Circuits and Systems
Mekelweg 4,
2628 CD Delft
The Netherlands
https://cas.tudelft.nl/

CAS-2022-00

M.Sc. Thesis

A trainable activation scheme for applying Neural Radiance Fields in novel view synthesis tasks

Chuhan Wang B.Sc.

Faculty of Electrical Engineering, Mathematics and Computer Science **Delft University of Technology**

A trainable activation scheme for applying Neural Radiance Fields in novel view synthesis tasks

Thesis

submitted in partial fulfillment of the
requirements for the degree of

Master of Science

in

Electrical Engineering

by

Chuhan Wang B.Sc.
born in Wuhan, China

This work was performed in:

Circuits and Systems Group
Department of Microelectronics
Faculty of Electrical Engineering, Mathematics and Computer Science
Delft University of Technology

Delft University of Technology

Delft University of Technology
Department of
Microelectronics

The undersigned hereby certify that they have read and recommend to the Faculty of Electrical Engineering, Mathematics and Computer Science for acceptance a thesis entitled **"A trainable activation scheme for applying Neural Radiance Fields in novel view synthesis tasks"** by **Chuhan Wang B.Sc.** in partial fulfillment of the requirements for the degree of **Master of Science**.

Dated: 17 October 2022

Chairman: _____
prof.dr.ir. Justin Dauwels

Advisor: _____
dr. Letao Liu

Committee Members: _____
prof.dr.ir. Justin Dauwels

_____
dr. Francesco Fioranelli

Abstract

3D scene reconstruction is a common computer vision task with many applications. The synthesized virtual environments are beneficial for many downstream applications such as 3D modeling, building inspection, virtual reality, etc. As conventional scene reconstruction methods often require expensive data collections and prior information on the geometry, a learning-based method named Neural Radiance Fields (NeRF) has gained a surge of interest within the computer vision community recently for its capacity to achieve state-of-the-art view synthesis performance and achieve photorealistic rendering from only a sequence of RGB images as the input. However, NeRF has a strict requirement for input images with accurate camera poses, which is often not available in real-life applications. To this end, we provide an end-to-end guideline for 3D scene reconstruction using NeRF under a real-world scenario the objective is to reconstruct the HDB facade in Singapore. This guideline requires only RGB images as the input and can achieve photorealistic rendering results which are competitive with the results from the conventional point cloud-based method. Besides, we build our own models upon NeRF and also improve its performance in representing fine details. We first research that the ability to represent high-frequency contents in the signal is limited by its ReLU activations in the Multi-layer Perceptron (MLP) network, and demonstrate that mapping the input to the MLPs from low dimensional space to high dimensional space significantly improves the reconstruction and view synthesis quality. Afterward, We make several attempts to perform input mapping. We first use Gaussian-distributed Fourier features to replace the original positional encoding used in NeRF. Then, we research the activations in coordinate MLP and propose an embedding-less NeRF model equipped with parameterized sine activations called SIRENeRF. Next, we extend the use of parameterized activations from sine activations to a class of non-periodic activations and propose a trainable activation scheme that not only achieves higher scene reconstruction results but also enjoys better flexibility to different datasets. Experiments show that all of the above attempts outperform the positional encoding in terms of scene reconstruction and view synthesis results.

Acknowledgments

The last two years have been a great journey for me. The Master's study at TU Delft is, though not always full of joy because of the stress and the pandemic, a wonderful experience. The campus life, the people I met, and all the days and nights in this beautiful city, will be a precious memory forever.

I would like to thank my supervisor, Dr. Justin Dauwels, for supporting and advising me over the last nine months. Your suggestion and guidance are always helpful for me to finish my work. I would also like to thank my advisor, Dr. Letao Liu, for your advice. Our discussions are so inspiring and your experience makes it easier for me to do the research. I also want to say special thanks to my family and friends. The support and love from you mean a lot to me and make me feel no longer lonely.

Chuhan Wang B.Sc.
Delft, The Netherlands
17 October 2022

Contents

List of Figures

List of Tables

1 Introduction

1.1 Motivation & Research Statement

3D reconstruction is a process of capturing shape and appearance of real objects and building a 3D profile for the targets. With a determined 3D profile, many downstream applications such as 3D modelling and novel view synthesis are unlocked. Therefore, 3D reconstruction has been a core problem of a wide variety of fields including computer vision, computer graphics, and virtual reality. However, applying photorealistic 3D scene reconstruction and rendering in the real world is highly challenging. This is because real-world scenes often contain structures and topology of high complexity, and some of the objects might be semi-transparent or shaded under certain circumstances. Moreover, these structures/topology are evolving over time due to many factors including the motions from the objects, the varying light conditions, etc. What is more, in a real-world scene the environment is possibly unbounded, which means that the objects in the background are of various distance and can be extremely far away. All these facts increase the difficulty of reconstructing a real-world scene.

Conventional 3D reconstruction methods usually represent the shape and appearance of the target objects in the form of point cloud models, mesh models or geometric models. As the most basic model, point cloud requires a collection of points in a 3D space which usually include x, y, z coordinates, vector information, colour, and other relevant information related to the points. These points can be joined together to form objects or surfaces, which can then be used to build a complete model of the targets. To collect these points, a laser scanning system is often used where laser beams are transmitted and bounces off a specific point and the time is recorded. With the time, the distance between the target and the transmitter is measured and a point cloud model is then built. On the basis of point clouds, a mesh model can be further constructed to illustrate clearer appearance. However, constructing a point cloud model for a real-world scene is always expensive in terms of both the labour of data collection and hardware requirements. Besides, to generate an accurate point cloud model, a lot of prior information about the targets is required, which is also difficult to acquire. To this end, recent researches leverage deep networks to save the work to acquire expensive data and generate point clouds. Many deep network-based methods work towards scene reconstruction and novel view synthesis with only RGB images as the input data. Since RGB images are much cheaper to collect in the real world applications, these deep network-based methods are supposed to show more flexibility and a significant advantage in terms of the cost.

Our research is in collaboration with TUV SUD, a company that is applying the 3D reconstruction technique in a real-world scenario and synthesizing the HDB facade in Singapore. There are over 4,000 buildings in Singapore, thus follow-up applications

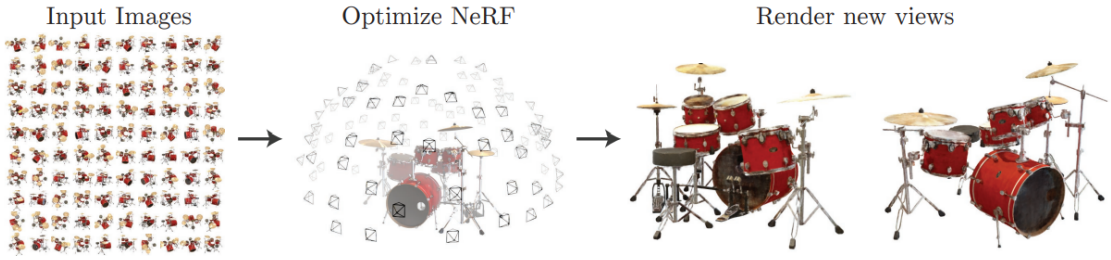

Figure 1.1: Novel views synthesized by Neural Radiance Fields [32] with posed RGB images as the input.

Figure 1.2: The target building in our research.

such as building inspection and repair can be seamlessly managed through the 3D synthesized model. However, regularly monitoring a large number of buildings using conventional 3D reconstruction methods requires huge time and work. Therefore, the motivation of our research is to find a solution that we can feed only the RGB images as the input to the algorithm to generate the 3D virtual environment in a much cheaper way. In our research, the objective scene is a building shown in figure 1.2.

At present, TUV SUD uses software named ContextCapture to do the view synthesis

Figure 1.3: The view synthesis results from ContextCapture.

tasks in real life. Figure 1.3 presents the reconstruction results of the target building. ContextCapture provides a sophisticated point cloud-based method to automatically generate high-resolution 3D models without any human intervention. This method has a high requirement on the input dataset, which is supposed to be captured at a constant focal length during the acquisition and under suitable light conditions. It also requires a large number of input views to ensure the accuracy of the output 3D models. However, ContextCapture is commercial software that is not open-sourced and not free. Therefore, a secondary goal of this research is to build a free method that can achieve competitive performance compared to ContextCapture.

1.2 Research Objectives and Expected Outcomes

The primary objective of this research is to provide an effective and cheap guideline to synthesize the target building with only RGB images as the input data. This guideline is expected to serve as an alternative way that requires fewer data and resources than the commercial software they are using to do the same task. We perform the following tasks in this research:

1. Study a Neural Radiance fields (NeRF) method [32] which can achieve state-of-the-art view synthesis results with cheap resources required.

2. Conceptualize the coordinate MLP in NeRF and implement enhancements on NeRF to improve the performance.
3. Provide detailed analysis of the data cleaning and pre-processing strategies.
4. Perform a quantitative and qualitative evaluation of the generated 3D scenes.
5. Provide a detailed comparison between the existing NeRF model and our proposed methods.

1.3 Research Contributions

In this research, we study the recent works of view synthesis and focus on implementing and improving NeRF [32] to synthesize novel views of our target scene. We specifically research the use of coordinate MLP in view synthesis tasks. Combine Fourier Features [52] with NeRF, we propose an alternative way of mapping the inputs of the coordinate MLP from lower dimensional space to higher dimensional space and show that this method can help represent the reconstruction of the scene in better detail and reduce the noise in the background. What is more, this method outperforms the original positional encoding method used in NeRF [32] in terms of the qualitative results. We also study the activations in coordinate MLP. Inspired by recent works on activations [48][39], we propose our frameworks that leverage parameterized activations to achieve better results on our target scene than NeRF, and leave out the need for positional encoding. We propose SIRENeRF, which uses sine activation-based SIREN layers to replace the MLP in NeRF and achieves better qualitative results. We also theoretically discuss the importance of the parameters in activations and provide a principle of tuning these parameters. Further, we extend our research to a more general class of parameterized activations and propose a trainable activation-based NeRF (TARF) which not only outperforms NeRF in terms of the qualitative results and quantitative results on our target scene but also shows significantly better flexibility and efficiency than SIRENeRF by leaving out the need of manually tuning the parameters in activations. In addition, we also provide a guideline to clean the data and pre-process the data, and we discuss and show two different manners of reconstruction: reconstruct the target with forward-facing renderings and reconstruct the target with one 360-degree spherical rendering. In the end, we also outline the drawback and limitations of our methods in training speed and resolution and suggest possible future work that can be pursued for improvement.

In summary, we present the following contributions:

- We implement a neural radiance fields-based method to achieve novel view synthesis on our own target scene with only RGB images as the input.
- We leverage Fourier Features as an alternative positional encoding method in NeRF and outperform the original positional encoding.
- We leverage parameterized activations in NeRF, and propose a SIRENeRF framework that uses sine functions instead of conventional activations. We show that SIRENeRF can achieve better results on our target scene, and we provide principles for tuning the parameters in activations.

- We further propose a class of general trainable activations and achieve better results on our target scene with our proposed TARF framework. We also show that TARF outperforms SIRENeRF in terms of flexibility and efficiency on different datasets.
- We provide a guideline to clean and pre-process the data, and discuss different data preparation and rendering strategies

We demonstrate that our research successfully achieves the research objectives and paves the way for implementing NeRF in other real-world applications.

1.4 Report Structure

The following research report contains 5 chapters. In chapter 2, We begin with an analysis of recent works related to our view synthesis task, then introduce NeRF as our base method. In chapter 3, a more detailed theoretical analysis of NeRF is given. In chapter 4, we demonstrate our study on coordinate MLP and demonstrate our proposed methods related to positional encoding and parameterized activations. In chapter 5, We demonstrate our data preparation strategy and related discussion, then we present the experiment results. A discussion about the results and limitations of our work is also given.In chapter 6, a conclusion of our research is given.

2 Related Research

2.1 Structure from Motion

Structure from Motion (SfM) is a technique of reconstructing 3D structures from 2D images of an object taken from different viewpoints. As a fundamental goal of a large number of researches in the field of computer vision, SfM is used in many applications such as 3D scanning, visual simultaneous localization, and 3D reconstruction. Generally, SfM includes three main stages. The first stage is to extract the features such as points or lines from the images and match the features between a pair of images. One of the most popular methods is the scale-invariant feature transform (SIFT) [27] [35] [1], which is widely used as the backbone of many corresponding point-based SfM methods. As a multidimensional descriptor, SIFT transforms an image into a collection of local features that are invariant to the scaling and rotation of the underlying image. Other methods such as PCA-SIFT [22] [60], gradient location and orientation histogram (GLOH) [42] [28], and ORB [1] [34] are also universal feature descriptors. The second stage is then to estimate the camera motion, where the corresponding extracted feature points are used to measure the relative motion between a pair of images. The estimated motion is then decomposed into the pairwise rotation and translation measurements. The final step is to recover the 3D structure by minimizing the reprojection error. Most of the recent methods utilize bundle adjustment techniques which optimize a cost function known as the total reprojection error. This cost function enables SfM methods to simultaneously determine the structure and the calibration parameters of each viewpoint given several images of a stationary scene.

2.1.1 Incremental SfM and COLMAP

Among various SfM strategies, incremental SfM has been shown up as the most frequently used strategy for the reconstruction of unordered photo collections. Schonberger et al [43] propose an incremental SfM pipeline that has been proved effective on various datasets and can serve as a solution for general purposes. Meanwhile, an open-source implementation of this pipeline named COLMAP is also released. This pipeline involves two stages. The first stage is the correspondence search, where the scene overlap in the input sequence is found and projections of the same points in overlapping images are identified. The process starts by doing feature extraction, where the local feature set at each location is detected for each input image. Then, SfM matches images that observe the same view of the scene by using the features as an appearance description of the images. The following step is to verify the matching by estimating a transformation that maps feature points between images using projective geometry. The correspondence search stage outputs a set of geometrically verified image pairs and a graph of image projections for each point. The scene graph then serves as the input

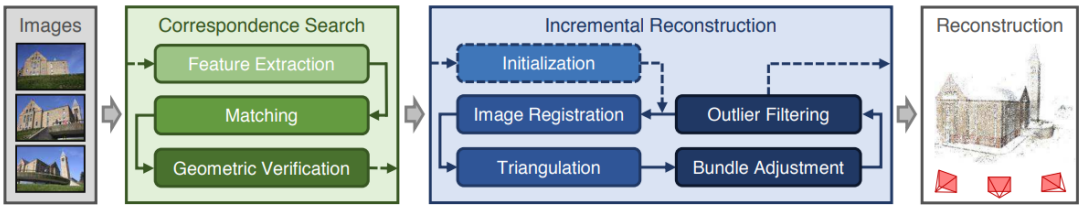

Figure 2.1: The pipeline of Incremental Structure-from-Motion (COLMAP) [43].

for the second stage, incremental reconstruction.

Moving to the reconstruction stage, SfM first initializes the model by carefully selecting an initial pair for the two-view reconstruction. A well-chosen initial pair can bring more robustness and accuracy to the reconstruction because initializing from a dense location with more overlapping cameras increases redundancy. Then new images are registered to the current model. The image registration procedure is achieved by solving a Perspective-n-Point problem [14] using feature correspondences to triangulated points in registered images (2D-3D correspondences). Image triangulation is another significant step that serves to increase the stability of the existing model by adding more redundancy and boosting the image registration procedure by providing additional 2D-3D correspondences. Bundle adjustment then conducts a joint refinement of camera parameters and point parameters to minimize the reprojection error. The whole process is summarized in figure 2.1.

2.2 Novel view synthesis

Novel view synthesis methods can be divided into three main groups including image-based methods, learning-based methods, and geometry-based methods. Image-based methods attempt to visualize 3D scenes and objects in a realistic way without full 3D model reconstruction [8] [55] [47]. image-based methods treat the view synthesis and rendering problem as a multidimensional sampling problem, where new views are generated from a collection of densely sampled images or videos instead of an accurately pre-built 3D model of the scenes. Well-known image-based methods include panoramas [51] [44] [4], light fields [25] [23] [13], concentric mosaics [45] [46], etc.

Geometry-based methods refer to techniques that utilize explicit scene geometry to synthesize new views from unstructured input views. Different from image-based methods, geometry-based methods need to reconstruct a 3D model first before rendering novel views. Geometry-based methods can be categorized based on either using global or local geometry. Methods that leverage the global geometry of the scenes attempt to compute a single global mesh. Methods that globally texture mapping the global mesh generally fit in constrained scenes such as a panoramic view with limited rotation and translation [16], but they mostly require accurate geometry from dense range scans and hundreds of captured images to sample the outgoing radiance at points on an object's surface. In addition, some rendering strategies are designed to compute per-pixel blending weights for reprojected images [5]. However, these strategies expect

an estimation of high-quality meshes whose geometric boundaries align well with edges in images, which is difficult for global geometry-based methods. Some methods [17] [53] [18] try to overcome this limitation by combining both global mesh and local depth estimation, but precisely defining view sampling requirements for robust mesh estimation remains difficult for these methods, and make them not effective for casual content capture scenarios. Many methods, instead, turn to leverage the local geometry to leave out the global mesh estimation which is often too expensive. local geometry-based methods compute local geometry in detail for each input image and render novel views by reprojecting and blending nearby input images [9][6]. Geometry-based methods can produce high-quality rendering results, though they are often limited in the applications by their requirements on the amount of data and the computation capacity.

Learning-based methods train deep learning networks to provide an end-to-end solution for novel view synthesis and have been a trend recently. Deep learning-based methods often transform view synthesis into a supervised learning problem. Some methods that make a combination of light fields and deep networks are proposed to achieve tasks including synthesizing a 4D RGBD light field from a single image[49], synthesizing novel views for light fields cameras[21] [56], or synthesizing novel views directly from pixels[15]. These methods utilize deep networks to predict depth separately for each view. However, these methods often face a shortcoming that rendering from varying viewpoints can lead to inconsistency because of the principle that predicting local geometry separately for each view. In order to reduce the inconsistency, Zhou et al. [59] introduce a pipeline that enforces consistency by using the same predicted scene representation to render all novel views. Similar to [59], Mildenhall et al. propose a local light field fusion [31] method which also makes use of multiplane images. Local light field fusion adopts multiplane images as the light field representation and predicts multiple multiplane images for each input view. The predicted multiplane images are optimized through a convolutional neural network (CNN) architecture which dynamically adjusts the number of depth planes based on the input view sampling rate. Recently, Neural Radiance Fields (NeRF) [32] has been trending as a novel view synthesis method that can achieve state-of-the-art rendering results. NeRF represents a scene as neural radiance fields and can synthesize novel views of complicated scenes by optimizing an underlying continuous volumetric scene function.

2.3 Neural Radiance Fields

Mildenhall et al. [32] propose Neural Radiance Fields(NeRF) in 2020, which is a method to synthesize novel views of complex scenes from a collection of posed images. Over the last couple of years, NeRF has drawn dramatic attention within the Computer Vision and Machine Learning community because of its state-of-the-art performance in synthesizing novel views and surprisingly simplicity. NeRF models the target scene as continuous radiance fields(volume density and view-dependent color at any continuous location), and optimizes the representation by optimizing a continuous volumetric scene function with the help of a fully-connected network (Multilayer Perceptron), following up a differentiable volume rendering technique to accumulate samples of this scene representation along rays to render the scene from any viewpoint.

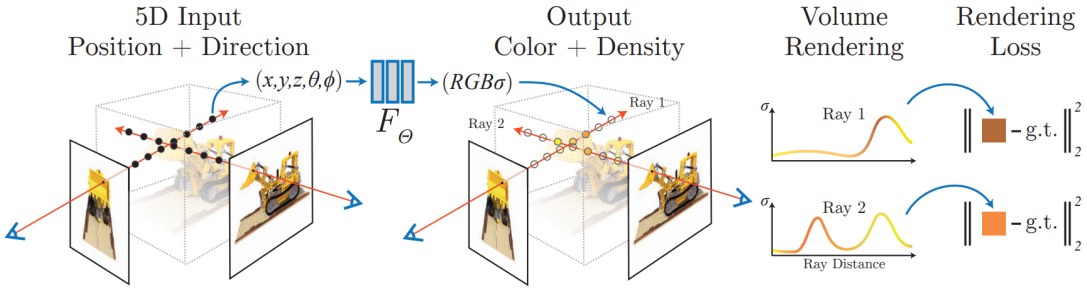

Figure 2.2: Neural Radiance Fields [32] encodes a 3D scene as a continuous function which takes 5D coordinates including the 3D position $\boldsymbol{x} = (x, y, z)$ and the 2D viewing direction $\boldsymbol{d} = (\theta, \phi)$ as the inputs, and predicts the color $\boldsymbol{c}(\boldsymbol{x}, \boldsymbol{d}) = (r, g, b)$ and volume density $\sigma(\boldsymbol{x})$ via MLPs. Later using volume rendering techniques to composite the volume density and the color into an image. The mean squared error between the predicted image and the ground truth are back-propagated to optimize the representation.

Figure 2.2 shows the pipeline of NeRF. NeRF takes 5D coordinates (extracted from a set of posed images) as the input, feeds each input (location (x, y, z) and viewing direction (θ, σ) for one single spatial location) into the MLP, and obtains the volume density and view-dependent emitted radiance at that location as the output. Differentiable volume rendering is then adopted to n composites the volume density and view-dependent emitted radiance into an actual image. The rendering loss is computed on a per-pixel basis.

There are two additional strategies used in NeRF to improve performance. Positional encoding is adopted to map the input from lower dimensional space to higher dimensional space, which overcomes a limitation that coordinate MLPs are not capable of handling high-frequency content in the signal. Hierarchical volume rendering is adopted to improve the efficiency of rendering by allocating samples along the ray proportionally to their contribution to the scene. More details about the NeRF are discussed in section 3.

2.4 Recent trends in improving Neural Radiance Fields

The impressive results achieved by NeRF have started a trend within the research community, and it has been extended to many fronts. Some extensions target a better performance of implementing NeRF in real-world scenarios, where NeRF might be limited by many factors. One of the limitations is that NeRF assumes a geometrically and photometrically static object and a constant density and radiance of the world, which means two images taken at the same location and from the same direction should be identical. This assumption can hardly be satisfied in real-world scenarios. NeRF in the wild [29] manages to overcome this limitation and achieve state-of-the-art results on unconstrained internet photo collections where the images are taken from different time of the day, weather, and environmental conditions. The key ideas of NeRF in the wild are two-fold: Firstly, it models per-image appearance variations in a learned low-dimensional latent space. This appearance embedding process means learning a

Figure 2.3: (a) internet photo collections (b) view synthesis results from NeRF in the wild [29].

shared appearance representation across the entire photo collection. The learned low-dimensional latent space enables extra controls of the appearance of output renderings. Secondly, it models the scene as the union of shared and image-dependent elements, which allows unsupervised decomposition of scene content into "static" and "transient" components. These enhancements allow NeRF in the wild to demonstrate novel view renderings from some internet photo collections of famous landmarks, as shown in figure 2.3.

Some extensions aim to use NeRF with fewer input views. NeRF has been proved to work on a large number of images (the LLFF NeRF Real dataset [31] has around 50 images for each narrow forward-facing scene). With fewer images in the dataset, NeRF is often challenged by a higher risk of fitting incorrect geometries. However, in real-life tasks, data is expensive and a sufficient number of input views of the target scene is not always guaranteed. Depth-supervised NeRF [12] addresses this issue by leveraging depth as additional supervision to guide the geometry learned by NeRF. It shows that an RGB image combined with depth can make a stronger signal than only RGN images in the 3D reconstruction scenario, and the depth supervision can dramatically reduce the number of training data needed and help produce the results with the same quality. An additional merit of Depth-supervised NeRF is that depth is a cheap resource and can be obtained with no extra effort. This is because NeRF requires camera poses as the input, and the poses are often estimated from COLMAP. Apart from the camera poses, COLMAP also outputs sparse 3D point clouds along with their reprojection errors, which provide depth information. Similarly, MVSNeRF [10] also focuses on modeling 3D geometry correctly with fewer input views. To achieve this goal, It leverages plane-swept cost volumes for geometry-aware scene reasoning

and combines this with physically based volume rendering for neural radiance field reconstruction.

Another limitation of NeRF is that it has a strict requirement on the accuracy of the poses, while in real-life tasks, accurate camera poses are not always accessible. Some extensions attempt to overcome the downgrade of performance out of inaccurate poses by jointly optimizing the camera poses and learning the NeRF representation. Self-Calibrating NeRF [20] adopts a self-calibration algorithm for generic cameras with arbitrary non-linear distortions. It also leverages a geometric loss function consisting of projected ray distance loss to incorporate geometric consistency. These enhancements enable it to learn the geometry of the scene and the accurate camera intrinsic and extrinsic (pose) without COLMAP. NeRF– [57] introduces a two-stage pipeline that treats camera poses and intrinsics as learnable parameters. This pipeline allows NeRF– to be an end-to-end approach without pre-processing the RGB images using COLMAP. Bundle adjusting NeRF [26] borrows the ideas from image alignment approaches and showcases that the coarse-to-fine registration strategy can improve the positional encoding in NeRF and help recover scene representations from imperfect camera poses.

Backgrounds: Neural Radiance Fields

3

NeRF models the target scene as radiance fields and optimizes the representation including the volume density and the view-dependent color using MLP networks, then uses a differentiable volume rendering technique to project the volume density and the color into an image. The details of the MLP networks are discussed in section 3.1, and the details of the volume rendering used in NeRF are discussed in section 3.3. The hierarchical volume sampling strategy used to improve the rendering efficiency is discussed in section 3.3. Positional encoding serves to map the input of the MLPs from lower dimensional space to higher dimensional space. This technique significantly improves the performance of MLPs because MLPs cannot learn the information contained in higher frequency contents of the input images well. The positional encoding used in NeRF and some other strategies which can achieve a similar or even better performance are discussed in section 4.

3.1 Optimizing NeRF via MLP networks

NeRF [32] encodes a 3D scene as a continuous 3D representation by using two concatenated MLP networks. The input is a 5D coordinates, which contains the 3D location $\boldsymbol{x} = (x, y, z)$ and the 2D viewing direction $(\boldsymbol{d} = (\theta, \phi)$. The output is an emitted color $\boldsymbol{c} = (r, g, b)$ and a volume density σ. The MLP networks can be summarized as $F_\Theta : (x, y, z, \theta, \phi) \rightarrow (r, g, b, \sigma)$, where Θ are the weights of the networks that are optimized to map from each input 5D coordinate to its corresponding volume density and emitted color.

Figure 3.1 shows the architecture of the concatenated MLP networks. The first MLP takes the location $\boldsymbol{x} = (x, y, z)$ as the input, where the corresponding volume density σ for each input location is learned through a fully-connected network. In the experiments, the MLP is set to have 8 layers with 256 channels per layer by default.

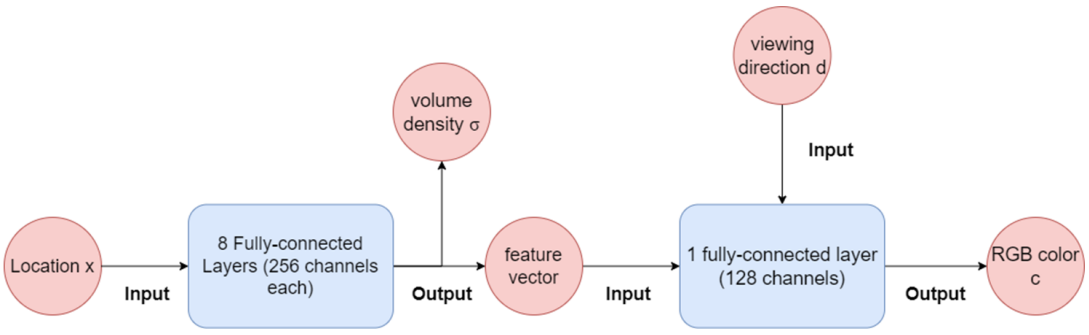

Figure 3.1: Overview of the MLPs used in NeRF.

ReLU activations are used to bring about the non-linearity. Apart from the volume density, the first MLP also outputs a 256-dimensional feature vector. The feature vector is then combined with the viewing direction $\boldsymbol{d}$ as the inputs for the second MLP, where the RGB color $\boldsymbol{c}$ is learned through a 1-layer fully-connected network with 128 channels. To conclude, the volume density σ is purely predicted by the location $\boldsymbol{x}$, while the RGB color $\boldsymbol{c}$ is predicted by both the location (the feature vector) and viewing direction. This allows a multi-view consistency of the representation [32].

3.2 Volume Rendering with Radiance Fields

NeRF adopt principles from classical volume rendering. The volume density $\sigma(\boldsymbol{x})$ is defined as the differential probability of a ray terminating at an infinitesimal particle at location $\boldsymbol{x}$. Assume a camera ray is represented as $\boldsymbol{r}(x) = \boldsymbol{o} + t\boldsymbol{d}$, the expected color $C(\boldsymbol{r})$ of this ray can be represented as:

$$C(\boldsymbol{r}) = \int_{t_n}^{t_f} T(t)\sigma(\boldsymbol{r}(t))\boldsymbol{c}(\boldsymbol{r}(t), \boldsymbol{d})dt, \tag{3.1}$$

where

$$T(t) = exp(-\int_{t_n}^{t_f} \sigma(\boldsymbol{r}(s))ds) \tag{3.2}$$

In equation 3.1, t_n and t_f represent the near and far bound. The function $T(t)$ denotes the accumulated transmittance along the ray from t_n to t_f, which can be interpreted as the probability that the ray travels from t_n to t_f without hitting any other particle. Rendering a view from an imaginary camera requires the estimate of this integral $C(\boldsymbol{r})$. In NeRF, this integral is estimated using quadrature. Different from deterministic quadrature which is often used for rendering voxel grids, NeRF adopts a stratified sampling on the quadrature. The stratified sampling segments $[t_n, t_f]$ into N equally spaced bins and draws one sample uniformly at random from within each bin:

$$t_i \sim \mathcal{U}[t_n + \frac{i-1}{N}(t_f - t_n), t_n + \frac{i}{N}(t_f - t_n)] \tag{3.3}$$

The the samples are used to estimate $C(\boldsymbol{r})$ with the quadrature rule:

$$\hat{C}(\boldsymbol{r}) = \sum_{i=1}^{N} T_i(1 - exp(-\sigma_i\delta_i))(c)_i, \ where\ T_i = exp(-\sum_{j=1}^{i-1} \sigma_j\delta_j) \tag{3.4}$$

In 3.4, $\delta_i = t_{i+1} - t_i$ represents the distance between two adjacent samples. The function 3.4 is trivially differentiable and reduces to traditional alpha compositing with alpha values $\alpha_i = 1 - exp(-\sigma_i\delta_i)$.

Once the predicted image is composited, it is used to be compared with the corresponding ground truth. NeRF is trained to minimize the mean-squared error (MSE) between the predicted renderins and the ground truth:

$$\ell_{MSE} = \sum_{p\in\varphi} \|\widehat{C}(r_p) - C(r_p)\|_2^2 \tag{3.5}$$

In 5.2, φ denotes all the pixels in the training set images, and $r_p(t) = o + td_p$ denotes the ray shooting from camera center to the corners of a given pixel p. $\widehat{C}(r_p)$ and $C(r_p)$ denotes the predicted RGB value of the pixel and the corresponding ground truth. The MSE loss is back-propagated to the MLPs mentioned in section 3.1 to optimize the representation.

3.3 Hierarchical volume sampling

A trick called hierarchical volume sampling is adopted in the volume rendering strategy of NeRF mentioned in section 3.2. Inspired by another work in volume rendering [24], NeRF leverage a hierarchical representation which proves to be able to increase the efficiency of rendering. The motivation behind is that volume rendering is meant to densely evaluate the neural radiance field network at all N query points along the ray between the near and far bound. However, this strategy cannot achieve the highest efficiency because free space and occluded regions are also repeatedly sampled, though they do not contribute to the rendered image.

To increase efficiency, instead of optimizing only one network, two networks are simultaneously optimized to represent the target scene. These two networks consist of a coarse network and a fine network. When adopting the hierarchical volume sampling, a set of N_c locations is firstly sampled by stratified sampling, and the coarse network is evaluated at these locations according to equation 3.3 and equation 3.4. To achieve these, the alpha composited color from the coarse network $\hat{C}(\boldsymbol{r})$ is rewritten as a weighted sum of all sampled colors c_i along the ray:

$$\hat{C}(\boldsymbol{r}) = \sum_{i=1}^{N_c} w_i(c)_i, \ where \ w_i = T_i(1 - exp(-\sigma_i\delta_i)) \tag{3.6}$$

and the weights are normalized as in 3.7, which can produce a piecewise-constant PDF along the ray

$$\hat{w}(i) = \frac{w_i}{\sum_{j=1}^{N_c} w_j} \tag{3.7}$$

With the output of the coarse network known, a biased sampling of points along each ray is then performed where samples are chosen according to the relevance of the volume. This is achieved by sampling the second set of N_f locations based on the piecewise-constant PDF along the ray using inverse transform sampling. With the combination of samples N_c and N_f, the fine network is then evaluated and the final rendered color of the ray $\hat{C}_f(\boldsymbol{r})$ is then computed (using both sets of samples). This coarse-to-fine strategy can ensure more samples are allocated to regions that have more visible contents and contribute more to the rendered scene.

3.4 Camera parameters and Normalized Device Coordinate

Camera parameters include extrinsic parameters and intrinsic parameters. The extrinsic parameters consist of a translation and a rotation, which define the location and the

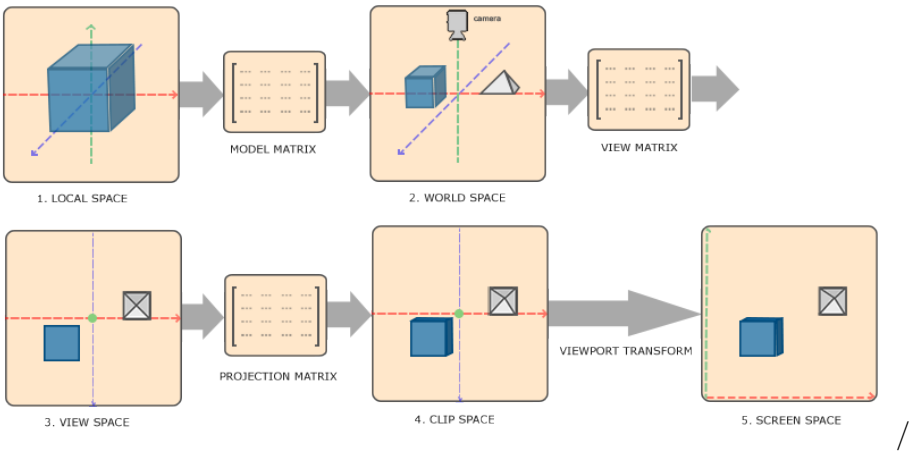

Figure 3.2: The process of coordinate transformation.

orientation of the camera relative to the world space. The intrinsic parameters consist of a focal length f, principle points c_x and c_y for a pinhole camera model, and the skew coefficient. The camera intrinsic matrix k is defined as:

$$\begin{bmatrix} f_x & 0 & 0 \\ s & f_y & 0 \\ c_x & c_y & 1 \end{bmatrix} \tag{3.8}$$

Generally, camera principle points are defined as the center of the camera, which means that $c_x \approx W/2$ and $c_y \approx H/2$, where H and W are the height and the width of the image.

A coordinate system is a system that uses coordinates to determine the position of the points on a manifold. In the field of computer vision, the vertices of the target object are normally transformed into several coordinate systems. Figure 3.2 illustrates the process. As can be seen, the vertices coordinates are firstly transformed from local space to world space. Local coordinates of an object are a set of x, y, and z axes associated with each node of that object, while world coordinates define the position of the object (relative to other objects) in the world. Then the world coordinates are transformed into view space (also known as camera space), where the coordinates are seen from the camera or viewer's point of view. The next step is to project the view-space coordinates to clip-space coordinates, and finally transform the clip coordinates to screen coordinates. This 3D to 2D projection determines which vertices will be presented on the screen. In the application of Neural Radiance Fields, when reconstructing a "forward-facing" scene, the camera coordinates are transformed to a Normalized device coordinate (NDC) space which is a screen coordinate system. In an NDC space, the x, y, and z coordinates of each vertex should be ranged between -1.0 and 1.0, and coordinates outside this range will not be visible.

The transformation process is briefly explained as follows. The standard 3D perspective projection matrix for homogeneous coordinates is:

$$M = \begin{bmatrix} \frac{n}{r} & 0 & 0 & 0 \\ 0 & \frac{n}{r} & 0 & 0 \\ 0 & 0 & \frac{-(f+n)}{f-n} & 0 \\ 0 & 0 & -1 & 0 \end{bmatrix} \tag{3.9}$$

where n, f define the near and far clipping planes and r and t are the right and top bounds of the scene at the near clipping plane. A homogeneous point $(x, y, z, 1)$ is projected from 3D to 2D by left-multiplying by M:

$$\begin{bmatrix} \frac{n}{r} & 0 & 0 & 0 \\ 0 & \frac{n}{r} & 0 & 0 \\ 0 & 0 & \frac{-(f+n)}{f-n} & 0 \\ 0 & 0 & -1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{n}{r}x \\ \frac{n}{t}y \\ -\frac{f+n}{f-n}z - \frac{2f_n}{f-n} \\ -z \end{bmatrix} \tag{3.10}$$

Then divide by the fourth coordinate:

$$\begin{bmatrix} \frac{n}{r}x \\ \frac{n}{t}y \\ -\frac{f+n}{f-n}z - \frac{2f_n}{f-n} \\ -z \end{bmatrix} \rightarrow \begin{bmatrix} \frac{n}{r}\frac{x}{-z} \\ \frac{n}{t}\frac{y}{-z} \\ -\frac{f+n}{f-n} - \frac{2f_n}{f-n}\frac{1}{-z} \end{bmatrix} \tag{3.11}$$

The projected point is now in normalized device coordinate (NDC) space, where the original viewing frustum has been mapped to the cube $[1, 1]^3$. Rendering the forward facing scene in a NDC space brings some convenience because in forward-facing captures the far bound of the scene is assumed infinity. Converting to NDC can preserve parallel lines while converting the z axis (camera axis) to be linear in disparity.

4 Methodology

Learning the implicit neural representations by means of continuous functions parameterized by MLPs has been proved to be effective in computer vision fields. Different from traditional representations which use meshes or voxel grids to represent objects, scene geometry, and appearance, MLPs take low-dimensional coordinates as inputs and are optimized to output a representation of shape, volume density, and color for each input. As a result, these MLPs are often called "coordinate MLPs". Coordinate MLPs unlock the possibility of leveraging gradient descent and deep learning methods in the field of computer vision. The performance of coordinate MLPs has been compelling, thus they are used to perform various tasks such as representing shape [11], occupancy [30], signed distance [37], volume density [32], etc. As mentioned in chapter3, NeRF also benefits from leveraging coordinate MLPs to learn and represent the volume density and the emitted color for each input 5D coordinate.

However, an analysis [52] which models the behavior of deep networks using kernel regression with a neural tangent kernel regression shows that simple MLPs are limited to learning high-frequency functions due to a phenomenon called "spectral bias" [3][38] [41]. According to the analysis, the high-frequency fitting error decreases exponentially slower than the low-frequency error. This means that simple coordinate MLPs (normally equipped with ReLU activation) have difficulty representing the high-frequency content in the signals. To address this issue, multiple attempts have been proposed. One effective attempt is to leverage a heuristic sinusoidal mapping of input coordinates, which is also called "positional encoding". Positional encoding is first introduced in the Neural Language processing fields, where it is used to describe the 1D position of an entity in a sequence. The use of positional encoding in Neural Radiance Fields proves to be capable of improving the coordinate MLPs and allowing MLPs to approximate a higher frequency function more easily.

4.1 Positional Encoding in NeRF

Mildenhall et al. [32] demonstrate that directly feeding the x,y,z,θ,ϕ coordinates to the MLPs returns poor performance on representing high-frequency variation in color and geometry in NeRF. This is because neural networks are biased towards learning lower frequency functions [38][40]. Therefore, mapping the inputs to a higher dimensional space before feeding them to the MLPs is a necessary step. The positional encoding in NeRF uses the sinusoidal functions as the encoding function which maps the input to a higher dimensional space on sinusoidal frequency bases. The encoding function is defined as:

$$\gamma(\boldsymbol{x}) = [\boldsymbol{x}, \gamma_0(\boldsymbol{x}), \gamma_1(\boldsymbol{x}), ..., \gamma_{L-1}(\boldsymbol{x})] \in \mathbb{R}^{3+6L} \tag{4.1}$$

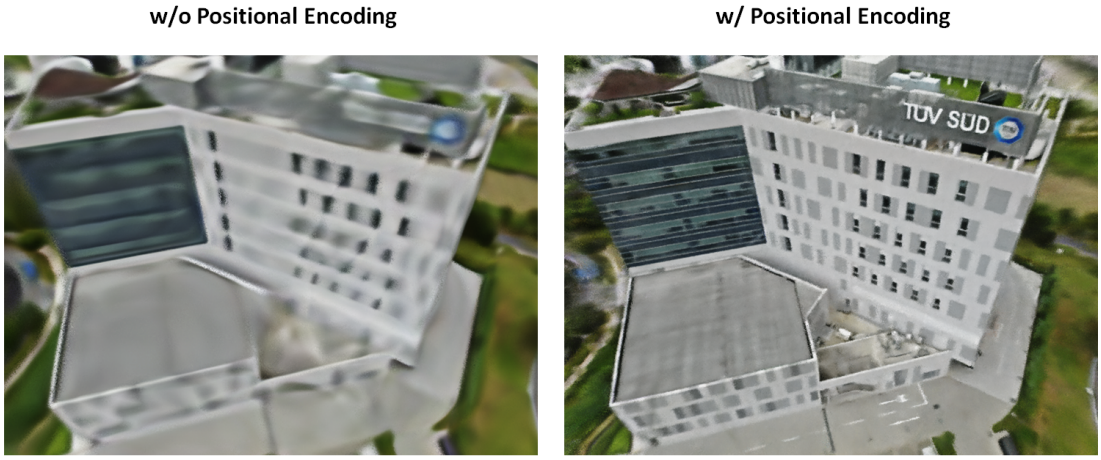

Figure 4.1: We show the difference between synthesizing our target scene using NeRF with and without Positional Encoding.

For the $k-th$ frequency, the encoding function $\gamma_k(\boldsymbol{x})$ is:

$$\gamma_0(\boldsymbol{x}) = [cos(2^k\pi\boldsymbol{x}, sin(2^k\pi\boldsymbol{x}] \in \mathbb{R}^6 \tag{4.2}$$

In this way, the inputs are mapped from $\mathbb{R}^3$ to $\mathbb{R}^{3+6L}$, where L defines the number of the frequency bases. The encoding functions $\gamma(\cdot)$ are separately applied to three location coordinates xyz and viewing direction unit vector $\boldsymbol{d}$. Therefore, L for the 3D location inputs and L for the viewing direction are set differently based on the network structure and the data. Figure 4.1 shows how positional encoding affects the view synthesis performance on our target scene. Positional encoding can significantly improve the ability of NeRF model to depict details of the scene and represent objects with complex shapes and structures (small windows on the wall, texture of the ground, reflection on the glass, etc).

4.2 Fourier Features

Other than mapping the input with sinusoidal functions in NeRF, a series of mappings called Fourier features are proposed by Tancik et al. [52]. Among these Fourier features, one mapping leveraging Gaussian distribution shows higher flexibility and capability of representing high-frequency contents. Function 4.14.2 shows that positional encoding maps the input coordinates along the axes, either X-axis or Y-axis. Take image reconstruction tasks as an example, the positional encoding assumes that the variation of the pixel values (color, shape, content) only changes rapidly along the axes of the image. However, this mapping is limited when the variation of the pixel values does not only fluctuate along the X or Y-axis but also fluctuates along random directions. To address this issue, Gaussian mapping enables mapping along random directions by concatenating the mappings along X-axis and Y-axis. Gaussian mapping is defined as follows:

$$\gamma(\boldsymbol{x}) = [\boldsymbol{x}, \gamma_0(\boldsymbol{x}), \gamma_1(\boldsymbol{x}), ..., \gamma_{L-1}(\boldsymbol{x})] \in \mathbb{R}^{3+6L} \tag{4.3}$$

For the $k-th$ frequency, the mapping $\gamma_k(\boldsymbol{x})$ is:

$$\gamma_0(\boldsymbol{x}) = [cos(2\pi B\boldsymbol{x}, sin(2\pi B\boldsymbol{x}] \in \mathbb{R}^6 \tag{4.4}$$

In function4.4, each entry in $\boldsymbol{B}$ is sampled from $\mathcal{N}(0, \sigma^2$. σ is hyperparameter which should be chosen carefully for each task and dataset.

4.3 Parameterized Activations in Coordinate MLPs

Apart from mapping the inputs to a higher dimensional space with the help of Fourier features, some other attempts focus on the activation functions in the coordinate MLPs. Most of the coordinate MLPs use ReLU as the activation function. However, ReLU-MLP without a positional encoding cannot represent the high-frequency content in signals, which leads to a blurry and insufficient reconstruction of the images in the view synthesis tasks as shown in figure 4.1. This is partly because the properties of ReLU (ReLU can be regarded as a piecewise-linear function, whose first-order derivative is either 1 or 0, and the second-order derivative is 0) significantly limit the ability of MLP to represent high-frequency signals. In this section, we first discuss how activations affect the capacity of coordinate MLPs in memorization and generalization and why parameterized activations can handle high-frequency signals better than ReLU. Then, we introduce an attempt that uses the sine function to replace ReLUs in coordinate MLPs [48] to overcome the limitations of ReLU-MLP. We implement this attempt in NeRF and propose a SIRENeRF framework as an alternative method in our view synthesis task. Further, we draw inspiration from a recent work [39] which extends the use of the sine function in MLP to a class of non-periodic functions and proposes a trainable activation scheme that not only benefits from being parameterized as the same as sine functions but also enjoys higher flexibility and efficiency to different tasks and data. We also combine our proposed activation scheme with NeRF and propose a TARF(NeRF with trainable activations) framework which serves as an embedding-less method for our view synthesis task.

4.3.1 Memorization and generalization of a coordinate MLP

The efficacy of a coordinate-MLP significantly depends on its ability to memorize and generalize the training data. Ramasinghe et al. [39] research the key factors that affect the memorization and generalization abilities of a coordinate-MLP.

A coordinate-based MLP with L layers can be formulated as:

$$F(\boldsymbol{x}) = (g^{[L]} \circ \phi^{[L-1]} \circ g^{[L-1]} \circ ...\phi^{[1]} \circ g^{[1]})(\boldsymbol{x}^{[1]}) + \boldsymbol{b}^{[L]} \tag{4.5}$$

where $g^{[l]} = \boldsymbol{W}^{[l]}\boldsymbol{x}^{[l]} + \boldsymbol{b}^{[l]}$, $\boldsymbol{W}^{[l]}$ are trainable weights at the l^{th} layer, $\boldsymbol{b}^{[l]}$ is the bias, and $\phi^{[l}(\cdot)$ is the activation function. Considering only the last layer of a MLP (normally in a k-layer MLP the last layer is linear, while the non-linearity comes from the preceding $k-1$ hidden layers), assume that the input is X, which equals to:

$$X \in \mathcal{R}^{D\times N} := [\phi(x_1)^T \phi(x_2)^T ... \phi(x_N)^T] \tag{4.6}$$

where N is the dimension of the layer, D is the depth of the MLP network, and $X_{n}{}_{n=1}^{N}$ is the raw inputs. The output of the last layer is then an affine projection of X:

$$\hat{Y} \in \mathcal{R}^{q\times N} := A^k X \tag{4.7}$$

Suppose $Y \in \mathcal{R}^{q\times N}$ is the ground truth that the MLP tries to reconstruct, a perfect memorization means that the learned $\hat{Y}$ is very close or even equal to the ground truth Y. Observe that the memorization depends on the rank of X. A higher rank can lead to a better reconstruction, and perfect reconstruction of Y can be guaranteed if rank(X) equals N. Therefore, the memorization ability of a coordinate MLP correlates with the ability of non-linear hidden layers to induce high-rank representations.

The generalization ability of an MLP correlates to the Lipschitz smoothness of the activation function $\phi(\cdot)$. Natural signals may have varying local Lipschitz smoothness, for example, in the view synthesis tasks, an input image may consist of highly fluctuating areas where the variations of pixels values are high and constant areas where the variations are relatively low. The local Lipschitz smoothness thus varies among these fluctuating and constant areas. The ability to construct the representations with varying local Lipschitz smoothness for an MLP is highly related to the first and second-order gradients of the activation function [39].

To be able to encode the signals with high fluctuations (same meaning as 'high frequency), the activation function should meet two requirements. The first requirement is that the activation functions should be parameterized so that the magnitude of the first-order derivative of the activation function can be controlled via the parameters. This is because the upper-bound on the Lipschitz constant of the angle variation in a local interval can be increased by increasing the local Lipschitz constant of the activation function, while the local Lipschitz constant of the activation function correlates to the magnitude of its first-order derivative. Therefore, a higher magnitude of the first-order derivative brings better performance to encoding signals with large fluctuations. However, though an activation with a high magnitude of the first-order derivative can handle high-frequency signals, it is not good at handling smooth signals with fewer fluctuations. Therefore, a more suitable activation function should be able to control the magnitude of its first-order derivative via parameters based on different signals. The first requirement is however not necessarily sufficient for ensuring a good performance of encoding signals with extremely varying fluctuations across different intervals. To handle these situations, the activation function should also have non-negligible second-order derivatives to obtain varying Lipschitz smoothness. This allows the linear layer in the MLP to project the points to different regions of the activation function and achieve varying local Lipschitz smoothness.

it is apparent that ReLU activation $\phi(x) = max(0, x)$, as the most commonly used activations in coordinate MLP, does not meet the both requirements. The first-order derivative of the ReLU function is either 1 or 0, while the second-order derivative is 0. Some other popular activation functions such as Sigmoid, Tanh, and PReLU violate the requirements either. To address these issues, sine function $\phi(x) = sin(wx)$ shows up as an alternative activation function. The first-order derivative of the sine function is

$wcos(wx)$, thus its magnitude can be adjusted by a parameter w, and it also has a non-negligible second-order derivative as $-w^2 sin(wx)$. Therefore, the sine function satisfies both requirements. In section 4.3.2, we will discuss how the sine function can encode higher frequency content in the signals better because of its better generalization ability, and how to leverage the sine function in NeRF and eliminate the need for positional encoding.

4.3.2 SIREN: MLPs with Periodic Activations

SIREN [48] is a neural network architecture that uses sine functions as activations:

$$x^{[l]} \mapsto \phi^{[l]}(x^{[l]}) = sin(2\pi w x^{[l]}), \tag{4.8}$$

$$\Phi(x) = \boldsymbol{w}_n(\phi_{n-1} \circ \phi_{n-2} \circ ... \circ \phi_0)(x) + b_n, x^{[l]} \mapsto \phi^{[l]}(x^{[l]}) = sin(2\pi w x^{[l]}), \tag{4.9}$$

where w is a hyperparameter.

The periodicity brought from the sine function helps SIREN cope with high-frequency content in the target signals better than ReLU-MLP. Observing the first-order derivative $-w^2 \pi sin(wx)$ of the sine function $\phi(x) = sin(2\pi wx)$, it can also be regarded as a phase-shifted sine as the derivative of sine is a cosine. Therefore, the derivatives of a SIREN can be regarded as the SIREN itself. According to the analysis in section 4.3.1, this periodic property also brings a better generalization in SIREN. The inheritance between the derivatives allows SIREN to retain the same ability to represent signals in the derivatives. Embedding SIREN in NeRF can eliminate the need for positional encoding, and significantly decrease the number of neurons used in the network. The decrease of weights to be optimized can lead to faster convergence and a shorter training time in NeRF.

4.3.2.1 A principled initialization scheme designed for SIREN

Experiments show that SIRENs are very sensitive to the weight initialization scheme of the MLPs. Popular initialization schemes such as Xavier uniform initialization and Xavier normal initialization cannot ensure effective training of SIRENs. To solve this problem, a principled initialization scheme is specially designed for SIREN [48], which should be strictly followed to avoid a significant drop in training performance. The key idea behind the proposed initialization scheme is that to eliminate the dependency between the output at initialization and the number of layers, the distribution of activations should be preserved throughout the whole network.

To achieve this goal, the input to a neuron should be kept the same as the distribution of its output. Since SIRENs are used to take coordinates as the inputs, assume an input X which is uniformly distributed in the interval $[-1, 1]$. Passing this input to a simple sine function will generate an output $Y = sin(\frac{\pi}{2}X)$ which meets $Y \sim Arcsin(-1, 1)$. The Arcsin distribution is defined as:

$$X \sim Arcsin(a, b), with\ CDF : F_X(x) = \frac{2}{\pi} arcsin(\sqrt{\frac{x-a}{b-a}}, with\ b > a. \tag{4.10}$$

The variance of the outputs meets 4.11:

$$For\ X \sim Arcsin(a,b), Val[X] = \frac{1}{8}(b-a)^2 \tag{4.11}$$

Then, the next layer will take these Arcsin distributed outputs as the new inputs, and a linear combination will be calculated. According to Lindeberg's condition for the central limit theorem, the linear combination will be normal distributed. The variance of the linear combination can also be calculated via :

$$Val[X] = Val[X] \cdot Val[Y] + E[Y^2]Val[X] + E[X^2]Val[Y] \tag{4.12}$$

Now, assume that at the layer $l+1$, the linear combination is (for simplification, the bias is not considered):

$$w_l^T X_l = \sum_i^n w_{i,l} X_{i,l} \tag{4.13}$$

By initializing the weights as $w_i^l \sim \mathcal{U}(-c,c)$, the variance of the linear combination is then:

$$Var[w_l^T X_l] = Var[w_l] \cdot var[X_l] = \frac{1}{12}(2c)^2 \cdot \frac{1}{2} = \frac{1}{2}c^2 \tag{4.14}$$

If choosing $c = \sqrt{\frac{6}{n}}$ with the fan-in n, then the variance can be calculated as $Var[w_l^T X_l] = n \cdot \frac{1}{6}\frac{6}{n}$ according to Central Limit Theorem. Meanwhile, a Normal distribution of the output $w_l^T X_l \sim \mathcal{N}(0,1)$ can be gained, which is the same as the distribution of the input.

To sum up, When initializing each element of the weight w to meet the uniform distribution $w_i \sim \mathcal{U}(-c/\sqrt{n}, c/\sqrt{n}), c \in \mathbb{R}$, the linear combination at each linear layer will meets the normal distribution $w^T X \sim \mathcal{N}(0, c^2/6)$, and passing this normally distributed linear combination into another sine activation will also yield an output that meets the arcsin distribution if $c > \sqrt{6}$. In this way, the distribution of the output from each layer remains the same throughout the whole network.

4.3.2.2 The choice of w

w in the sine function $\phi(x) = sin(2\pi wx)$ is an important hyperparameter. A larger w increases the bandwidth of the network, allowing it to encode increasingly higher frequency functions.

4.3.3 A Trainable Activation Scheme for Coordinate MLPs

Despite Sitzmann et al. [48] have theoretically and empirically proven that sine activations used in SIREN can successfully achieve a performance as good as the positional encoding on encoding high-frequency signals, there are still some limitations existing for sine activations. Firstly, applying sine activation requires strictly adhering to the initialization scheme proposed in [48], otherwise, the performance drops significantly. Secondly, sine activations bring more hyperparameters to the network compared to

Figure 4.2: we compare the results from SIREN with Xavier initialization scheme and SIREN with the proposed initialization scheme[48] in our view synthesis task. The results show that without the carefully designed initialization scheme, a deep SIREN network might not be able to converge and the performance significantly drops.

normal activation functions such as ReLU or sigmoid, and tuning the w in each SIREN layer takes more time and effort. To address these issues, Ramasinghe et al. [39] propose a unifying framework for the use of activation functions in coordinate MLPs, showcasing that non-periodic activations can also be suitable for encoding signals and bring more robustness against different initialization schemes. Based on the research on non-periodic functions used in coordinate MLPs, we combine them with NeRF and explore their applications in 3D view synthesis fields. Moreover, we simplify the way of tuning hyperparameters in these activation functions by making them trainable. This strategy saves the effort of tuning hyperparameters in activation functions and also significantly improves performance.

4.3.3.1 Non-periodic activations

Besides the sine activations used in SIREN, non-periodic functions can also serve as activation functions used in coordinate MLPs. Table 4.1 shows several non-periodic functions, including Gaussian function, Quadratic function, Laplacian function, and super-Gaussian function. Take the Quadratic function as an example, in Quadratic-based MLPs, the activation function is defined as:

$$x^{[l]} \mapsto \phi^{[l]}(x^{[l]}) = \frac{1}{1 + (ax^{[l]})^2}, \tag{4.15}$$

where a is an important parameter which affects the bandwidth of the network. a larger a corresponds to a lower bandwidth, and vise-versa. Similar to sine function, the derivatives of these non-periodic functions are not zero (unlike ReLU), and this property enables them to encode high-frequency content of the signals.

Activations	Equation	Parameterized	First-order derivative	Second-order derivative
ReLU	$max(0,1)$	✗	$\mathrm{L}_\delta = \begin{cases} 1 & if\ x < 0 \\ 0 & otherwise \end{cases}$	0
Sine	$sin(wx)$	✓	$wcos(wx)$	$-w^2 sin(wx)$
Gaussian	$e^{\frac{-0.5x^2}{a^2}}$	✓	$-\frac{xe^{-\frac{x^2}{2a^2}}}{a^2}$	$-\frac{(x^2-a^2)e^{-\frac{x^2}{2a^2}}}{a^4}$
Quadratic	$\frac{1}{1+(ax)^2}$	✓	$-\frac{2a^2x}{(a^2x^2+1)^2}$	$-\frac{2a^2(3a^2x^2-1)}{(a^2x^2+1)^3}$
Laplacian	$e^{\frac{-\lvert x\rvert}{a}}$	✓	$\frac{xe^{\frac{\lvert x\rvert}{a}}}{a\lvert x\rvert}$	$\frac{e^{\frac{\lvert x\rvert}{a}}}{a^2}$
Super-Gaussian	$[e^{\frac{-0.5x^2}{a^2}}]^b$	✓	$-\frac{bxe^{-\frac{bx^2}{2a^2}}}{a^2}$	$-\frac{b(bx^2-a^2)e^{-\frac{bx^2}{2a^2}}}{a^4}$

Table 4.1: A set of parameterized activations whose derivatives meet the requirements mentioned in section 4.3.1

4.3.3.2 Make activations trainable

In complex tasks such as 3D view synthesis, a rather deep MLP networks with more than 5 layers is often required. This means that, take Gaussian activations as an example, one more hyperparameter a is added along with each layer, and the total number adds up quickly with the number of layers increasing. Therefore, exhausting work is often required to find the best combination of these a if keeping them fixed in each layer and tuning them as hyperparameters. Besides, the flexibility of Gaussian activations should be questioned if these combination only works on their own dataset and same effort should be taken again every time the dataset changed. To overcome this limitation, we come up with a strategy, which is that, instead of tuning the hyperparameters, making the activations trainable. The a are then no longer fixed for each layer, and their value will be updated during the training. By doing this, the MLP network can enjoy more flexible activations adapted to the input data.

Experimental Setup and Results

5

5.1 Data Analysis and Pre-processing

Data preparation is of extreme importance and often underestimated in real-world tasks, especially in view synthesis tasks which are sensitive to the data. Here we will discuss our methods of cleaning the data, obtaining geometry information, and adjusting the input views included in the datasets. The first step is the data cleaning. Given thousands of images taken by drones from different angles and at different heights, one of the principles is to ensure a single-scale dataset. This means that in the dataset all the images should be taken from the same distance. We recommend a faraway view of the object (taken more than 50 meters away depending on the size of the object). This is because relatively panoramic views can ensure a higher accuracy when extracting the poses and camera parameters from the images. Another tip is that the images should be selected consecutively in a dataset. The translation and rotation of the camera between two consecutive images is lower the better because this reflects a more stable camera trajectory. A turbulent camera motion leads to higher risks of failure in pose estimation and worse scene representation [57].

The second step is to estimate the poses and extract camera intrinsic and extrinsic. NeRF is not originally an end-to-end solution for view synthesis because it relies heavily on photogrammetry. NeRF has a strict requirement on accurate camera poses and it cannot work without known poses initially. Therefore, since most of the researches on NeRF are tested and evaluated on well-prepared synthesis data with known and perfectly accurate camera parameters, obtaining accurate camera poses from noisy real-world datasets becomes an important but hidden problem in the applications of NeRF. Some extensions of NeRF [36][7] provide an solution to enable view synthesis from raw RGB images. However, these methods cannot achieve satisfying results on our own data according to the experiments. An alternative way, which is adopted in our task, is to run a SfM pipeline called COLMAP to recover the 6-DoF camera poses of the images, by which the location and direction of the camera for each image are estimated as shown in figure 5.1, and the near/far bound of the scene can also be computed.

Though COLMAP can help obtain the poses, the performance of NeRF is still strictly limited by the accuracy of pose estimation and the complexity of the target scene. Wang et al. [57] study how camera motions affect the pose estimation by COLMAP. In our task, the data is collected by a drone flying around the target building. Therefore, our images are collected in a track-to-object style. In ideal cases, a track-to-object camera is supposed to move within the xy plane all the time and turn towards the z direction. However, perturbations of the camera trajectory are inevitable in real-world scenarios because of the turbulence or manual control. Wang et al. [57] show that

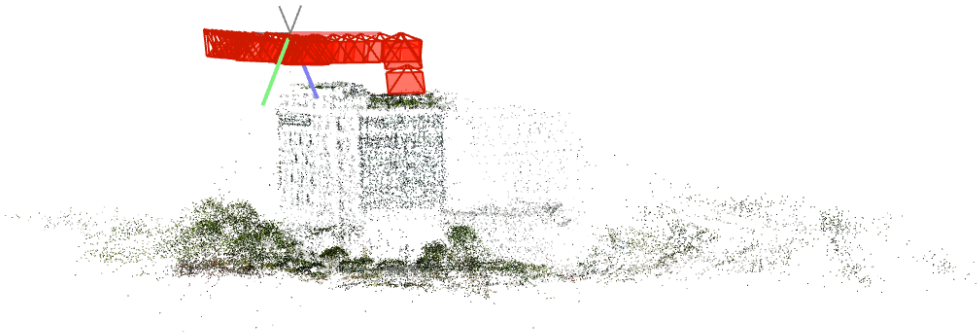

Figure 5.1: Recover the 6-DoF camera poses by COLMAP.

the breaking points for translation and rotation are ±20% in COLMAP, which means that camera motions with a rotation larger than ±20% or a translation larger than ±20% might significantly decrease the accuracy of the pose estimation from COLMAP, or even cause a failure. This finding explains why the view synthesis results on our own data are less competitive with the results on standard NeRF-LLFF data, and can also serve as a guideline for collecting data in future research or applications.

Besides, the inaccurate pose estimation also affects our choice in the data preparation. We observe that for a narrow range of forward-facing viewpoints of the scene, the camera motion can be considered averagely less turbulent and the pose estimation from COLMAP is more accurate. Meanwhile, the viewpoints are generally of higher density if the input views are carefully selected. Therefore, one strategy is to observe the target building from different angles separately, that is to say, render several narrow forward-facing views. Figure 5.2 presents how we choose the views and the range of these views. As can be seen, the whole view of the target is segmented into front-view, back-view, and side-views. The experiment results show that this strategy enables a relatively more accurate pose estimation and better view synthesis qualities. Therefore, these forward-facing views are used as major input views to test our methods in the following sections.

Another strategy is to render the 360-degree view of the target at once. This can be operated by turning off the normalized device coordinate (NDC) transformation when reconstructing the scene. NDC assumes forward-facing captures in the normalized device coordinate space. This space is convenient because it preserves parallel lines while converting the z axis (camera axis) to be linear in disparity. Without NDC, the scene can be then reconstructed in a spherical manner. Figure 5.3 shows rendering results with or without NDC on a 360-degree dataset.

However, a 360-degree spherical view synthesis faces two major difficulties. First, the input views are more sparse than those of the forward-facing dataset. This is partly because of the data collection. In our available data, the images taken from some angles are sparser than others. For forward-facing datasets, we carefully select the views with the densest taken images, while for a 360-degree dataset we have to

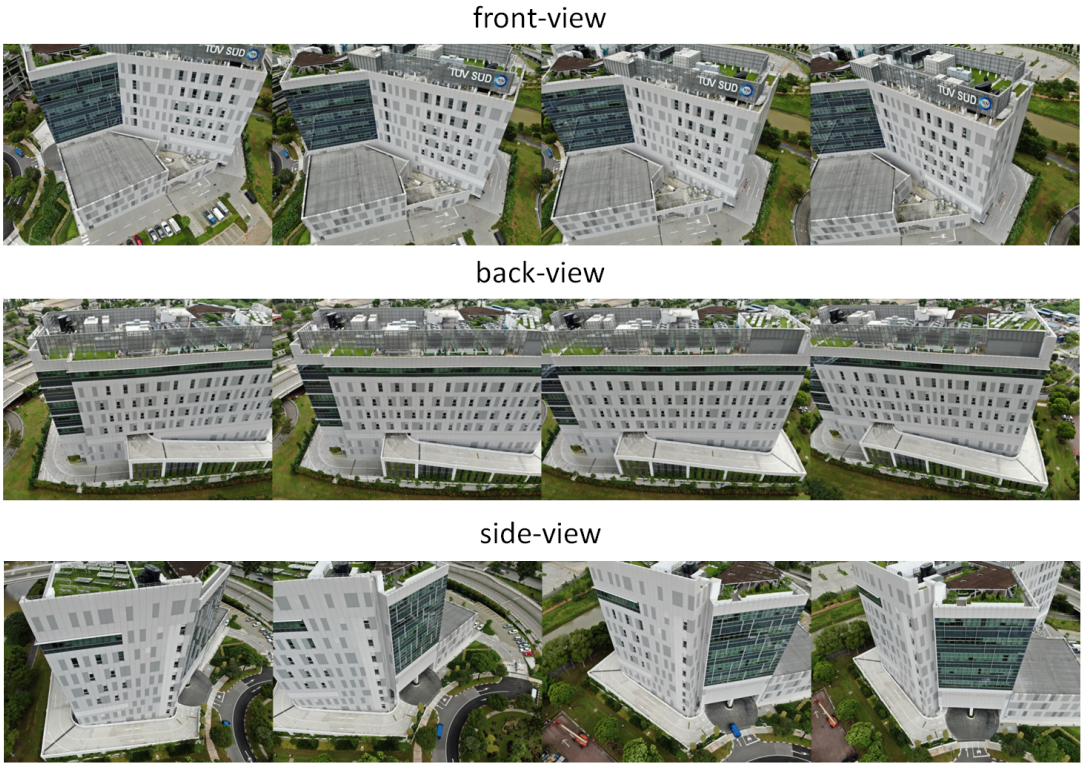

Figure 5.2: we split the 360 view of the target building into front view, back view, and side view scenes.

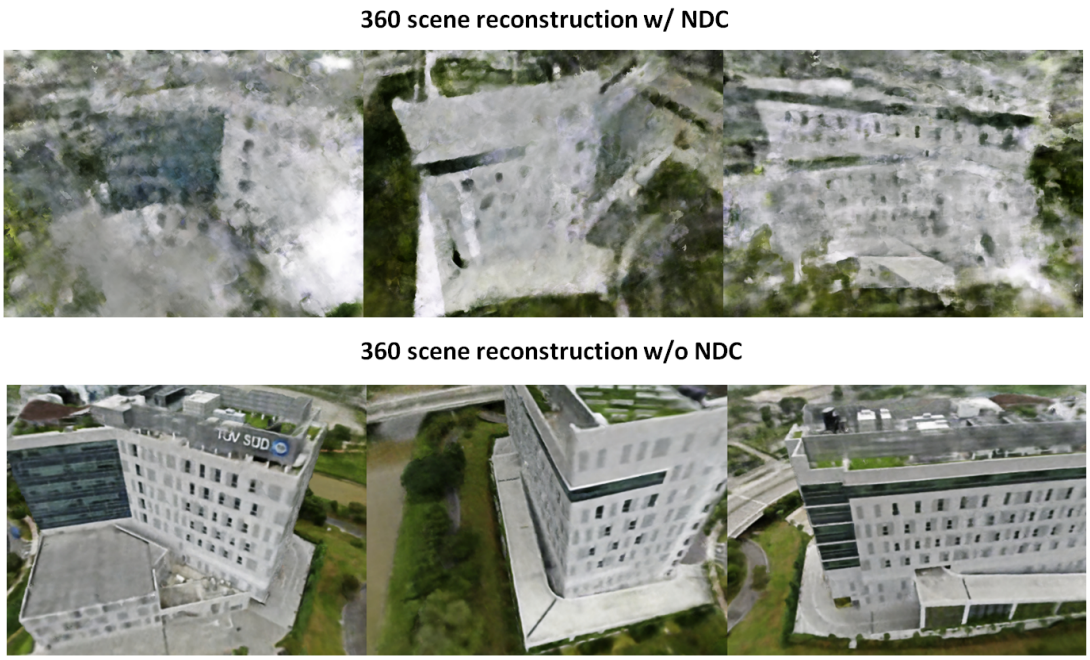

Figure 5.3: 360-inward view synthesis is problematic when NDC transformation is operated.

tolerate that some of the angles are easier for our methods to reconstruct because of the sparser inputs. Second, as mentioned above, a spherical camera motion makes it harder for COLMAP to estimate the poses. We will show that both the qualitative and quantitative results on the 360-degree dataset are worse. Therefore, this 360-degree dataset will be regarded as an additional dataset that is only used to validate our methods in the following experiments.

5.2 Quantitative Evaluation

Evaluating the results in the computer vision and view synthesis tasks can be very subjective, and the standards of the image quality could vary from person to person. Therefore, it is also significant to find a quantitative standard to measure the results, for example, the quality of the images, coming from different methods.

In our experiments, three metrics including Peak Signal to Noise Ratio (PSNR), Structural Similarity Index Metric (SSIM), and Learned Perceptual Image Patch Similarity (LPIPS) [58] are taken into consideration as the quantitative evaluation metrics.

5.2.1 Peak Signal to Noise Ratio

Peak Signal to Noise Ratio (PSNR) measures the ratio between the maximum power of a signal and the power of distorting noise that affects the signal. The unit of PSNR is decibel (dB). PSNR is a common metric that is widely used in novel view synthesis methods such as NeRF and its many extensions. PSNR can measure how closely a learned representation resembles its original ground truth images. PSNR can be calculated as in 5.1:

$$PSNR = 20log_{10}(\frac{MAX_f}{\sqrt{MSE}}) \tag{5.1}$$

where MSE is the mean square error expressed as:

$$MSE = \frac{1}{mn}\sum_{0}^{m-1}\sum_{0}^{n-1} \parallel f(i,j) - g(i,j) \parallel^2 \tag{5.2}$$

MSE is used to compute the difference in pixel values between the generated image and the ground truth image, thus the MSE measures the average of the squares of the errors. In 5.1 and 5.2, f represents the matrix of the ground truth image, g represents the synthesized image (learned representation). m represents the number of rows of pixels of the images and i represents the index of that row$n$ represents the number of columns of pixels of the image and j represents the index of that column. For PSNR, the value is higher the better.

5.2.2 Structural Similarity Index Metric

A limitation of PSNR is that it can only compare two images pixel by pixel, and no biological factors about the similarity between images are taken into consideration.

To address this issue, The structural similarity index measure (SSIM) is introduced. SSIM is a method used to predict the perceived quality of images. SSIM measures the similarity between two images. Pixels that are spatially close to each other have inter-dependencies. These dependencies carry structural information, which is used in SSIM. SSIM is calculated as 5.3:

$$\begin{aligned} SSIM(x,y) &= l(x,y)^{\alpha} \cdot c(x,y)^{\beta} \cdot s(x,y)^{\gamma} \\ &= \frac{(2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_x^2 + c_1)(\sigma_x^2 + \sigma_x^2 + c_2)} \end{aligned} \tag{5.3}$$

When α, β, γ equal to 1. In 5.3, x is the ground truth image while y is the generated image. μ_x, μ_y are averages of x, y, σ_x^2, σ_y^2 are variances of x, y, σ_{xy} is the covariance of x and y. c_1 and c_2 are two variables that stabilize the division with a weak denominator. SSIM has a range between 0 and 1. Closer to 1 means that two images have higher structural similarity.

5.2.3 Learned Perceptual Image Patch Similarity

Though PSNR and SSIM are widely used, they are believed to be very simple and shallow assessments that cannot measure many nuances of human perception. To better assess the perceptual similarity between images, a method named Learned Perceptual Image Patch Similarity (LPIPS), which is also often called "Perceptual Losses", is proposed by Zhang, Richard, et al [58]. With the help of pre-trained CNN networks, LPIPS is capable of matching human perceptual judgements on the images well according to the experiment results [58]. A lower LPIPS value means that two images resemble more to each other perceptually.

5.3 Experiment results

In this section, we show the experiment results of our attempts to improve the performance of NeRF on our view synthesis task. The baseline method in all the experiments is NeRF [32], which is referred to as ref-NeRF in the following sections. In section 5.3.1, we demonstrate the experiment results on using Gaussian distributed Fourier Features to replace the positional encoding in ref-NeRF. For simplicity, we refer to our method as GFF-NeRF. In section 5.3.3, we demonstrate the experiment results on building an embedding-less NeRF model with parameterized activations to achieve view synthesis. For simplicity, we refer to our model which leverages SIREN layers as SIRENeRF, and we refer to our models which leverage a class of trainable non-periodic activations as TARFs.

In all the experiments, the performance is evaluated from two aspects. First, we demonstrate the qualitative evaluations where synthesized images are shown. We first show the scene reconstruction capacity of our models by comparing the test images reconstructed by our models and baseline ref-NeRF with the ground truth images. Then we show and compare the novel views synthesized by both models. For the quantitative evaluation, we present the measurements of PSNR, SSIM, and LPIPS.

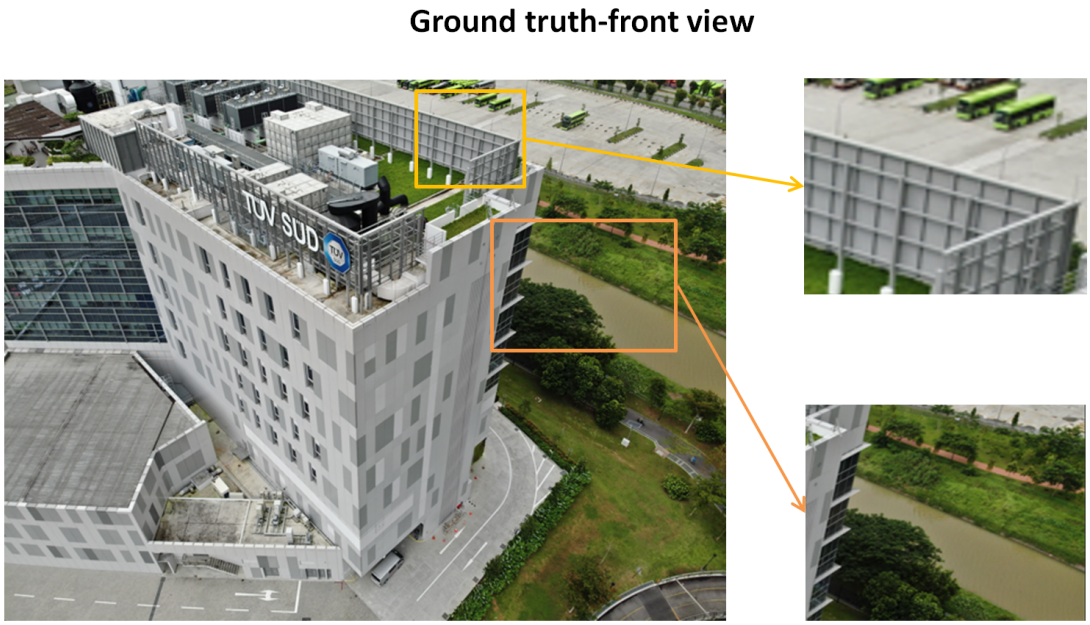

Figure 5.4: The ground truth used to test the scene reconstruction performance in the front-view scene.

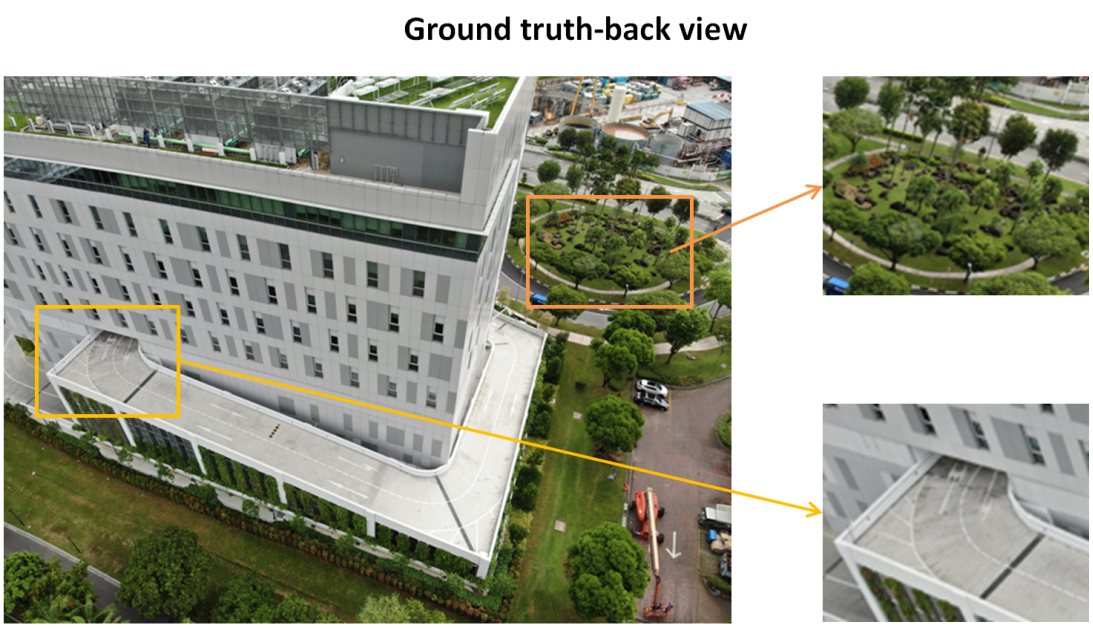

Figure 5.5: The ground truth used to test the scene reconstruction performance in the back-view scene.

In all the experiments, the performance of the models is evaluated in our three forward-facing scenes. To begin with, We present the ground truth of these three scenes in figure 5.45.55.6.

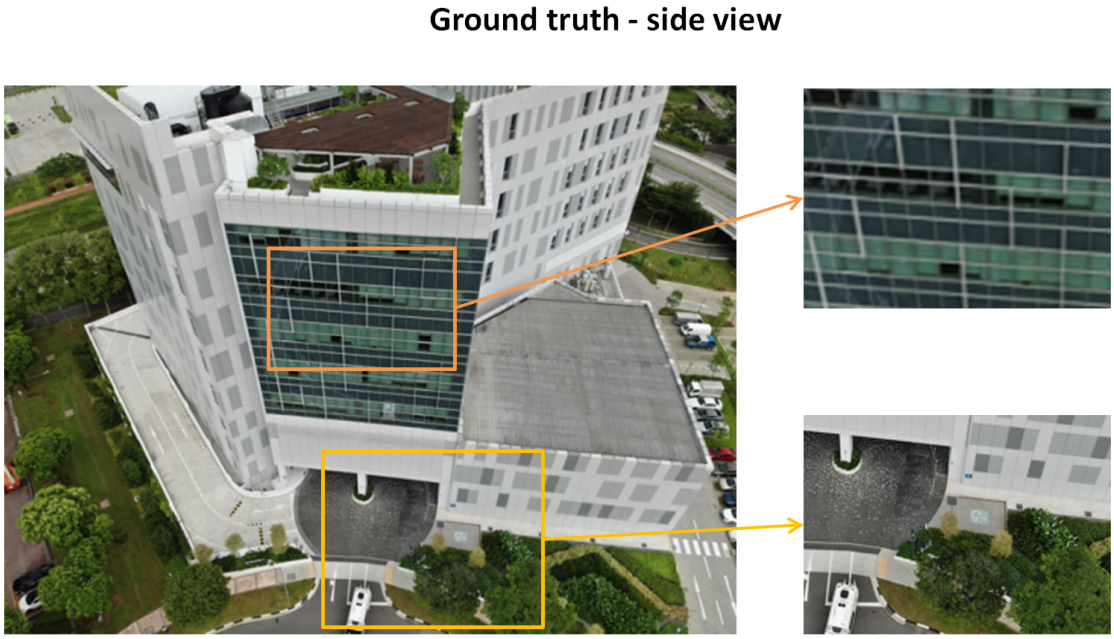

Figure 5.6: The ground truth used to test the scene reconstruction performance in the side-view scene.

5.3.1 GFF-NeRF

5.3.1.1 Experimental Settings

We compare our proposed GFF-NeRF with ref-NeRF. We use our forward-facing datasets in the experiments. We assume known intrinsic in all the cases.

5.3.1.2 Implementation details

We implement our GFF-NeRF method as follows: We train a 7-layer SIREN with 256 hidden units in each layer to learn the volume density and features, and a 2-layer SIREN with 256 hidden units in each layer to learn the RGB colors. We resize the images to 480×640 pixels and randomly sample 2048 pixel rays every iteration, each sampled at $N = 128$ coordinates. We use the Adam optimizer and train all the models for 200k iterations. For both ref-NeRF and SIRENeRF, the learning rate begins at 1×10^{-3} and decays exponentially to 1×10^{-3}. For ref-NeRF, the number of frequencies in the positional encoding is set as $L = 10$, while for our GFF-NeRF, we set the frequency as $L = 256$ following the instructions from [52].

5.3.1.3 Results

Figure 5.11 compares the reconstruction results of the front-view scene from our GFF-NeRF model and the ref-NeRF model. From the observation, we can clearly find that GFF-NeRF achieves better reconstruction of the complex background. The figure shows that ref-NeRF, as the baseline, fails to represent any objects in the background,

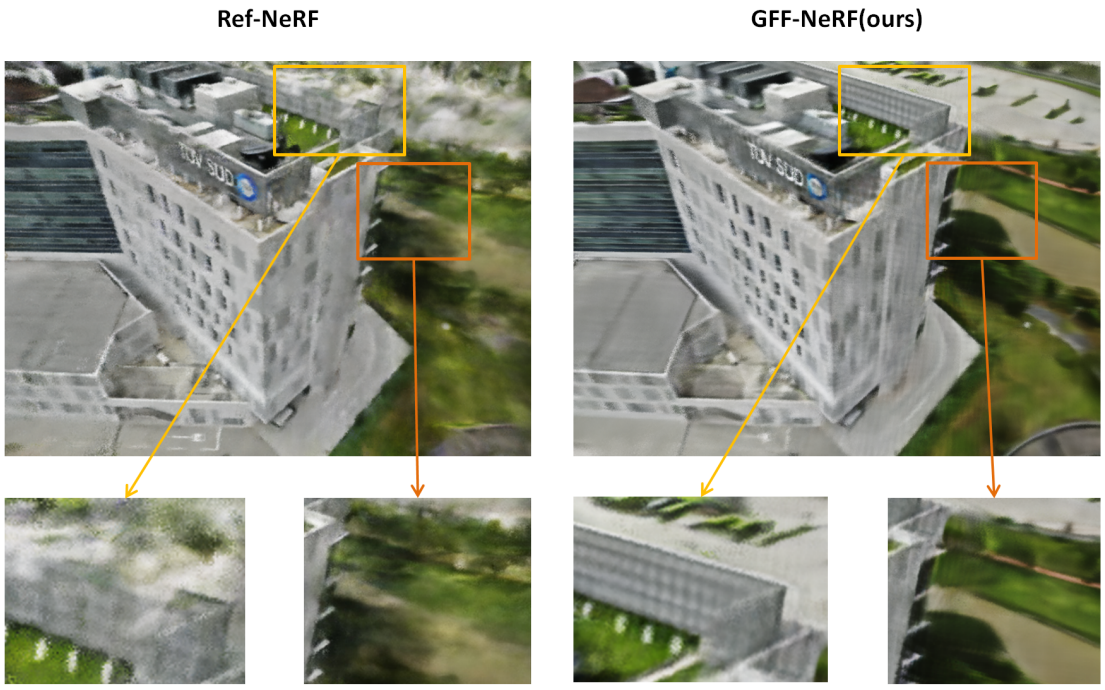

Figure 5.7: We compare the test image reconstruction results between GFF-NeRF and ref-NeRF on the front-view scene.

even the dividing lines of regions with different shapes and colors (e.g. the dividing line between the gray parking lot and green grassland). This means that ReF-NeRF is less capable of representing unrestricted scenes, where the distance between the camera and the objects in the background is various and unlimited. Our model, on the contrary, achieves a much better reconstruction of the background. By comparing the reconstruction from GFF-NeRF and the ground truth shown in figure 5.4, we can find that GFF-NeRF can represent the background with fine qualities. As can be seen, different regions are clearly divided with much slighter blur and noise, and the shape of the small objects is also sketched. The zoom-in figures on the bottom 5.11 show more details. As the reconstruction of tiny objects such as vehicles in the parking lot and trees on the riverbank is a failure by ref-NeRF with huge noise and mess, our model manages to achieve a recognizable reconstruction of these objects though the qualities are not as good as the ground truth. It is also seen that GFF-NeRF achieves clearer reconstruction of the edges on our target building (e.g. edges between the gray wall and the blue glass, edges between the roads and the grasslands) and the dividing lines in the background (e.g. riverbanks that separate the yellow-green river and dark-green grassland, edge of the parking lot). Edges and dividing lines separate two regions that highly vary in pixel values. Therefore, a clearer sketch of the dividing lines proves that Gaussian-distributed Fourier features to handle the high-frequency content better than positional encoding. Besides, as can be seen from the zoom-in figure on the left-bottom that our model also improves the representation of some small but complicated structures, where the guardrails are represented with clearer details.

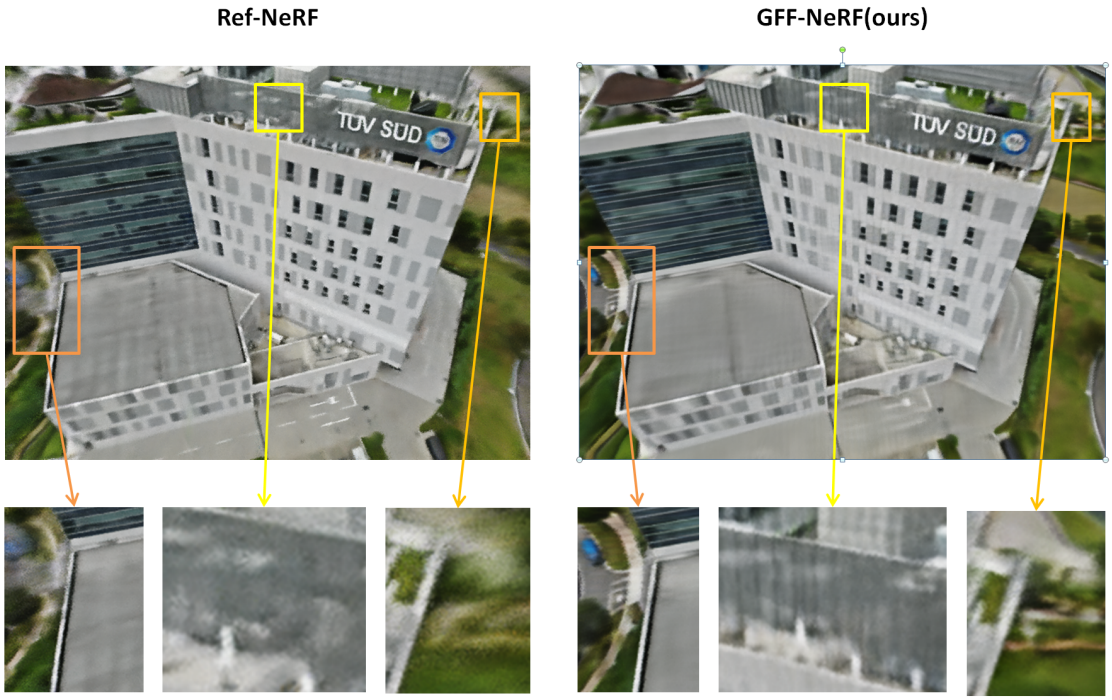

Figure 5.8: We compare the synthesized novel view from GFF-NeRF and ref-NeRF on the front-view scene.

Figure 5.8 compares the novel views synthesized by Ref-NeRF and GFF-NeRF. We can also draw similar conclusions that though both methods manage to represent the target object, GFF-NeRF performs better in representing the background such as the white traffic signs on the road shown in the zoom-in figure on the left bottom and the gray parking lot on the zoom-in figure on the right-bottom. In addition, GFF-NeRF also produces a clearer representation of some complex details such as the grids of the guardrails in the zoom-in figure on the middle bottom.

We also test our model on the side-view scene dataset. This dataset contains a wider range of viewpoints, thus the actual input views are sparser than the front-view scene dataset. Therefore, the reconstruction results are less competitive in terms of both qualitative and quantitative measurements. However, it is still shown that our model can produce better results than ref-NeRF. Figure 5.9 presents the scene reconstruction results from both methods. We can find that Gaussian-distributed Fourier features dramatically reduce the noise and can smooth the transition of fluctuating pixel regions. The zoom-in figures on the bottom support our observation that our model outperforms ref-NeRF by improving the resolution and depicting cleaner edges and lines. As for the view synthesis results shown in figure 5.10 which are of better quality and higher resolution than the scene reconstruction results, we can still see that our model produces a cleaner image with sharper details.

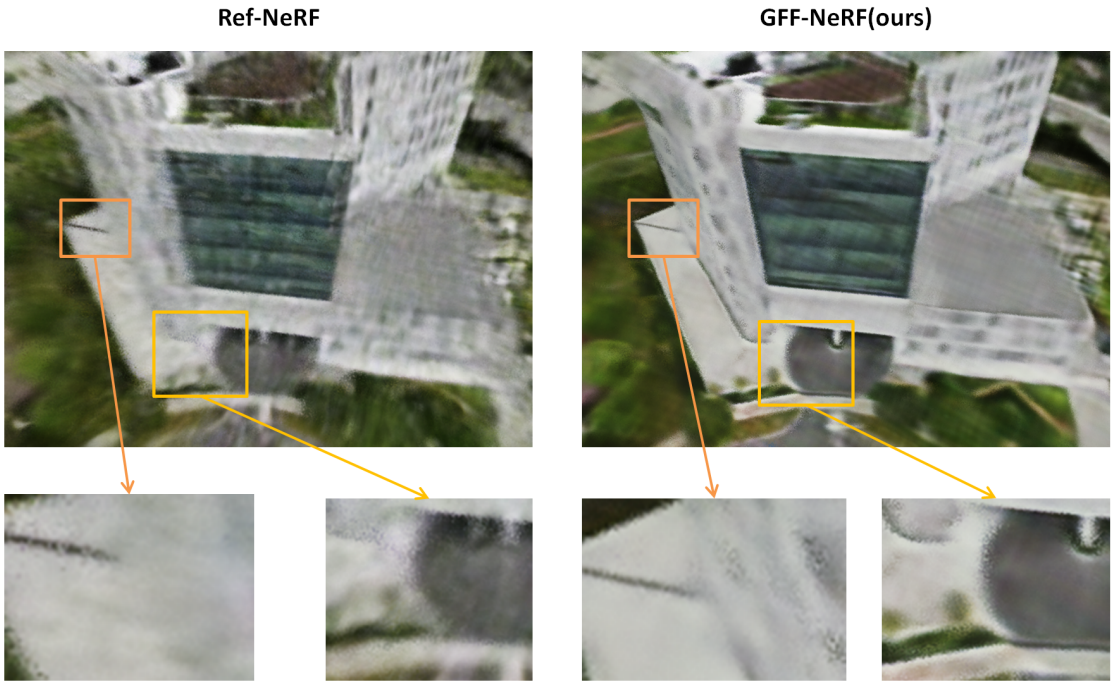

Figure 5.9: We compare the test image Reconstruction results between GFF-NeRF and reference NeRF on side-view scene.

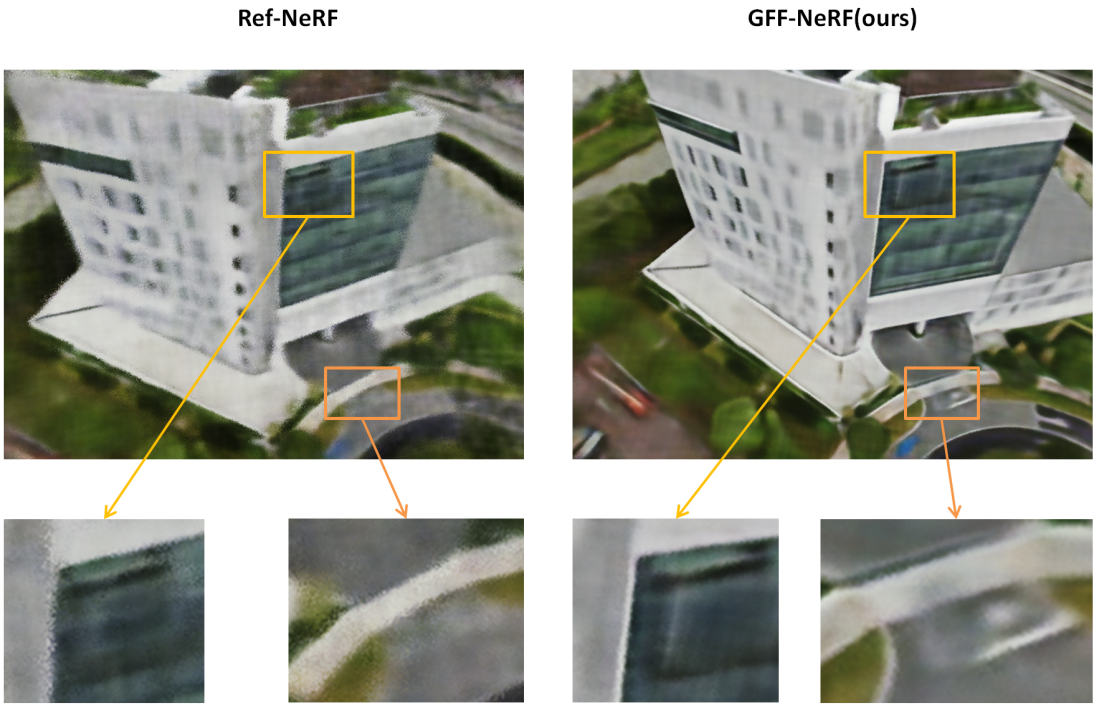

Figure 5.10: We compare the synthesized novel view from GFF-NeRF and ref-NeRF on the side-view scene.

	Front-view		
Models	PSNR ↑	SSIM ↑	LPIPS ↓
ref-NeRF	17.97	0.43	0.63
GFF-NeRF	**19.31**	**0.48**	**0.53**
	Back-view		
Models	PSNR ↑	SSIM ↑	LPIPS ↓
ref-NeRF	12.76	0.17	0.63
GFF-NeRF	**14.56**	**0.26**	**0.53**
	Side-view		
Models	PSNR ↑	SSIM ↑	LPIPS ↓
ref-NeRF	15.42	0.30	0.86
GFF-NeRF	**17.43**	**0.39**	**0.71**

Table 5.1: Comparisons of quantitative metrics between GFF-NeRF and ref-NeRF

5.3.2 SIRENeRF

5.3.2.1 Experimental Settings

We compare our proposed SIRENeRF model with ref-NeRF. We assume known intrinsic in all the cases.

5.3.2.2 Implementation details

We build and implement the SIRENeRF following the instructions from [48] with our own modifications and adjustments. We train a 8-layer SIREN with 256 hidden units in each layer to learn the volume density and features, and a 2-layer SIREN with 256 hidden units in each layer to learn the RGB colors. We resize the images to 480×640 pixels and randomly sample 2048 pixel rays every iteration, each sampled at $N = 128$ coordinates. We use the Adam optimizer and train all the models for 200k iterations. For both reference NeRF and SIRENeRF, the learning rate begins at 1×10^{-3} and decays exponentially to 1×10^{-3}. For hyperparameter w in sine activations, different from the suggestions in [48], we set $w = 5$ in the first SIREN layer, and $w = 1$ in the rest of the layers. We believe that this is the most optimal setting combination according to our experiments.

5.3.2.3 Results

We first test our SIRENeRF model on the front-view scene dataset. Figure 5.11 presents the reconstruction results of the front-view scene from SIRENeRF and ref-NeRF. We can learn from the results that SIRENeRF outperforms ref-NeRF overall. Similar to GFF-NeRF, SIRENeRF is capable of producing a significantly cleaner reconstruction of the complex background. As can be seen in the zoom-in figures on the bottom, our model handles the unrestricted scene better as it clearly depicts the objects which are of various distances to the camera. Besides, the cleaner sketch of the edges and boundaries of different regions in the image is also produced by producing a smoother image. This can prove that SIREN layers can encode the high-frequency content better than the

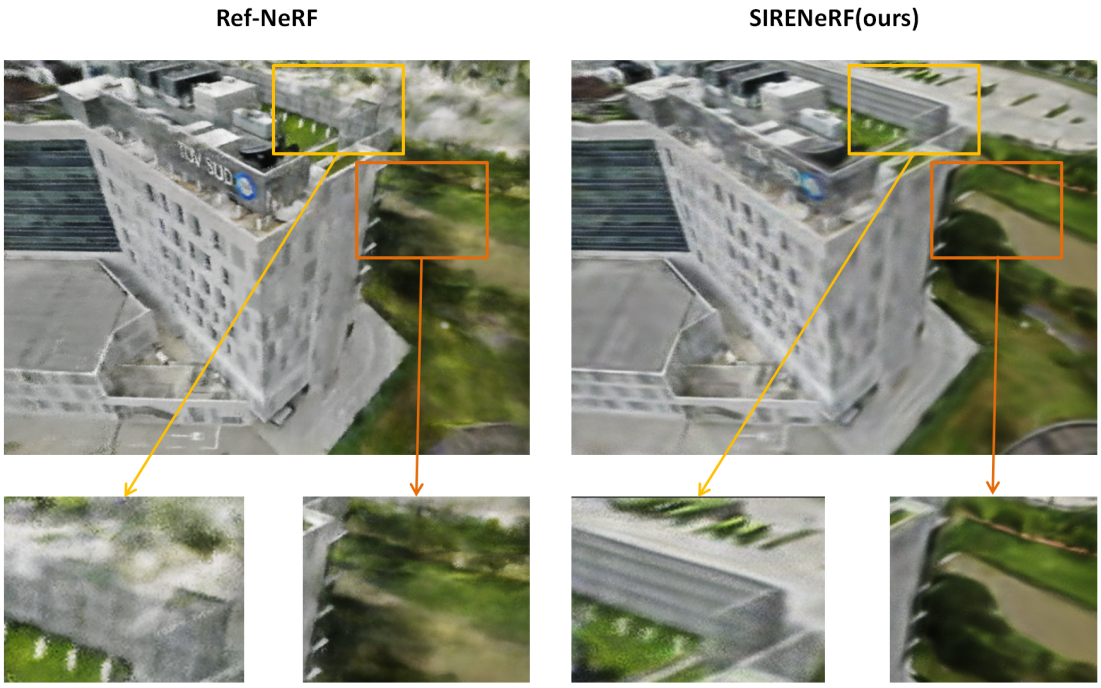

Figure 5.11: We compare the test image reconstruction results between SIRENeRF and ref-NeRF on the front-view scene.

position encoding used in ref-NeRF. The quantitative measurements presented also support this observation as can be seen in the table. However, we can also conjecture that better scores and clearer overall reconstructions are achieved at the cost of some trivial details. Comparing the results from GFF-NeRF in the figure 5.7, we can find that the target building reconstructed by SIRENeRF is less sharp, and some slight variations (e.g. the dark-gray/white color blocks on the light-gray facade) are smoothed out. This unpleasant smoothing effect also happens when high-frequency contents take up a large part of one region. For example, the sign of the company name is reconstructed in a blurry manner by SIRENeRF. This is because the sign varies a lot in terms of the RGB values compared to its surroundings but only contains a small number of pixels. When SIRENs smooth the boundary between the sign and the surroundings, it is inevitable that the major part of the sign is also affected. Therefore, a conclusion can be drawn that though SIRENeRF outperforms ref-NeRF, it is less competitive with GFF-NeRF on the front-view scene.

The novel view synthesis results also support our observation. Though both models manage to synthesize the target building, the zoom-in figures on the bottom show that SIRENeRF synthesizes a more accurate background. As ref-NeRF fails to learn the shape and position of the objects in the background, SIRENeRF can learn and represent objects such as traffic signs and small lanes. However, SIRENeRF can only produce a less sharp synthesis of the same details discussed above including the sign of the company name and color blocks on the facade.

Then we test our model on the side-view scene dataset. Figure 5.13 shows the scene

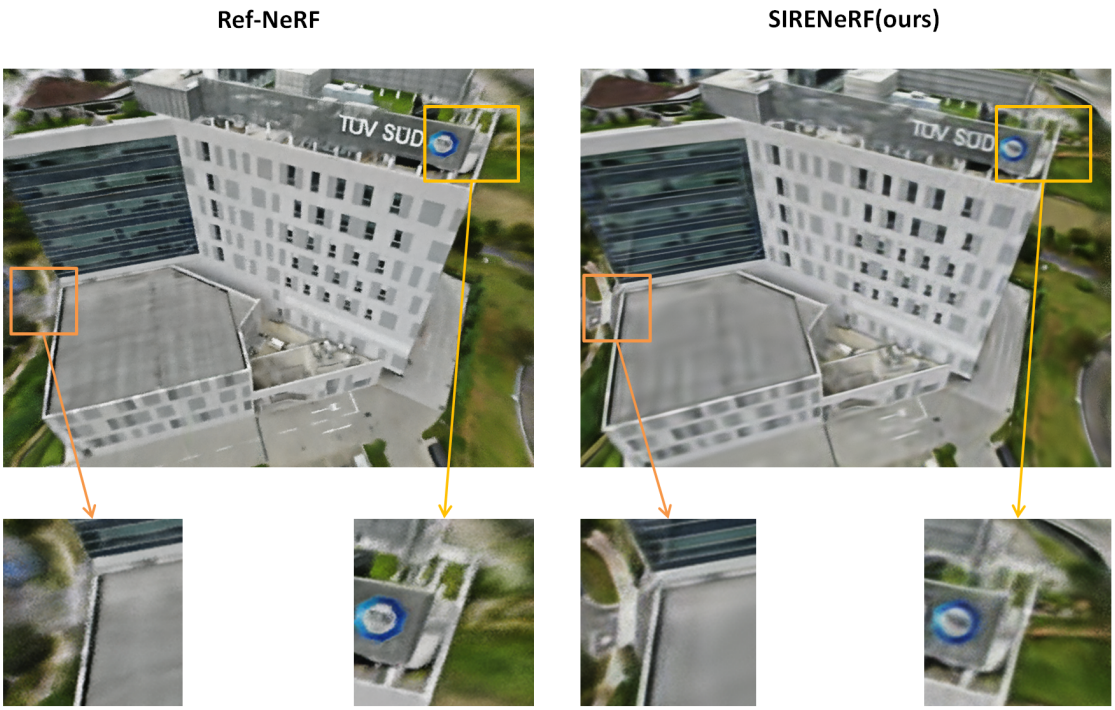

Figure 5.12: We compare the synthesized novel view from SIRENeRF and ref-NeRF on the front-view scene.

reconstruction results from SIRENeRF and ref-NeRF. As can be seen, SIRENeRF achieves a much better reconstruction, where the noise level is significantly decreased and the messy background is remarkably cleaned. The zoom-in figures allow us to observe the details of the scene. Compared to the ground truth shown in figure 5.6, we can see that SIRENeRF can reconstruct fine details such as the window grids that are made of two colors. As ref-NeRF finds it difficult to distinguish the light-blue windows and the dark-blue windows, SIRENeRF succeeds in depicting the grids and windows in different colors. By comparing the results from GFF-NeRF in the figure 5.9, we can clearly see that SIRENeRF also outperforms GFF-NeRF in representing these window grids. We can also see from the zoom-in figure on the right bottom that SIRENeRF achieves much better reconstruction of the edges and boundaries which are assumed high frequency. While ref-NeRF fails to deal with almost all the boundaries (e.g. boundaries between the facade and the terrace, boundaries between the grey ground and the green grassland), SIRENeRF clearly distinguishes these boundaries and produces a clean sketch of the shapes of the target objects. This proves that SIRENeRF is capable of encoding high-frequency content better than ref-NeRF. We can also find that SIRENeRF outperforms GFF-NeRF in representing these boundaries as GFF-NeRF fails to reconstruct the boundaries on the right side of the target building.

The synthesized views shown in figure5.14 also support our analysis. The zoom-in figures show that SIRENeRF remarkably outperforms ref-NeRF in learning and representing clearer and sharper details including the window grids and white car stop lines on the ground.

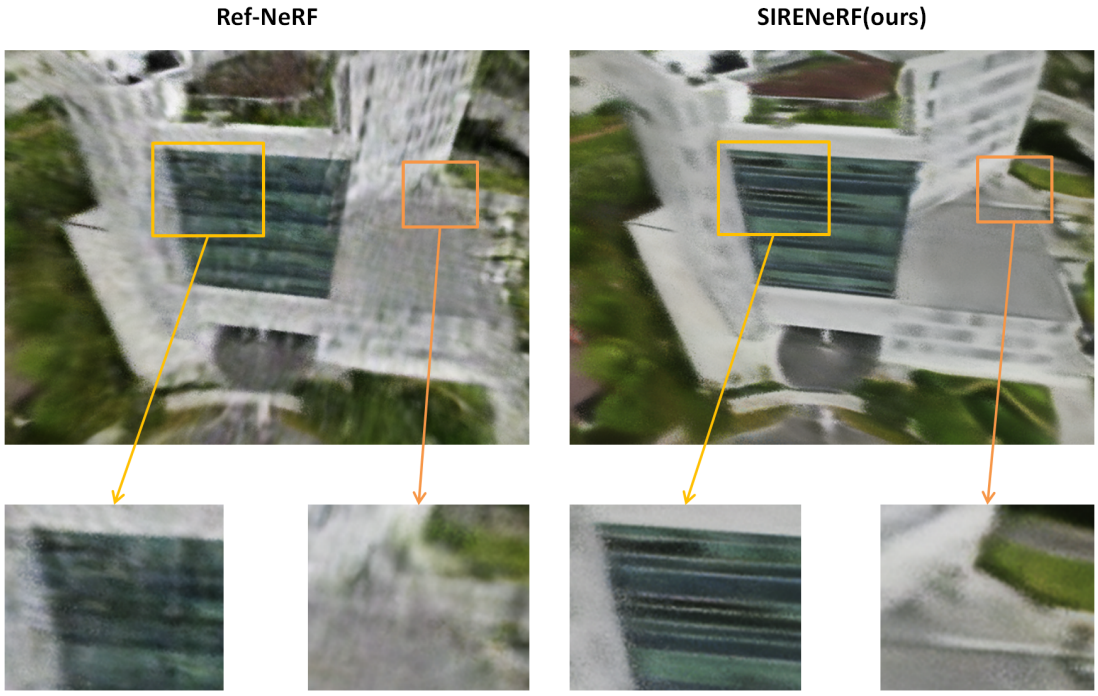

Figure 5.13: We compare the test image reconstruction results between SIRENeRF and ref-NeRF on the side-view scene.

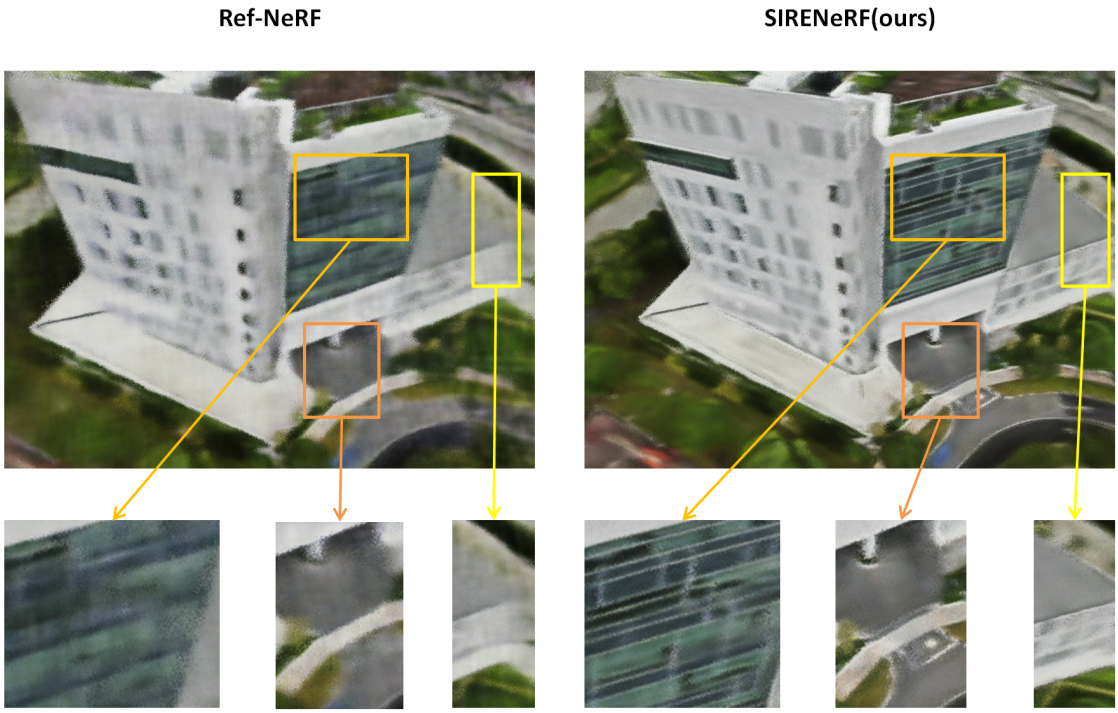

Figure 5.14: We compare the synthesized novel view from SIRENeRF and ref-NeRF on the side-view scene.

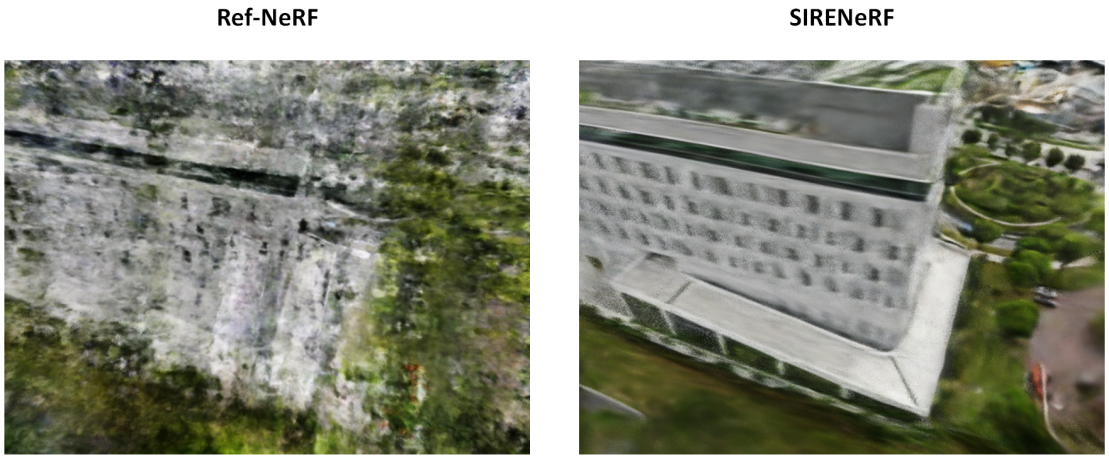

Figure 5.15: We compare the test image reconstruction results between SIRENeRF and ref-NeRF on the back-view scene.

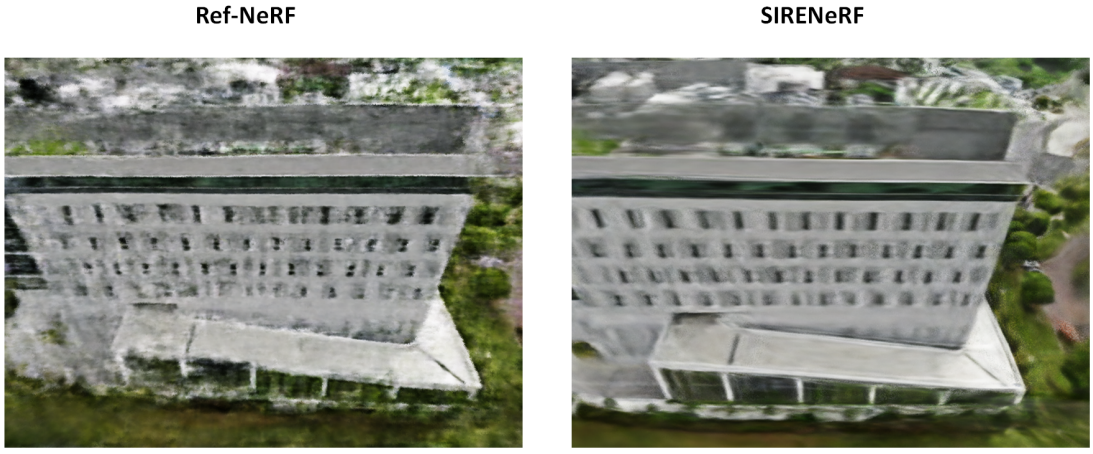

Figure 5.16: We compare the synthesized novel view from SIRENeRF and ref-NeRF on the back-view scene.

As we discussed in the section 5.3.1, the side-view scene dataset is less accurate and properly prepared than the front-view scene dataset, we may conjecture that SIRENeRF is capable of handling datasets that are of low quality better than ref-NeRF. This can be further proved by testing our model on the back-view scene dataset which is of even less quality. Figure 5.15 shows the scene reconstruction results from our model and ref-NeRF. Since the back-side scene dataset has fewer input views and a wider range of viewpoints than the side-view scene dataset, it is a failure for ref-NeRF to reconstruct the image. SIRENeRF, however, succeeds in recovering the scene. The novel view synthesis results shown in figure 5.16 also show that while ref-NeRF can only produce a seriously noisy image, SIRENeRF can synthesize a much better and clearer view.

The quantitative results are shown in table 5.2. The scores of PSNR, SSIM, LPIPS

	Front-view		
Models	PSNR ↑	SSIM ↑	LPIPS ↓
ref-NeRF	17.97	0.43	0.63
SIRENeRF	**19.29**	**0.47**	**0.54**
	Back-view		
Models	PSNR ↑	SSIM ↑	LPIPS ↓
ref-NeRF	12.76	0.17	0.63
SIRENeRF	**14.36**	**0.26**	**0.58**
	Side-view		
Models	PSNR ↑	SSIM ↑	LPIPS ↓
ref-NeRF	15.42	0.30	0.86
SIRENeRF	**17.80**	**0.36**	**0.76**

Table 5.2: Comparisons of quantitative metrics between SIRENeRF and ref-NeRF

agree with our qualitative observations that SIRENeRF outperforms ref-NeRF on all of the three datasets.

5.3.3 TARFs

5.3.3.1 Experimental Settings

We compare our proposed SIRENeRF and TARFs models with ref-NeRF. We assume known intrinsic in all the cases.

5.3.3.2 Implementation details

We implement our TARF framework following the settings from [32][26] with some modifications. Similarly, we train a 8-layer MLP with 256 hidden units in each layer to learn the volume density and features, and a 2-layer MLP with 256 hidden units in each layer to learn the RGB colors. We resize the images to 480×640 pixels and randomly sample 2048 pixel rays every iteration, each sampled at $N = 128$ coordinates. We use the Adam optimizer and train all the models for 200k iterations. For ref-NeRF, the learning rate begins at 1×10^{-3} and decays exponentially to 1×10^{-3}. For our proposed TARFs, the learning rate begins at 1×10^{-4} and decays exponentially to 5×10^{-5}. In Gaussian activation function, the initial parameter $a = 1$. In Laplacian activation function, the initial parameter $a = 1$. In SuperGaussian activation function, the initial parameter $a = 1$, $b = 1$. In Quadratic activation function, the initial parameter $a = 1$.

5.3.3.3 Evaluation Details

We evaluate the performance of SIRENeRF, TARFs, and ref-NeRF in terms of view synthesis quality for the scene reconstruction and their quantitative metrics. For simplicity, we only present the qualitative results from NeRF with trainable Quadratic activations (Q-TARF). The quantitative metrics used in the experiments are PSNR, SSIM, and LPIPS.

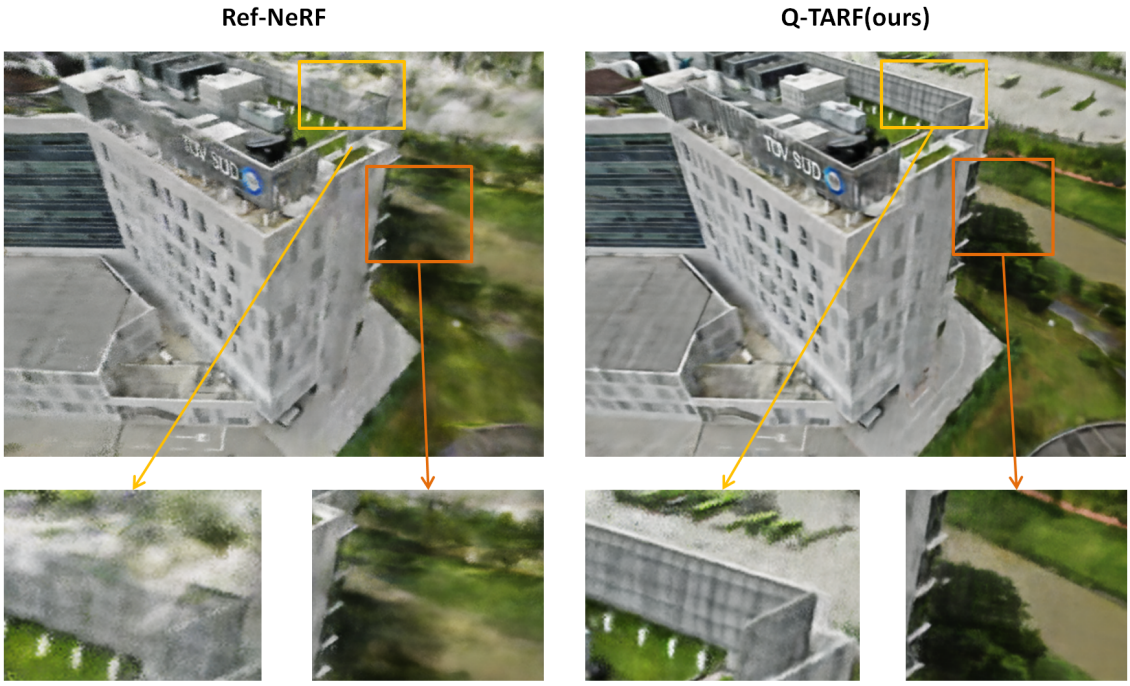

Figure 5.17: We compare the scene reconstruction results between Q-TARF and ref-NeRF on front-view scene

5.3.3.4 Results

We first test our Q-TARF model on the front-view scene dataset. The scene reconstruction results from Q-TARF and ref-NeRF shown in Figure 5.19 clearly indicate that Q-TARF remarkably outperforms the baseline. Compared to ref-NeRF, Q-TARF is not only able to reconstruct the unrestricted background accurately, but also represents better details of the target building. The zoom-in figures show that similar to GFF-NeRF and SIRENeRF, Q-TARF achieves a fine representation of the objects in the background. Moreover, Q-TARF outperforms GFF-NeRF and SIRENeRF in representing fine details. Compared to the reconstruction results from GFF-NeRF in figure 5.7 and from SIRENeRF in figure 5.11, the zoom-in figure on the right shows that Q-TARF can depict the branches and leaf on the canopy of the trees, while both GFF-NeRF and SIRENeRF can only depict a rough silhouette of the tree canopy. The zoom-in figure on the left also indicates that Q-TARF can produce a clear representation of the grids on the guardrails, while by SIRENeRF the guardrails are represented with more vagueness. This means that Q-TARF is more capable of handling high-frequency content in the signals than both the baseline and our previously proposed models. The novel view synthesis results shown in figure 5.18 also support our observation that Q-TARF has a better ability to represent fine details and the complex background. The zoom-in figures on the bottom show that Q-TARF produces a clearer representation of details.

We then test our model on the side-view scene dataset. Figure 5.19 shows the scene

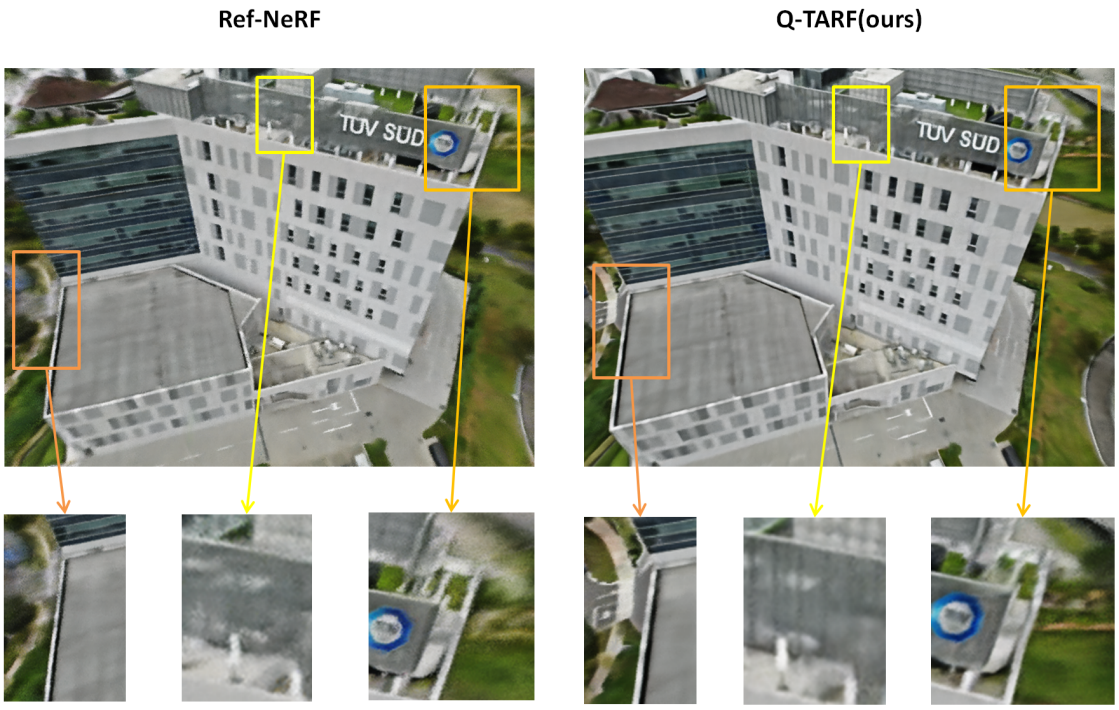

Figure 5.18: We compare the synthesized novel view from Q-TARF and ref-NeRF on the front-view scene

reconstruction results from our model and ref-NeRF. From the observation, we can find that Q-TARF easily outperforms ref-NeRF and is also competitive with our other proposed models. By significantly reducing the noise, Q-TARF produces a much cleaner reconstruction than the baseline. What is more, Q-TARF is capable of representing better details than others. Compared to the reconstruction results from SIRENeRF in figure 5.13, we can see that though Q-TARF is unable to reach as good noise-reducing effects as SIRENeRF, it depicts more accurate details. The zoom-in figure on the left-bottom shows that Q-TARF can reconstruct the grid of windows with accurate shapes and colors. As a comparison, ref-NeRF can only produce a vague and blurry representation without a clear distinction of the color changes, while SIRENeRF smooths the grids to stripes though it is better for reducing the noise. The zoom-in figure on the right-bottom also indicates that Q-TARF represents finer details than both ref-NeRF and SIRENeRF.

The synthesized views shown in figure 5.20 also indicate that Q-TARF achieves the best performance among all our proposed models and the baseline. As can be seen, Q-TARF can render a near photo-realistic novel view with fine resolution. The zoom-in figure on the left-bottom shows that Q-TARF can depict the windows with a clarity close to the ground truth. We can also see that the reconstruction of the windows by Q-TARF remarkably outperforms SIRENeRF and GFF-NeRF which are unable to represent the windows with such fine details. The zoom-in figure on the right-bottom also suggests that Q-TARF outperforms other models in depicting the edges and corners which are assumed high-frequency contents.

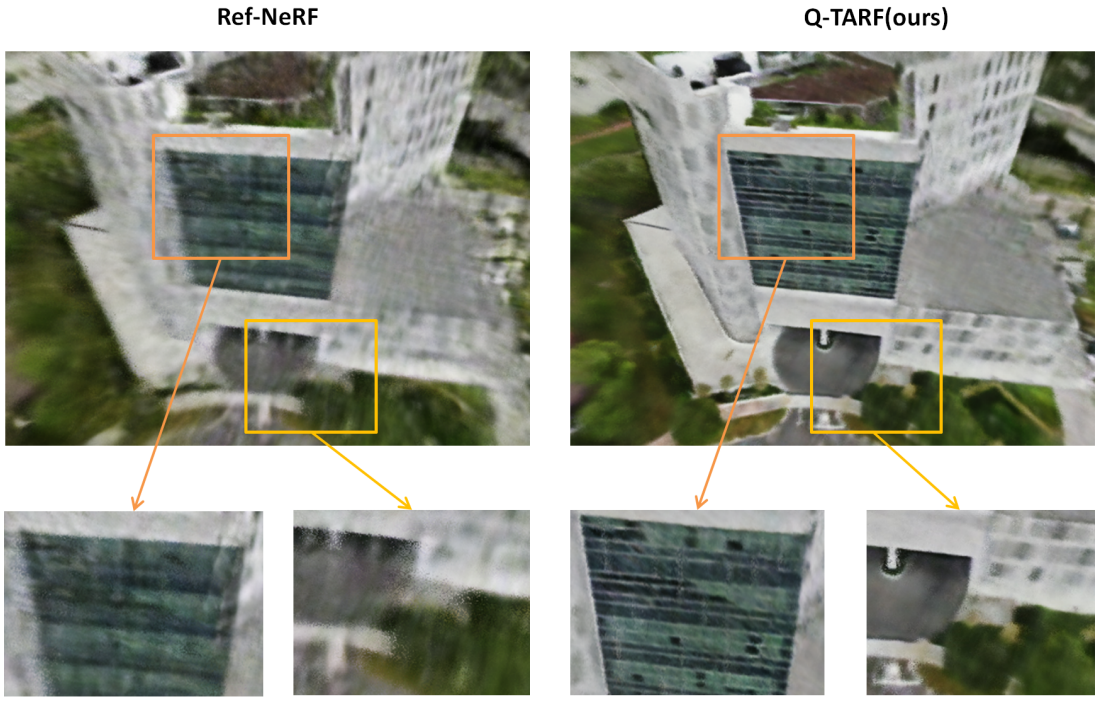

Figure 5.19: We compare the scene reconstruction results between Q-TARF and ref-NeRF on side-view scene.

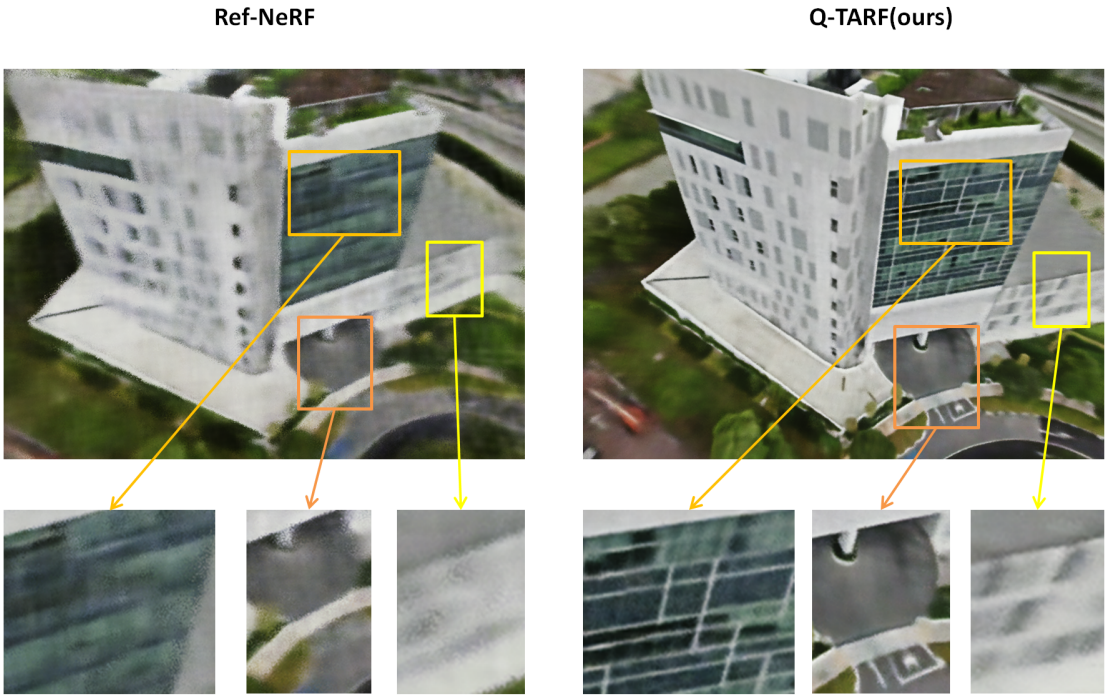

Figure 5.20: We compare the synthesized novel view from Q-TARF and ref-NeRF on the side-view scene.

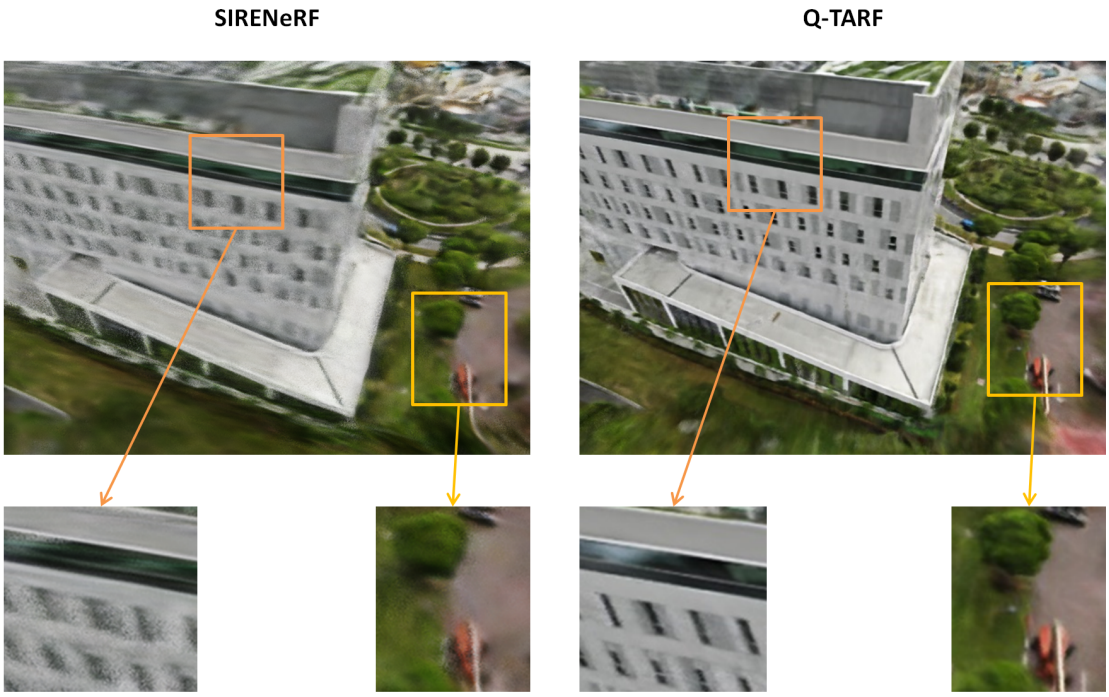

Figure 5.21: We compare the synthesized novel view from SIRENeRF and Q-TARF on the back-view scene.

Similar to SIRENeRF, Q-TARF can also recover the scene from the back-view scene dataset which is of lower quality. Figure 5.21 compares the scene reconstruction results from SIRENeRF and Q-TARF. As can be seen, Q-TARF achieves a reconstruction with higher clarity. Though both models are able to recover an understandable scene, Q-TARF performs better on reducing the noise and representing details. The zoom-in figures on the bottom show validate our observation that Q-TARF provides a clear representation of some complex details such as small windows on the facade and the reflection on the window. The zoom-in images also support that Q-TARF raises the quality of the image by achieving better noise reduction.

The synthesized views shown in figure 5.22 also show that Q-TARF outperforms SIRENeRF with higher clarity. The zoom-in figure on the left-bottom indicates that, while SIRENeRF can only render a rather blurry representation of the windows on the facade which is believed a difficulty for our models, Q-TARF manages to render a near photo-realistic representation with fine quality. The zoom-in image on the right-bottom also suggests that while SIRENeRF fails to learn and represent some complex structures such as the objects on the rooftop, Q-TARF succeeds in depicting distinct colors and shapes of these objects. However, we can also see that both models are not able to eliminate the huge noise at the edge of the background.

Quantitative results are given in table 5.3. In table 5.3, we illustrate the measurements of PSNR, SSIM, and LPIPS for all the TARF models and ref-NeRF. The results supports our observation that TARFs achieve better performance than the baseline and also outperform SIRENeRF.

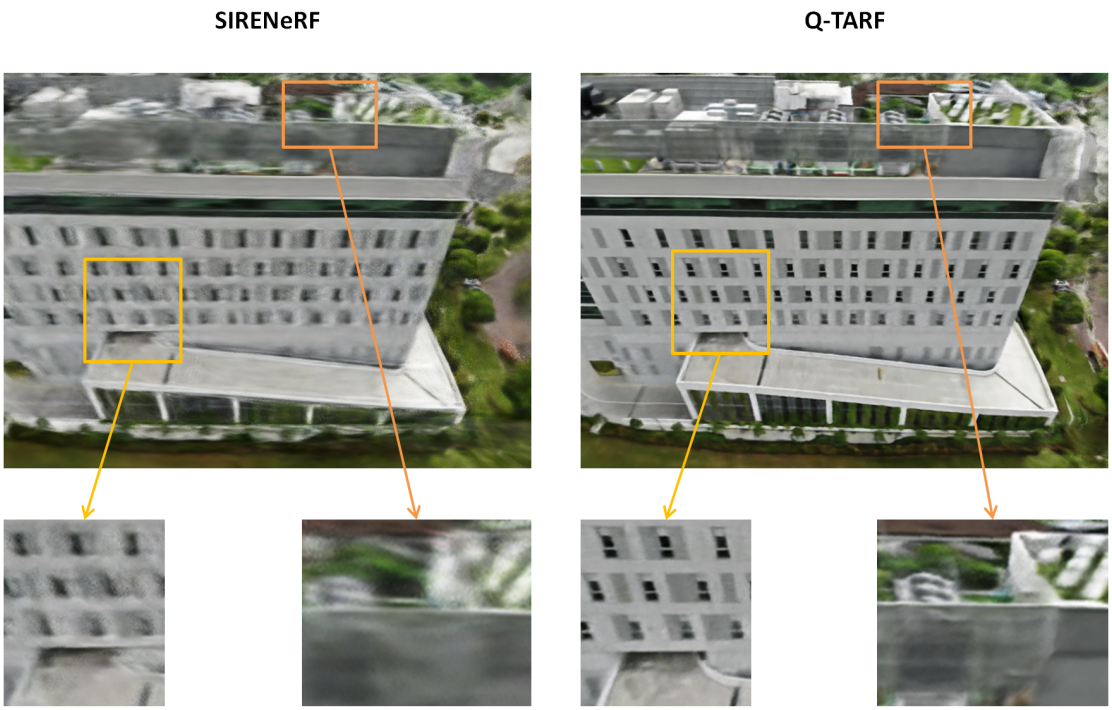

Figure 5.22: We compare the synthesized novel view from Q-TARF and Q-TARF on the back-view scene.

5.4 Comparison with ContextCapture

In this section, we briefly compare the qualitative reconstruction results from our best-performed model TARF with the reconstruction results from ContextCapture. As shown in figure5.23, our model can achieve a competitive performance with looser requirements on the dataset. Besides, our model can also render a photorealistic background while ContextCapture only extracts the main body of the target objects.

5.5 Discussion

According to the experiment results shown above, we can draw the following conclusions: Firstly, all of our attempts including Gaussian-distributed Fourier features, SIRENeRF, and Trainable-activated NeRF outperform the reference NeRF in terms of both qualitative measurements and quantitative measurements. Gaussian-distributed Fourier features expand the frequency mapping directions, which enables GFF-NeRF to recover the representation of the background and represent the details with higher fidelity. SIRENeRF and TARFs eliminate the use of positional encoding and build a self-contained architecture by leveraging parameterized activations. Parameterized activations enable SIRENeRF and TARFs to outperform reference NeRF in representing better details and significantly reducing the noise. SIRENeRF shows fine noise reduction and image smoothing effects, while TARFs show an ability to achieve a representation with high clarity. Besides, SIRENeRF and TARFs show higher robustness

Front-view scene			
Models	PSNR ↑	SSIM ↑	LPIPS ↓
Ref-NeRF	17.97	0.43	0.63
SIRENeRF	19.31	0.48	**0.53**
Laplacian TARF	19.86	0.46	0.60
Gaussian TARF	19.97	0.48	0.56
Quadratic TARF	**20.08**	**0.48**	0.58
SuperGaussian TARF	19.87	0.47	0.58
Back-view scene			
Models	PSNR ↑	SSIM ↑	LPIPS ↓
Ref-NeRF	12.76	0.17	0.63
SIRENeRF	14.36	0.26	0.58
Laplacian TARF	14.56	0.26	0.63
Gaussian TARF	14.60	0.26	0.58
Quadratic TARF	**14.70**	**0.27**	**0.57**
SuperGaussian TARF	14.57	0.27	0.61
Side-view scene			
Models	PSNR ↑	SSIM ↑	LPIPS ↓
Ref-NeRF	15.42	0.30	0.86
SIRENeRF	17.80	0.36	0.76
Laplacian TARF	17.97	0.34	0.78
Gaussian TARF	18.05	0.36	0.78
Quadratic TARF	**18.20**	**0.36**	**0.76**
SuperGaussian TARF	17.96	0.34	0.79

Table 5.3: Comparisons of quantitative metrics between SIRENeRF, TARFs and ref-NeRF

to the low-quality dataset in which the reference NeRF fails to learn the scene accurately. Both the qualitative observations and quantitative measurements indicate that, among all our proposed models, TARFs achieve the best performance which is mostly close to the ground truth. Besides, TARFs also stand out for their better flexibility since the proposed trainable scheme eliminates the need for exhaustively tuning the hyperparameters in the activation functions when applying SIRENeRF. We also compare the results from our best-performing model with those from ContextCapture. The comparison shows that our model can achieve competitive performance to ContextCapture with a simplier architecture and less requirements on the input data.

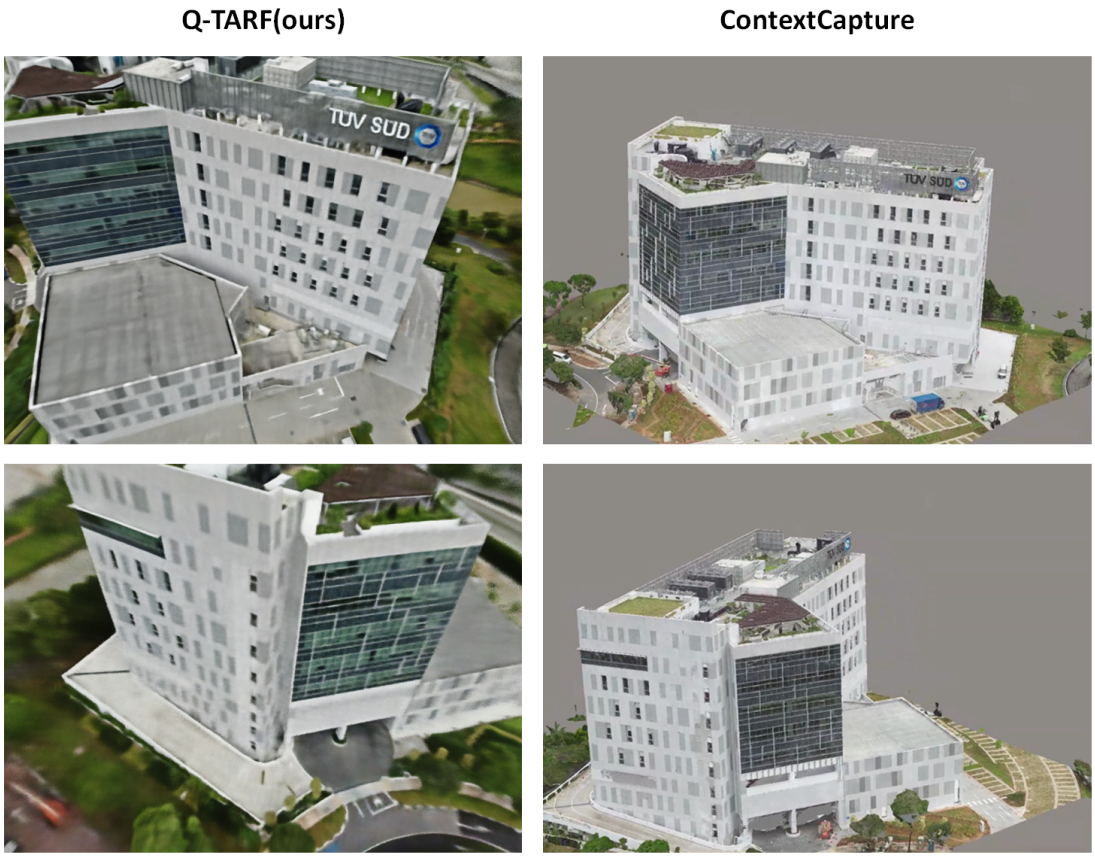

Figure 5.23: We compare the our reconstruction results with the results from ContextCapture

6 Conclusions

6.1 Contributions

In the research report, we address all the research objectives listed in section 1.2. The initial goal is to synthesize novel views of a large object, the facade of our target building in Singapore, with only RGB images as the inputs. To solve this problem, we provide an effective guideline that can serve as an end-to-end approach to reconstructing the scene and achieving photorealistic rendering from RGB images taken by drones. In this guideline, we first use COLMAP, an incremental SfM pipeline to estimate the camera poses. With the posed images, we then use NeRF, which stands out as a proper method among other point cloud or mesh-based traditional methods for its simplicity and outstanding performance, as the base method to reconstruct our target scene. NeRF encodes a 3D scene as a continuous function that turns the 5D coordinate inputs (3D location and 2D viewing direction) into color and volume density and uses a differentiable volume rendering technique to composite the volume density and the color into an image. This continuous representation is optimized via a fully-connected network to minimize the loss between the composited image and the ground truth. We demonstrate that representing scenes as neural radiance fields can produce rendering results that are competitive with the results from the point cloud-based method used in the commercial software ContextCapture.

We also take a step forward and address the deficiencies of NeRF in reconstructing high-frequency content by researching the coordinate MLP and how inputs are mapped in NeRF. We demonstrate that coordinate MLPs cannot handle the high-frequency content in the input images because they are biased towards learning low-frequency content. NeRF solves these problems by using positional encoding to map the inputs from low-dimensional space to high-dimensional space. We further find that this is not the best strategy for mapping, and we make three attempts to improve the performance of NeRF. We first use Gaussian-distributed Fourier features to replace the original positional encoding. Different from positional encoding that assumes the variation of pixels only fluctuates rapidly along the X and Y axes, Gaussian-distributed Fourier features enable a mapping along other directions (different angles between X-axis and Y-axis). We empirically show that this mapping outperforms positional encoding both visually and numerically. Next, we build a self-contained coordinate MLP architecture to achieve the input mapping within the network. We first study the memorization and generalization of a coordinate MLP. We find that to encode the high-frequency content in the signals, the activation functions in the MLP are supposed to meet two requirements. First, the activation functions should be parameterized so that the magnitude of its first-order derivative can be controlled via the parameters. This is because the upper-bound on the Lipschitz constant of the angle variation in a local interval can be

increased by increasing the local Lipschitz constant of the activation function, while the local Lipschitz constant of the activation function correlates to the magnitude of its first-order derivative. Therefore, a higher magnitude of the first-order derivative enables a better capacity to encode signals with large fluctuations. The first requirement is however not necessarily sufficient for ensuring a good performance of encoding signals with extremely varying fluctuations across different intervals. Therefore, the second requirement that should be obeyed is that the activation function should also have non-negligible second-order derivatives to obtain varying Lipschitz smoothness. The reference NeRF uses ReLU as the activation function in MLPs, while ReLU does not meet both requirements. Therefore, we use a set of other functions as activation functions. Our first attempt is to use sine functions to replace the ReLUs in the coordinate MLP. The choice is based on the fact that the periodicity is inherited along the derivatives of the sine function that the derivative of sine can be regarded as a phase-shifted sine. Besides, the magnitude of the derivatives is controlled over the parameter in the sine. We empirically present that this carefully-chosen activation function can enable a better fitting of the high-frequency content in the images, and a NeRF model combined with sine activations (referred to as SIRENeRF in the report) outperforms the reference NeRF in recovering the unbounded background and depicting the target building with sharper details. We also illustrate that SIRENeRF has two major limitations. First, sine activations are very sensitive to the initialization scheme. Without strictly following a carefully-designed initialization scheme demonstrated in section 4.3.2.1, the view synthesis performance drops significantly. Second, a sine activation, which is parameterized by a fixed coefficient w, brings more hyperparameters to the networks. This adds cumbersome hyperparameter tuning work in complex tasks where a deep fully-connected network is required. We empirically find that there are no general settings for these hyperparameters when applying SIREN-based NeRF with different input data. Therefore, the flexibility of sine activations is significantly limited in real-world applications where the inputs vary all the time. Moreover, we find that a fixed parameter in the activation function limits the ability to control the magnitude of its first-order derivative according to the varying content in the signals. A high magnitude though enables better fitting of the high-frequency content, lessens the ability to encode less-fluctuated content. We address these issues by proposing a trainable scheme for parameterized activations, which omits the effort to manually adjust the parameters in the activations by automatically optimizing these parameters along with other parameters in the deep network. We show that our model, referred to as TARF, achieves better performance than the reference NeRF and SIRENeRF in reducing the noise and representing better details. We demonstrate that TARFs are also more capable of recovering unbounded scenes with imperfect camera poses.

Besides, we also present an analysis of the raw data provided by TUV SUD and the data preparation strategies in the report. We demonstrate that there are two strategies for synthesizing the target building. The first strategy is to synthesize a 360-degree view, while the second strategy is to divide the scene into several separated forward-facing views. We show that an important difference between implementing these two strategies is the coordinate transformation. When rendering forward-facing scenes, the camera coordinates are transformed to a Normalized device coordinate (NDC) space

where the x, y, and z coordinates of each vertex are ranged between -1.0 and 1.0, and coordinates outside this range will not be visible. This space is convenient because it preserves parallel lines while converting the z axis (camera axis) to be linear in disparity. For 360-degree spherical renderings, this NDC transformation should not be operated. We then use COLMAP to estimate the camera poses, which are necessary for implementing NeRF.

6.2 Future work

While our models can produce photorealistic reconstruction and rendering of our target building with a sequence of sampled RGB images as the input and outperform the baseline model NeRF, there are still several limitations. Firstly, the accuracy of the camera pose estimation remains a huge problem in real-life applications. Though COLAMP stands out as an effective and convenient method to estimate the camera poses from raw RGB images, in the experiments we find that the robustness of COLMAP is correlated to the camera motions. A forward-facing capture assumes that the camera moves within the xy plane and the vertical height keeps unchanged along the whole trajectory. However, this assumption cannot be always held in the real life. A study [57] shows that COLMAP starts breaking when trajectory perturbations exceed a certain level. Though our best-performed models can cope with the camera calibration errors better than NeRF, it is still very sensitive to the accuracy of the pose estimation. The experiment results show that covering a similar range of viewpoints, sparser views lead to a less accurate estimation of the camera poses, and the reconstruction performance degrades accordingly. This indicates that our models still require an adequate amount of images as the input data and still cannot achieve a fine rendering performance with only a few shots. Besides considering a better pose estimation method than COLMAP, many recent studies are working towards producing a scene representation from one or just a few input images. Some methods condition NeRF by utilizing convolutional neural networks to extract the image features. For a query point along a target camera ray with view direction, a corresponding image feature is extracted from the feature volume via projection and interpolation. These features are then fed to the MLP network as a residual. This architecture can enable a few-view novel-view synthesis and train NeRF on multi-view data without additional supervision. Other studies [19] leverage auxiliary semantic consistency loss to achieve a 3D neural scene representation from only a few images. These methods use a pre-trained visual encoder called CLIP, a Vision Transformer trained on millions of diverse online photographs with natural language supervision, to extract the semantics from the images. This semantic loss allows for improving the perceptual quality of few-shot view synthesis and enables rendering novel views with as few as one observed image when pre-trained on a multi-view dataset. These studies point out directions for us to improve the performance of our models in situations where the input data are sparse.

Another limitation is that our models, similar to most of the NeRF-based approaches, downsample the input images before feeding them into the neural networks out of the training speed and storage. This limits the use of our models in many real-life applications. A possible way to tackle this problem is to combine NeRF with super-

resolution techniques. A study [54] indicates that 3D consistency can help NeRF produce better representations where an observed pixel absorbs information from nearby views. To this end, an extension of NeRF called NeRF-SR[54] has been proposed that leverage this property by applying a super-sampling strategy that shoots multiple rays at each image pixel to enforce multi-view constraint at a sub-pixel level. NeRF-SR then further boosts the performance of super-sampling by a refinement network that leverages the estimated depth to complete details from related patches on only one high-resolution reference image. This technique paves a road for us to further modify our models and let them able to improve the resolutions of the synthesized images.

Furthermore, another factor that hampers the use of our models in real-life applications is the lengthy training speed. On the one hand, the training process in our experiments is restricted by the limited GPU resources (all our experiments are run on Google Colab with an NVIDIA K80 GPU). On the other hand, optimizing the hundreds of millions of parameters in a deep fully-connected network in NeRF is no doubt time-consuming. As a result, training the reference NeRF model often requires 12-15 hours (depending on the size of the input data), while training our models requires less time but still around 10 hours for one single scene. This limitation not only omits the possibility of a real-time application but also restricts the efficiency of modeling and inspecting buildings using our models. To speed up the training time, an effective approach presented in [50] replaces the coordinate MLP with a voxel grid. Different NeRF, this approach uses a dense voxel grid to model the 3D geometry (volume density) and a feature voxel grid to model the complex view-dependent appearance (color). The key ideas that enable a fast convergence include introducing a post-activation interpolation on voxel density that allows for producing sharp surfaces in lower grid resolution and modifying the voxel density optimization process to cope with the suboptimal geometry solutions. This approach is believed to significantly reduce the GPU memory consumption and cut the training time from a few hours to 15 minutes. Another compelling method proposed by NVIDIA named Instant Neural Graphics Primitives (Instant NGP) [33] has achieved extraordinary training speed that can train a NeRF model in under 5 seconds. The key contribution of this approach is a proposed multi-resolution hash encoding that is used to map the input coordinate. This study demonstrates that the positional encoding, along with its further extensions including our randomly oriented Fourier features and level-of-detail filtering [2], can be generalized into a frequency encoding family. Though the frequency encoding plays a key role in extracting high approximation quality from compact models, it complicates the training process and usually requires a large and deep MLP to learn complex tasks such as 3D novel view synthesis, which limits the performance on GPUs where control flow and pointer chasing are expensive. The multi-resolution hash encoding, in contrast, allows using a smaller network without sacrificing quality, where the small network is augmented by a multi-resolution hash table of trainable feature vectors whose values are optimized through stochastic gradient descent. The multi-resolution structure allows the network to disambiguate hash collisions, making for a simple architecture that is trivial to parallelize on modern GPUs. The parallelism is leveraged by implementing the whole system using fully-fused CUDA kernels with a focus on minimizing wasted bandwidth and compute operations. This work paves the road for using a much smaller

network to train the 3D scene and reduces the training time by more than a thousand times.

In summary, all the research objectives have been properly achieved in our research, and in the end, We outline the drawbacks and limitations of our methods and suggest possible future work that can be pursued for improvement.

Bibliography

[1] Alaa E Abdel-Hakim and Aly A Farag. Csift: A sift descriptor with color invariant characteristics. In *2006 IEEE computer society conference on computer vision and pattern recognition (CVPR'06)*, volume 2, pages 1978–1983. Ieee, 2006.

[2] Jonathan T Barron, Ben Mildenhall, Matthew Tancik, Peter Hedman, Ricardo Martin-Brualla, and Pratul P Srinivasan. Mip-nerf: A multiscale representation for anti-aliasing neural radiance fields. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5855–5864, 2021.

[3] Ronen Basri, Meirav Galun, Amnon Geifman, David Jacobs, Yoni Kasten, and Shira Kritchman. Frequency bias in neural networks for input of non-uniform density. In *International Conference on Machine Learning*, pages 685–694. PMLR, 2020.

[4] Derek Bradley, Alan Brunton, Mark Fiala, and Gerhard Roth. Image-based navigation in real environments using panoramas. In *IEEE International Workshop on Haptic Audio Visual Environments and their Applications*, pages 3–pp. IEEE, 2005.

[5] Chris Buehler, Michael Bosse, Leonard McMillan, Steven Gortler, and Michael Cohen. Unstructured lumigraph rendering. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 425–432, 2001.

[6] Rodrigo Ortiz Cayon, Abdelaziz Djelouah, and George Drettakis. A bayesian approach for selective image-based rendering using superpixels. In *2015 International Conference on 3D Vision*, pages 469–477. IEEE, 2015.

[7] Eric R Chan, Marco Monteiro, Petr Kellnhofer, Jiajun Wu, and Gordon Wetzstein. pi-gan: Periodic implicit generative adversarial networks for 3d-aware image synthesis. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 5799–5809, 2021.

[8] SC Chan, Heung-Yeung Shum, and King-To Ng. Image-based rendering and synthesis. *IEEE Signal Processing Magazine*, 24(6):22–33, 2007.

[9] Gaurav Chaurasia, Sylvain Duchene, Olga Sorkine-Hornung, and George Drettakis. Depth synthesis and local warps for plausible image-based navigation. *ACM Transactions on Graphics (TOG)*, 32(3):1–12, 2013.

[10] Anpei Chen, Zexiang Xu, Fuqiang Zhao, Xiaoshuai Zhang, Fanbo Xiang, Jingyi Yu, and Hao Su. Mvsnerf: Fast generalizable radiance field reconstruction from multi-view stereo. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 14124–14133, 2021.

[11] Zhiqin Chen and Hao Zhang. Learning implicit fields for generative shape modeling. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5939–5948, 2019.

[12] Kangle Deng, Andrew Liu, Jun-Yan Zhu, and Deva Ramanan. Depth-supervised nerf: Fewer views and faster training for free. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12882–12891, 2022.

[13] Sarah Fachada, Daniele Bonatto, Mehrdad Teratani, and Gauthier Lafruit. Polynomial image-based rendering for non-lambertian objects. In *2021 International Conference on Visual Communications and Image Processing (VCIP)*, pages 01–05. IEEE, 2021.

[14] Martin A Fischler and Robert C Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.

[15] John Flynn, Keith Snavely, Ivan Neulander, and James Philbin. Deepstereo: learning to predict new views from real world imagery, March 13 2018. US Patent 9,916,679.

[16] Peter Hedman and Johannes Kopf. Instant 3d photography. *ACM Transactions on Graphics (TOG)*, 37(4):1–12, 2018.

[17] Peter Hedman, Julien Philip, True Price, Jan-Michael Frahm, George Drettakis, and Gabriel Brostow. Deep blending for free-viewpoint image-based rendering. *ACM Transactions on Graphics (TOG)*, 37(6):1–15, 2018.

[18] Peter Hedman, Tobias Ritschel, George Drettakis, and Gabriel Brostow. Scalable inside-out image-based rendering. *ACM Transactions on Graphics (TOG)*, 35(6):1–11, 2016.

[19] Ajay Jain, Matthew Tancik, and Pieter Abbeel. Putting nerf on a diet: Semantically consistent few-shot view synthesis. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5885–5894, 2021.

[20] Yoonwoo Jeong, Seokjun Ahn, Christopher Choy, Anima Anandkumar, Minsu Cho, and Jaesik Park. Self-calibrating neural radiance fields. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5846–5854, 2021.

[21] Nima Khademi Kalantari, Ting-Chun Wang, and Ravi Ramamoorthi. Learning-based view synthesis for light field cameras. *ACM Transactions on Graphics (TOG)*, 35(6):1–10, 2016.

[22] Yan Ke and Rahul Sukthankar. Pca-sift: A more distinctive representation for local image descriptors. In *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004.*, volume 2, pages II–II. IEEE, 2004.

[23] Reinhard Koch, Benno Heigl, and Marc Pollefeys. Image-based rendering from uncalibrated lightfields with scalable geometry. In *Multi-Image Analysis*, pages 51–66. Springer, 2001.

[24] Marc Levoy. Efficient ray tracing of volume data. *ACM Transactions on Graphics (TOG)*, 9(3):245–261, 1990.

[25] Marc Levoy and Pat Hanrahan. Light field rendering. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 31–42, 1996.

[26] Chen-Hsuan Lin, Wei-Chiu Ma, Antonio Torralba, and Simon Lucey. Barf: Bundle-adjusting neural radiance fields. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5741–5751, 2021.

[27] David G Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004.

[28] David Marimon and Touradj Ebrahimi. Orientation histogram-based matching for region tracking. In *Eighth International Workshop on Image Analysis for Multimedia Interactive Services (WIAMIS'07)*, pages 8–8. IEEE, 2007.

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

[30] Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy networks: Learning 3d reconstruction in function space. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4460–4470, 2019.

[31] Ben Mildenhall, Pratul P Srinivasan, Rodrigo Ortiz-Cayon, Nima Khademi Kalantari, Ravi Ramamoorthi, Ren Ng, and Abhishek Kar. Local light field fusion: Practical view synthesis with prescriptive sampling guidelines. *ACM Transactions on Graphics (TOG)*, 38(4):1–14, 2019.

[32] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. In *European conference on computer vision*, pages 405–421. Springer, 2020.

[33] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *arXiv preprint arXiv:2201.05989*, 2022.

[34] Raul Mur-Artal, Jose Maria Martinez Montiel, and Juan D Tardos. Orb-slam: a versatile and accurate monocular slam system. *IEEE transactions on robotics*, 31(5):1147–1163, 2015.

[35] Pauline C Ng and Steven Henikoff. Sift: Predicting amino acid changes that affect protein function. *Nucleic acids research*, 31(13):3812–3814, 2003.

[36] Michael Niemeyer and Andreas Geiger. Giraffe: Representing scenes as compositional generative neural feature fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11453–11464, 2021.

[37] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 165–174, 2019.

[38] Nasim Rahaman, Aristide Baratin, Devansh Arpit, Felix Draxler, Min Lin, Fred Hamprecht, Yoshua Bengio, and Aaron Courville. On the spectral bias of neural networks. In *International Conference on Machine Learning*, pages 5301–5310. PMLR, 2019.

[39] Sameera Ramasinghe and Simon Lucey. Beyond periodicity: Towards a unifying framework for activations in coordinate-mlps. *arXiv preprint arXiv:2111.15135*, 2021.

[40] Sameera Ramasinghe and Simon Lucey. Learning positional embeddings for coordinate-mlps. *arXiv preprint arXiv:2112.11577*, 2021.

[41] Sameera Ramasinghe, Lachlan MacDonald, and Simon Lucey. On regularizing coordinate-mlps. *arXiv preprint arXiv:2202.00790*, 2022.

[42] Johannes L Schönberger, Alexander C Berg, and Jan-Michael Frahm. Efficient two-view geometry classification. In *German Conference on Pattern Recognition*, pages 53–64. Springer, 2015.

[43] Johannes L Schonberger and Jan-Michael Frahm. Structure-from-motion revisited. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4104–4113, 2016.

[44] Vitor Sequeira, Erik Wolfart, Emanuele Bovisio, Ester Biotti, and Joao GM Goncalves. Hybrid 3d reconstruction and image-based rendering techniques for reality modeling. In *Videometrics and Optical Methods for 3D Shape Measurement*, volume 4309, pages 126–136. SPIE, 2000.

[45] Heung-Yeung Shum and Li-Wei He. Rendering with concentric mosaics. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, pages 299–306, 1999.

[46] Heung-Yeung Shum, King-To Ng, and Shing-Chow Chan. Virtual reality using the concentric mosaic: Construction, rendering and data compression. In *Proceedings 2000 International Conference on Image Processing (Cat. No. 00CH37101)*, volume 3, pages 644–647. IEEE, 2000.

[47] Sudipta N Sinha, Johannes Kopf, Michael Goesele, Daniel Scharstein, and Richard Szeliski. Image-based rendering for scenes with reflections. *ACM Transactions on Graphics (TOG)*, 31(4):1–10, 2012.

[48] Vincent Sitzmann, Julien Martel, Alexander Bergman, David Lindell, and Gordon Wetzstein. Implicit neural representations with periodic activation functions. *Advances in Neural Information Processing Systems*, 33:7462–7473, 2020.

[49] Pratul P Srinivasan, Tongzhou Wang, Ashwin Sreelal, Ravi Ramamoorthi, and Ren Ng. Learning to synthesize a 4d rgbd light field from a single image. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2243–2251, 2017.

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

[51] Richard Szeliski and Heung-Yeung Shum. Creating full view panoramic image mosaics and environment maps. In *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, pages 251–258, 1997.

[52] Matthew Tancik, Pratul Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan Barron, and Ren Ng. Fourier features let networks learn high frequency functions in low dimensional domains. *Advances in Neural Information Processing Systems*, 33:7537–7547, 2020.

[53] Justus Thies, Michael Zollhöfer, and Matthias Nießner. Deferred neural rendering: Image synthesis using neural textures. *ACM Transactions on Graphics (TOG)*, 38(4):1–12, 2019.

[54] Chen Wang, Xian Wu, Yuan-Chen Guo, Song-Hai Zhang, Yu-Wing Tai, and Shi-Min Hu. Nerf-sr: High-quality neural radiance fields using supersampling. *arXiv*, 2021.

[55] Qianqian Wang, Zhicheng Wang, Kyle Genova, Pratul P Srinivasan, Howard Zhou, Jonathan T Barron, Ricardo Martin-Brualla, Noah Snavely, and Thomas Funkhouser. Ibrnet: Learning multi-view image-based rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4690–4699, 2021.

[56] Yunlong Wang, Fei Liu, Zilei Wang, Guangqi Hou, Zhenan Sun, and Tieniu Tan. End-to-end view synthesis for light field imaging with pseudo 4dcnn. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 333–348, 2018.

[57] Zirui Wang, Shangzhe Wu, Weidi Xie, Min Chen, and Victor Adrian Prisacariu. Nerf–: Neural radiance fields without known camera parameters. *arXiv preprint arXiv:2102.07064*, 2021.

[58] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 586–595, 2018.

[59] Tinghui Zhou, Richard Tucker, John Flynn, Graham Fyffe, and Noah Snavely. Stereo magnification: Learning view synthesis using multiplane images. *arXiv preprint arXiv:1805.09817*, 2018.

[60] Stefan Zickler and Alexei Efros. Detection of multiple deformable objects using pca-sift. In *AAAI*, pages 1127–1133, 2007.