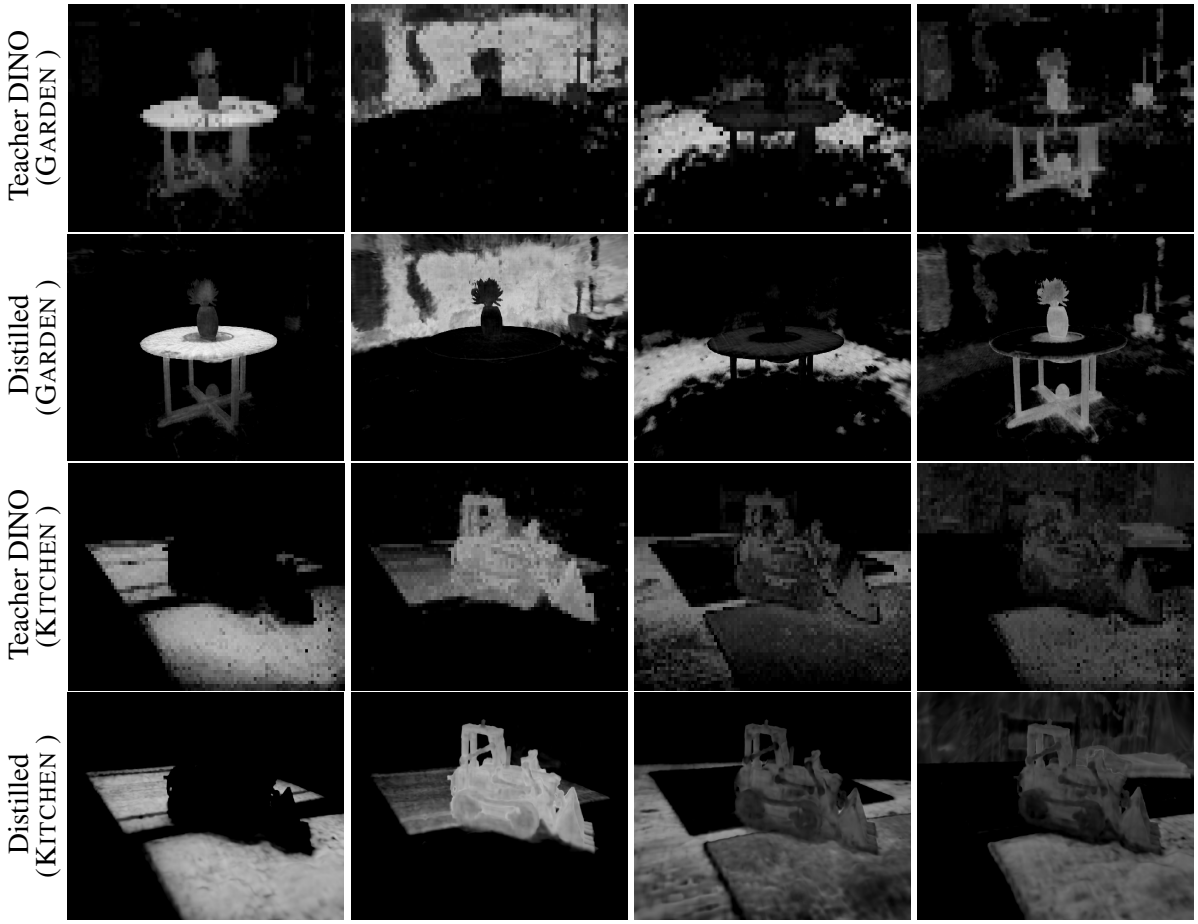


Figure 4.9: *Student Surpasses Teacher*: The 4 columns of this figure shows the DINO features used as teacher vs the ones learnt by student post optimization. Since, the student learns finer features than the teacher due to assistance from the volumetric density, we can claim that the student surpasses the teacher. This is consistent with the prior art N3F and DFF.

The results for scene composition have been shown in the main paper and Fig. 4.6 of the supplementary.

Appearance Editing: Here, we apply style transfer on an already composed scene. We first calculate a 3D bitmap for the JCB lego in the KITCHEN scene. Then, we generate a new set of stylized training images using the method proposed by [12, 20] using a reference image. The appearance latent vectors and the rendering MLP is fine-tuned according to the new training images while keeping the density and feature weights frozen. This transfers the style from a reference image to the 3D object.

4.4.3 Interactive Segmentation

Our method provides interactive segmentation capabilities to the user with the incorporation of positive and negative brush strokes similar to GrabCut [39].

(a) Rendered View 1 (b) Rendered View 2 (c) Segmented Trex (d) Depth Map

Figure 4.10: *Finer Segmentation*: Images (a) and (b) show rendered views of T-Rex from the LLFF dataset [32]. Image (c) shows the segmented output of T-Rex scene. Our method achieves fine-grained segmentation of objects such as the rib-cage bones of T-Rex. However, on close observation, the region near the tail bones background bleeds in. This is due to the wall and the tail-bone lie at the similar depth as shown in the depth map (d). This can be mitigated by having more 3D information (better training views) or higher voxel grid resolution.

Upon the addition of a new positive stroke, a new segmentation mask b_p is calculated using the procedure described in the main paper. The user has the option to grow this new region using bilateral filtering until not required. The new segmentation mask b_{new} is given by $b \cup b_p$.

When the user adds a negative stroke, a new segmentation mask b_n is calculated. Similar to a positive stroke, the user has the option to grow this region using bilateral filtering until not required. The new segmentation mask b_{new} is given by $b \cap (b \cap b_n)'$ (X' denotes the complement of X).

4.4.4 Critical Analysis

DINO Features: The teacher DINO features calculated on the training set of images are for patches of size 8x8. This method associates a total of 64 pixels to the same feature vector. As shown in Fig. 4.9, the teacher features appear to be in low resolution due to this. When performing the teacher-student training using the joint loss function, the features learnt by the student are finer in detail due to assistance from volumetric density. Hence, the student surpasses the teacher during distillation. This is evident from Fig. 4.9 as features are allocated with distinct boundaries in the voxel space.

Finer Segmentation: Our method can segment out fine-grained details such as the ribs of a T-Rex as shown in Fig. 4.10. However, it requires accurate 3D information to achieve this. In the T-Rex scene, the tail-bones cannot be distinguished from the wall behind, since the training set images do not cover views which indicate the separation. Therefore, the optimized model containing the wall and the tail bones lie at similar depths as shown in Sec. 4.3. Use of additional images covering more viewpoints can circumvent this issue.

4.4.5 Discussions and Limitations

Our method improves upon the prior art on several fronts but has its own shortcomings. Like prior works, we rely on DINO features to represent object semantics and this can result in artefacts if the features do not capture the semantics properly. Third column in the last row in Fig. 4.3 shows a small

false appendage at the bottom of the utensil holder which can not be easily removed interactively without eating into object's body. Better semantic features can resolve this problem. Also, the leftmost example in Fig. 4.6 shows that the shadow of the pot is left behind on the granite center of the table even after the pot is edited out. Removing the pot from the geometric representation does not guarantee removal of its secondary effects on neighbouring objects like shadows or highlights, without elaborate geometric post-processing. Our method may also struggle in segmenting geometry well if the voxel resolution is low compared to the scale of object details as shown in supplementary results. Multiresolution voxel representations can solve this problem with additional overhead.

4.5 Conclusions and Future Work

In this paper, we presented an easy and accurate method to segment objects from a TensoRF representation of radiance fields and showed simple scene editing operations facilitated by this. The efficient voxel-based representation we use makes our method more versatile and simple compared to the prior works in this direction. We show several results on multiple challenging scenes (and present more in the supplementary document). Semantic segmentation is a first step towards interpretation, understanding, and manipulation of 3D scenes. This work provides high quality segmentation that can be the basis for several such downstream tasks. A simple extension to the current method would be to generalize the distance used for matching in the NNFM and region-growing steps to include other featurs like color latent vectors. Extending the current method to a InstantNGP [34] framework, while incorporating additional multi-domain explorations strategies like guided filtering [17] would be a good direction to explore.

In the future, multi-representation processing might be needed by combining parts of captured RFs, graphics models, SDFs, etc., to provide maximum flexibility in Virtual Reality and Augmented Reality applications. This requires processing parts of the RFs directly without going through the full learning process post-editing. This is a promising direction of work that we intend to pursue in the future.

Chapter 5

FusedRF: Fusing Multiple Radiance Fields

Figure 5.1: *NeRF-Feast:* Our method can be used to create a single composed representation. Here, we show a feast prepared on the table from the garden scene from MipNeRF-360 [3]. Since we have a single fused representation, the scene is rendered at the cost of a single RF representation while also occupying a memory footprint of a single RF. Scene components are picked from [1].

Radiance Fields (RFs) have shown great potential to represent scenes from casually captured discrete views. Compositing parts or whole of multiple captured scenes could greatly interest several XR applications. Prior works can generate new views of such scenes by tracing each scene in parallel. This increases the render times and memory requirements with the number of components. In this work, we provide a method to create a single, compact, *fused* RF representation for a scene composited using multiple RFs. The fused RF has the same render times and memory utilizations as a single RF. Our method distills information from multiple teacher RFs into a single student RF while also facilitating further manipulations like addition and deletion into the fused representation.

5.1 Method

Chapter 5 shows an overview of our FusedRF method. The following through Secs. 5.1.1 to 5.1.3 will detail the compositional, fusion, and convergence aspects of the proposed methodology, respectively.

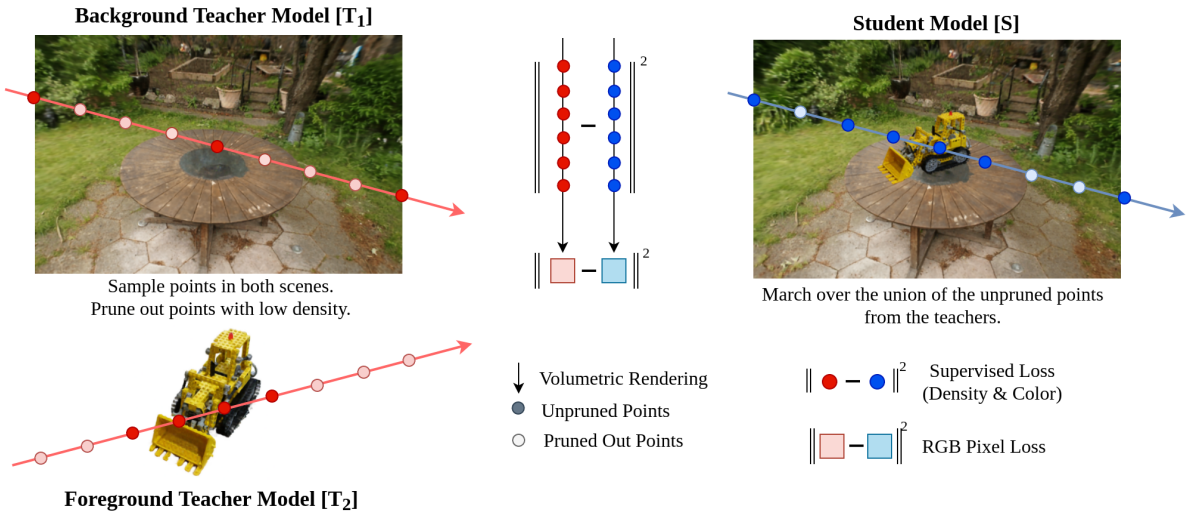


Figure 5.2: *FusedRF:* Method to fuse multiple RFs into single presentation. We shot the same ray into both teachers T_1, T_2 (left) created using ISRF [13] to distill a combined Student S(right). For every sampled point on a ray of both RFs we prune out ones with lower densities and apply supervised distillation losses to obtain faster convergence. Followed by a few iterations of Pixel Loss on alpha-composited RGB for smoothening. By fusing multiple RFs into a single representation, we reduce rendering and memory overheads of composition. Both scenes GARDEN and LEGO are picked from the MipNeRF-360 [3] dataset.

5.1.1 Composition of Radiance Fields

The composition of two distinct radiance fields can be performed by altering the compositional aspect of the volumetric rendering equation [33, 42]. For simplicity, let's assume we have to composite only two radiance fields (RF^1 and RF^2). Every ray is shot and sampled similarly in both RFs. For every sampled point along the ray, the activated volumetric density ($\alpha : \alpha^I = 1 - e^{-\sigma^I \delta^I}$) is calculated, where I corresponds to the respective Radiance Field (RF^I) [42]. The point with a higher α value is chosen for a contribution towards the rendering of color and density. The resultant RF is considered ground truth for scene composition. It can be observed that the tracing and sampling of the same ray twice are redundant; subsequently, it causes high render times and larger memory footprints. To address these issues, we fuse the RFs into a single representation.

5.1.2 Fusing Radiance Fields

Rendering two or more radiance fields simultaneously for composition is computationally expensive and cannot be scaled as the memory and computation increase linearly with the number of radiance fields involved in composition. To this end, we propose a method that quickly distills from multiple RFs to a single representation. The resulting representation is as compact and efficient as a single RF.

We leverage the already learned 3D information to take losses in 3D, which leads to faster learning (distillation).

Let the two radiance fields to be composed be T_1 and T_2 (teacher1 and teacher2), and the final fused radiance field be S (student). For brevity, let us assume we do not apply any rigid transformation on the radiance fields. We shoot a ray through both T_1 and T_2 and sample points on the ray. The points with low density [•] are pruned out while the ones with high density [•] are utilized. The union of these selected points from T_1 and T_2 acts as our training set for the student S [•] . We query the three radiance fields (T_1, T_2, S) for their density, alpha and color (σ, α, c) at every training sample point location. We apply a *Supervised loss* $||\bullet - \bullet||_2$ to the color and density values of the student (S) against the corresponding teachers (T_1, T_2) at every training point selected above. This supervised distillation will fuse the composition into one single scene.

As a final stage, we render the RGB values for the rays by accumulating the individual color values weighted by the activated volumetric density using Volumetric Rendering Equation [44] and take an alpha-composited *RGB pixel loss* $||\blacksquare - \blacksquare||_2$ for a few iterations. This helps smooth the result around the boundaries of the inserted object.

5.1.3 Fast convergence

To obtain a single representation of a composed radiance field, one could use the traditional *RGB loss* against the rays from composed RFs or rendered views extracted from the composition. But this would essentially be retraining and would amount to the same time as training an RF from scratch. However, since we have 3D information from already-trained RFs, we can leverage the *supervised losses* employed at every sampled point, which achieves faster convergences. The augmented convergence is due to 3D distillation. Pruning of low-density points suggested in Sec. 5.1.2 further speeds up the process.

Additionally, it is often the case that during the composition of radiance fields, one of the scenes is in the majority (dubbed as a background scene). Initializing the student with the background scene significantly speeds up the distillation process. Hence, we initialize the student representation S with the weights of the background scene (one of the dominant teachers T_i). The reduction in time in the case of our distillation-based fusion against total retraining is $3\times$.

5.2 Results

Performance: To validate the performance of our method, we provide Render times and Memory demands against other means of composition, namely 1) Neural-RF, 2) Explicit lattice structures 3) Fused Representation. We tabularize these results in the Tab. 5.3. It can be observed that the render times and memory footprints increase linearly in the case of Neural-RFs [33] and Explicit lattice representations [6,10,42,43] with the increase in the number of scenes used for composition. While it is possible to composite multiple scenes when leveraging Neural-RF, rendering times are a strong limitation, specifi-

Works	Native		Composition	
	Small Memory Footprint	Short Training Time	Small Memory Footprint	Short Inference Time
NeRF [33]	✔	✘	✔	✘
D^2NeRF [51]	✔	✘	✔	✘
PlenOxels [10]	✘	✔	✘	✘
DVGO [42]	✘	✔	✘	✘
InstantNGP [34]	✘	✔✔	✘	✔✔
CNeRF [26]	✘	✔	✘	✘
TensoRF [6]	✔	✔	✔	✘
PVD [9]	✔	✔	✘	✘
FusedRF (Ours)	✔	✔	✔	✔

Table 5.1: While works like NeRF and D^2NeRF struggle in render times, works like DVGO, Plenoxel, and ControlNeRF additionally also demand high memory. Leading to the infeasibility of the composition of more than a few scenes. Our method efficiently fuses the RFs and maintains memory, and renders times to a single RF.

Scene	Ours (w/o RGB)	Ours (Full)
Figure 5.1	36.71	39.89
Figure 5.3 (b)	37.20	40.32
Figure 5.3 (d)	35.18	38.78

Table 5.2: This table shows the PSNR of the images from some scenes in the paper. Please note that the PSNR reported is of the FusedNeRF images against composed NeRF images.

cally when employed in the case of XR applications. On the other hand, Explicit Lattice representations demand a large amount of memory, leading to infeasibility. On the other hand, our FusedRF representation alleviates the issues by fusing the compositions iteratively, constraining memory, and rendering budgets to that of a single RF.

Quantitive Results: Along with maintaining tighter memory and rendering budgets, our proposed FusedRF representation also retains the quality of composited scenes. To validate this, we provide quantitive metrics of our FusedRF representation against the naive composition(Refer Tab. 5.2). This validates the representative capacity of our FusedRF representation.

Qualitative Results: The Qualitative results of our method are presented in the Figs. 5.1 and 5.3.

5.3 Conclusion

We present FusedRF, a method to create a single RF representation for a scene composed of multiple RFs. This reduces the memory and rendering overheads without degradation of quality. We showed

#	Neural Based	Voxel Based		
	NeRF	**DVGO**	**TensoRF**	**Ours**
1	10 MB / 20s	800 MB / 2.01s	25 MB / 2.05s	25 MB / 2.04s
2	20 MB / 40s	1.6 GB / 4.11s	50 MB / 4.25s	25 MB / 2.05s
4	40 MB / 80s	3.2 GB / 8.58s	100 MB / 8.7s	25 MB / 2.04s
8	80 MB / 160s	OOMB	200 MB / 17.2s	25 MB / 2.04s

Table 5.3: Rendering composition of RFs is *slow* and *memory intensive* as the number of scenes increases. Our proposed FusedRF performs fusion once, maintaining the memory and computing constant even when the number of composed scenes increases. Experimented on RTX 3060 Ti (8GB).

(a) Naive Composited (b) Ours FusedRF (c) Naive Composited (d) Ours FusedRF

Figure 5.3: The figure shows results of compositing multiple RF into a single scene, (a) and (c) shows results of composition while (b) and (d) shows results of our FusedRF. Respective memory footprints and rendering times are mentioned in the insets. The scenes are picked form [1, 3, 33].

our method over TensoRF [6] representation here. However, our method can be extended to any RF representation that uses explicit 3D lattices like InstantNGP, DVGO, Plenoxels, etc. [10, 34, 42]. As our method provides tighter memory and rendering budgets, using our FusedRF in XR applications like [8, 29] can facilitate the composition of multiple RFs while maintaining real-time results. The supplementary video provides an overview of our method, the multiview visualization, and the iterative addition of scenes in our results.

Chapter 6

Conclusion and Future Work

6.1 Conclusion

In this thesis, we explored methods to segment, edit and fuse grid-based radiance fields. This enables applications such has generating multi-view segmentation masks, 3D segmentation and manipulation in radiance fields.

We introduced an efficient and easily adaptable stylization method that rapidly produces stylized new renderings of a scene. Our approach demonstrates the ability to integrate style into the representation of a radiance field for casually captured scenes.

We introduced a straightforward and precise technique for object segmentation from TensoRF representation, enabling easy scene editing operations. Our method utilizes an efficient voxel-based representation, enhancing versatility and simplicity compared to previous approaches. Our work offers high-quality segmentation as a foundation for interpreting, understanding, and manipulating 3D scenes. We also developed a user-friendly GUI allowing users to perform segmentation on pre-trained radiance fields through positive and negative strokes.

We introduced a technique for generating a *fused* radiance field representation for scenes with multiple radiance fields. This minimizes memory and rendering burdens without compromising quality. Our approach optimizes memory and rendering resources, making it suitable for XR applications, where real-time composition of multiple radiance fields is essential.

6.2 Future Work

As newer and better representations for radiance fields are popping up, the methods proposed for existing representations need to be adapted. Our method for rapid style transfer can be applied to almost any representation. Our feature distillation followed by region-growing method for segmentation can be applied to newer more promising representations like Instant-NGP [34] and 3D Gaussian Splatting [21]. Although, 3DGS does not require fusion, the latest high-quality grid-based radiance fields [4, 37] will require it for efficient scene composition.

List of Related Publications

Publications, part of the thesis:

[P1] Rahul Goel, Dhawal Sirikonda, Saurabh Saini, and P. J. Narayanan, **"StyleTRF: Stylizing Tensorial Radiance Fields"**, in *Proceedings of the Thirteenth Indian Conference on Computer Vision, Graphics and Image Processing*, 2022.

[P2] Rahul Goel, Dhawal Sirikonda, Saurabh Saini, and P. J. Narayanan, **"Interactive Segmentation of Radiance Fields"**, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023.

[P3] Rahul Goel, Dhawal Sirikonda, Rajvi Shah, and P. J. Narayanan, **"FusedRF: Fusing Multiple Radiance Fields"**, arXiv preprint arXiv:2306.04180, 2023. Accepted and presented at the XR-NeRF Workshop in CVPR 2023.

Publications, not a part of the thesis:

[P4] Vinayak Gupta, Rahul Goel, Dhawal Sirikonda and P. J. Narayanan, **"GSN: Generalizable Segmentaion in Neural Radiance Fields"**, in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2024.

Bibliography

[1] "Sketchfab," https://sketchfab.com/alban/models.

[2] J. T. Barron, B. Mildenhall, M. Tancik, P. Hedman, R. Martin-Brualla, and P. P. Srinivasan, "Mip-nerf: A multiscale representation for anti-aliasing neural radiance fields," in *ICCV*, 2021.

[3] J. T. Barron, B. Mildenhall, D. Verbin, P. P. Srinivasan, and P. Hedman, "Mip-NeRF 360: Unbounded Anti-Aliased Neural Radiance Fields," in *CVPR*, 2022.

[4] ——, "Zip-NeRF: Anti-Aliased Grid-Based Neural Radiance Fields," *ICCV*, 2023.

[5] M. Caron, H. Touvron, I. Misra, H. Jégou, J. Mairal, P. Bojanowski, and A. Joulin, "Emerging Properties in Self-Supervised Vision Transformers," in *ICCV*, 2021.

[6] A. Chen, Z. Xu, A. Geiger, J. Yu, and H. Su, "TensoRF: Tensorial Radiance Fields," in *ECCV*, 2022.

[7] P.-Z. Chiang, M.-S. Tsai, H.-Y. Tseng, W. sheng Lai, and W.-C. Chiu, "Stylizing 3d scene via implicit representation and hypernetwork," in *WACV*, 2022.

[8] N. Deng, Z. He, J. Ye, B. Duinkharjav, P. Chakravarthula, X. Yang, and Q. Sun, "FoV-NeRF: Foveated Neural Radiance Fields for Virtual Reality," *TVCG*, 2022.

[9] S. Fang, W. Xu, H. Wang, Y. Yang, Y. Wang, and S. Zhou, "One is all: Bridging the gap between neural radiance fields architectures with progressive volume distillation," *AAAI*, 2023.

[10] Fridovich-Keil and Yu, M. Tancik, Q. Chen, B. Recht, and A. Kanazawa, "Plenoxels: Radiance Fields without Neural Networks," in *CVPR*, 2022.

[11] L. A. Gatys, A. S. Ecker, and M. Bethge, "Image Style Transfer Using Convolutional Neural Networks," in *CVPR*, 2016.

[12] R. Goel, D. Sirikonda, S. Saini, and P. J. Narayanan, "StyleTRF: Stylizing Tensorial Radiance Fields," in *Proceedings of the Thirteenth Indian Conference on Computer Vision, Graphics and Image Processing*, 2022.

[13] R. Goel, D. Sirikonda, S. Saini, and P. Narayanan, "Interactive Segmentation of Radiance Fields," in *CVPR*, 2023.

[14] R. Goel, D. Sirikonda, R. Shah, and P. Narayanan, "FusedRF: Fusing Multiple Radiance Fields," *arXiv preprint arXiv:2306.04180*, 2023.

[15] S. J. Gortler, R. Grzeszczuk, R. Szeliski, and M. F. Cohen, "The Lumigraph," in *Proceedings of SIGGRAPH*, 1996.

[16] D. Ha, A. M. Dai, and Q. V. Le, "HyperNetworks," in *ICLR*, 2017.

[17] K. He, J. Sun, and X. Tang, "Guided Image Filtering," *IEEE Trans. Pattern Anal. Mach. Intell. (T-PAMI)*, 2013.

[18] H.-P. Huang, H.-Y. Tseng, S. Saini, M. Singh, and M.-H. Yang, "Learning to Stylize Novel Views," in *ICCV*, 2021.

[19] Y.-H. Huang, Y. He, Y.-J. Yuan, Y.-K. Lai, and L. Gao, "StylizedNeRF: Consistent 3D Scene Stylization as Stylized NeRF via 2D-3D Mutual Learning," in *CVPR*, 2022.

[20] J. Johnson, A. Alahi, and L. Fei-Fei, "Perceptual Losses for Real-Time Style Transfer and Super-Resolution," in *ECCV*, 2016.

[21] B. Kerbl, G. Kopanas, T. Leimkühler, and G. Drettakis, "3d gaussian splatting for real-time radiance field rendering," *ACM Trans. Graph.*, 2023.

[22] J. Kerr, C. M. Kim, K. Goldberg, A. Kanazawa, and M. Tancik, "Lerf: Language embedded radiance fields," in *International Conference on Computer Vision (ICCV)*, 2023.

[23] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," in *ICLR*, 2015.

[24] S. Kobayashi, E. Matsumoto, and V. Sitzmann, "Decomposing NeRF for Editing via Feature Field Distillation," in *NeurIPS*, 2022.

[25] K. Kutulakos and S. Seitz, "A theory of shape by space carving," in *ICCV*, 1999.

[26] V. Lazova, V. Guzov, K. Olszewski, S. Tulyakov, and G. Pons-Moll, "Control-NeRF: Editable Feature Volumes for Scene Rendering and Manipulation," in *WACV*, 2023.

[27] M. Levoy and P. Hanrahan, "Light field rendering," in *Proceedings of SIGGRAPH*, 1996.

[28] B. Li, K. Q. Weinberger, S. Belongie, V. Koltun, and R. Ranftl, "Language-driven Semantic Segmentation," in *ICLR*, 2022.

[29] C. Li, S. Li, Y. Zhao, W. Zhu, and Y. Lin, "RT-NeRF: Real-Time On-Device Neural Radiance Fields Towards Immersive AR/VR Rendering," in *Proceedings of the 41st IEEE/ACM International Conference on Computer-Aided Design*, 2022.

[30] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft COCO: Common Objects in Context," in *ECCV*, 2014.

[31] R. Martin-Brualla, N. Radwan, M. S. M. Sajjadi, J. T. Barron, A. Dosovitskiy, and D. Duckworth, "NeRF in the Wild: Neural Radiance Fields for Unconstrained Photo Collections," in *CVPR*, 2021.

[32] B. Mildenhall, P. P. Srinivasan, R. Ortiz-Cayon, N. K. Kalantari, R. Ramamoorthi, R. Ng, and A. Kar, "Local Light Field Fusion: Practical View Synthesis with Prescriptive Sampling Guidelines," *ACM Trans. Graph.*, 2019.

[33] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng, "NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis," in *ECCV*, 2020.

[34] T. Müller, A. Evans, C. Schied, and A. Keller, "Instant Neural Graphics Primitives with a Multiresolution Hash Encoding," *ACM Trans. Graph.*, 2022.

[35] T. Nguyen-Phuoc, F. Liu, and L. Xiao, "SNeRF: Stylized Neural Implicit Representations for 3D Scenes," *ACM Trans. Graph.*, 2022.

[36] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "PyTorch: An Imperative Style, High-Performance Deep Learning Library," in *NeurIPS*, 2019.

[37] C. Reiser, R. Szeliski, D. Verbin, P. P. Srinivasan, B. Mildenhall, A. Geiger, J. T. Barron, and P. Hedman, "MERF: Memory-Efficient Radiance Fields for Real-time View Synthesis in Unbounded Scenes," *Proceedings of SIGGRAPH*, 2023.

[38] Z. Ren, A. Agarwala, B. Russell, A. G. Schwing, and O. Wang, "Neural Volumetric Object Selection," in *CVPR*, 2022.

[39] C. Rother, V. Kolmogorov, and A. Blake, "GrabCut: interactive foreground extraction using iterated graph cuts," *Proceedings of SIGGRAPH*, 2004.

[40] J. L. Schönberger and J.-M. Frahm, "Structure-from-Motion Revisited," in *CVPR*, 2016.

[41] N. Snavely, S. M. Seitz, and R. Szeliski, "Photo tourism: exploring photo collections in 3d," *ACM Trans. Graph.*, 2006.

[42] C. Sun, M. Sun, and H. Chen, "Direct Voxel Grid Optimization: Super-fast Convergence for Radiance Fields Reconstruction," in *CVPR*, 2022.

[43] C. Sun, M. Sun, and H.-T. Chen, "Improved Direct Voxel Grid Optimization for Radiance Fields Reconstruction," *arXiv preprint arXiv:2206.05085*, 2022.

[44] A. Tagliasacchi and B. Mildenhall, "Volume Rendering Digest (for NeRF)," *arXiv preprint arXiv:2209.02417*, 2022.

[45] M. Tancik, P. P. Srinivasan, B. Mildenhall, S. Fridovich-Keil, N. Raghavan, U. Singhal, R. Ramamoorthi, J. T. Barron, and R. Ng, "Fourier features let networks learn high frequency functions in low dimensional domains," *NeurIPS*, 2020.

[46] Z. Teed and J. Deng, "Raft: Recurrent all-pairs field transforms for optical flow (extended abstract)," in *IJCAI*, 2021.

[47] C. Tomasi and R. Manduchi, "Bilateral filtering for gray and color images," in *ICCV*, 1998.

[48] V. Tschernezki, I. Laina, D. Larlus, and A. Vedaldi, "Neural Feature Fusion Fields: 3D distillation of self-supervised 2D image representations," in *International Conference on 3D Vision (3DV)*, 2022.

[49] R. Tucker and N. Snavely, "Single-View View Synthesis With Multiplane Images," in *CVPR*, 2020.

[50] W. Wang, S. Yang, J. Xu, and J. Liu, "Consistent Video Style Transfer via Relaxation and Regularization," 2020.

[51] T. Wu, F. Zhong, A. Tagliasacchi, F. Cole, and C. Oztireli, "D^2NeRF: Self-Supervised Decoupling of Dynamic and Static Objects from a Monocular Video," in *NeurIPS*, 2022.

[52] Y. Xie, T. Takikawa, S. Saito, O. Litany, S. Yan, N. Khan, F. Tombari, J. Tompkin, V. Sitzmann, and S. Sridhar, "Neural Fields in Visual Computing and Beyond," *Comput. Graph. Forum*, 2022.

[53] K. Zhang, N. Kolkin, S. Bi, F. Luan, Z. Xu, E. Shechtman, and N. Snavely, "ARF: Artistic Radiance Fields," in *ECCV*, 2022.

[54] K. Zhang, G. Riegler, N. Snavely, and V. Koltun, "NeRF++: Analyzing and Improving Neural Radiance Fields," *arXiv:2010.07492*, 2020.

[55] S. Zhi, T. Laidlow, S. Leutenegger, and A. Davison, "In-Place Scene Labelling and Understanding with Implicit Scene Representation," in *ICCV*, 2021.

[56] K. Zhou, Y. Hu, S. Lin, B. Guo, and H.-Y. Shum, "Precomputed Shadow Fields for Dynamic Scenes," *ACM Trans. Graph.*, 2005.