Title

Advancing Neural Radiance Fields through Self-Supervised NeRF Image Selector (SNIS)

Author

Kim, Nahyeon

Publication Date

2024

Peer reviewed|Thesis/dissertation

UNIVERSITY OF CALIFORNIA

Los Angeles

Advancing Neural Radiance Fields through Self-Supervised NeRF Image Selector (SNIS)

A thesis submitted in partial satisfaction
of the requirements for the degree Master of Science
in Mechanical Engineering

by

Nahyeon Kim

2024

ABSTRACT OF THE THESIS

Advancing Neural Radiance Fields through Self-Supervised NeRF Image Selector (SNIS)

by

Nahyeon Kim

Master of Science in Mechanical Engineering

University of California, Los Angeles, 2024

Professor Mohammad Khalid Jawed, Chair

Neural Radiance Fields (NeRF) have recently emerged as a promising research area due to their remarkable photorealistic rendering capabilities and diverse applications. By training NeRF network with various viewpoints and their corresponding images, we can generate new views of a scene. This approach, however, is highly dependent on the quality and composition of the training images. Effective learning in NeRF can be achieved by selecting images with high coverage rather than those having many overlaps. To date, there has been a lack of research focused on the selection of images for NeRF training. To address this gap, this paper proposes an innovative image selector designed to select optimal image viewpoints to enhance NeRF training. We created a custom dataset by creating a virtual environment in Unity, allowing for the flexible positioning of cameras to capture images. We introduce two methods to predict optimal image viewpoints: the first employs reinforcement learning, while the second utilizes a Self-supervised NeRF Image Selector (SNIS) based on a simplified deep neural network architecture. SNIS is trained using our novel pseudo-labels, which function analogously to the reward mechanism in reinforcement learning frameworks. Experimental results demonstrate that our selector significantly outperforms the random selection of images, providing stable and superior performance in NeRF training.

The thesis of Nahyeon Kim is approved.

Sriram Narasimhan

Veronica Santos

Mohammad Khalid Jawed, Committee Chair

University of California, Los Angeles

2024

Contents

List of Figures

List of Tables

Acknowledgments

Foremost gratitude is extended to Professor Khalid Jawed for his unwavering support, insightful guidance, and invaluable encouragement. His expertise and dedication have been instrumental in the successful completion of this thesis, and working under his mentorship has been a profound privilege.

Deep appreciation is also conveyed to the UCLA Structures-Computer Interaction (SCI) Lab. The collaborative environment and intellectual stimulation provided by the lab have greatly enriched this research experience. Special thanks are due to the lab mates for their camaraderie, constructive feedback, and the many enlightening discussions shared.

Also, I would like to express my profound appreciation to the thesis committee members, Professor Sriram Narasimhan and Professor Veronica Santos, for their time and valuable guidance. Further acknowledgment is made of the financial and institutional support provided by UCLA. This support has been essential for the execution of this research.

Personal gratitude is extended to family and friends for their constant encouragement and understanding. Their unwavering belief has been a source of great motivation and strength throughout this journey.

Finally, thanks are due to all who have directly or indirectly contributed to this work. Your support and contributions have been invaluable, and are deeply appreciated.

1 Introduction

1.1 Background of the Study

The integration of artificial intelligence (AI) and robotics has attracted growing attention due to its vast array of applications and its potential to execute highly sophisticated tasks. AI and robotics are revolutionizing various industries, including agriculture, healthcare, manufacturing, and logistics. One common method for training robots involves teaching them to perceive real-world environments and learn how to perform specific tasks. For instance, an agricultural robot designed to dispense pesticides on weeds would traditionally be brought to every targeted farm and allowed to learn by interacting with each environment. The robot would learn to recognize and differentiate between crops and weeds, enabling it to perform its task efficiently. However, this method comes with significant spatial constraints, as it requires the physical relocation of the robot to various real-world environments. This approach is not only time-consuming but also costly, as it involves substantial logistical efforts and expenses to transport the robot to different locations. Moreover, the variability in environmental conditions across different farms can pose additional challenges, as the robot may need to adapt to a wide range of scenarios, further complicating the training process.

To address these spatial and temporal constraints, a viable solution is to train the robot in a virtual environment that closely resembles real-world conditions. Virtual training environments offer numerous advantages, including the flexibility to modify the environment according to specific training needs and the ability to use a more diverse set of training datasets. By creating a virtual environment, robots can interact with a simulated setting that mimics the real world, allowing them to learn tasks as if they were in an actual environment. This approach not only saves time and reduces costs but also enables parallel training, where multiple robots can be trained simultaneously, leading to a significant reduction in overall training time. To make virtual learning practically achievable, it is essential to construct a virtual setting that accurately replicates the target environment and to establish an infrastructure that allows for seamless interaction

between the robot and the virtual environment. One effective method for constructing such an environment is through 3D scene rendering, which involves creating a 3D model scene from images. This technique provides a straightforward and efficient way to develop a detailed virtual representation of the target environment without the need for complex calculations or extensive manual modeling.

In this context, Neural Radiance Fields (NeRF) have emerged as a powerful tool for 3D scene rendering. NeRF is a novel approach that leverages neural networks to generate high-fidelity visual representations of real-world environments from images. By inputting a set of images into the NeRF model, it can reconstruct a detailed 3D scene that captures the intricate geometry and lighting conditions of the original environment. This capability makes NeRF particularly suitable for creating virtual training environments that are both accurate and realistic. The potential applications of NeRF extend beyond agriculture to various fields where precise 3D modeling is required. For example, in healthcare, NeRF can be used to create virtual simulations of medical procedures, allowing practitioners to train in a controlled and risk-free environment. In manufacturing, NeRF can facilitate the development of virtual prototypes, enabling engineers to test and refine designs before physical production. Additionally, in the realm of autonomous vehicles, NeRF can generate realistic driving scenarios for training self-driving algorithms, thereby enhancing their performance and safety.

Given the broad implications and benefits of virtual training environments, this research aims to explore and optimize the use of NeRF to create such settings. By systematically analyzing and improving the NeRF training process, we seek to enhance its efficiency and effectiveness, ultimately contributing to the advancement of AI and robotics training methodologies. This study will focus on developing algorithms that can predict optimal image sets for NeRF training, ensuring that the resulting images are of the highest quality and fidelity. In summary, the integration of AI and robotics holds immense potential for transforming various industries. However, traditional training methods are often constrained by spatial and temporal limitations. Training the vision of AI in virtual environments, enabled by advanced 3D scene rendering techniques like NeRF, offer

a promising solution to these challenges. By leveraging the capabilities of NeRF, we can create realistic and detailed virtual visual settings that facilitate efficient and effective robot training, paving the way for more sophisticated and practical applications of AI and robotics.

1.2 Problem Statement

The performance of Neural Radiance Fields (NeRF) models can vary significantly depending on the specific images used for training, as these models require sufficient visual information from diverse viewpoints to accurately learn the 3D structure of a scene and its complex lighting conditions. If the images are biased towards certain viewpoints or angles, or if they fail to capture critical parts of the scene adequately, the model may miss essential information during the learning process. Consequently, the resulting images may be inaccurate or lack detail, undermining the effectiveness of the NeRF approach.

Conversely, if the images are optimally captured from various angles and viewpoints, ensuring even coverage or focused attention on complex areas, the NeRF model can comprehensively understand the entire scene and produce accurate scenes. Therefore, to maximize NeRF's performance, it is crucial to carefully consider not only the number of images but also their quality and distribution. Optimal image selection is essential for the model to gain a complete and accurate understanding of the scene.

Achieving this requires research into image selection methods that aid the model in accurately understanding and reconstructing the scene. This research focuses on identifying optimal image selection strategies that enable the model to understand the scene most efficiently. At the same time, it is crucial to explore methods that yield efficient performance without excessively consuming computational resources, even when employing simpler models. In other words, there is a need for image selection methods that can produce optimal learning outcomes with minimal complexity and computational cost, thereby enabling NeRF models to perform high-quality 3D rendering with fewer resources.

In this paper, we implement and evaluate effective image selection algorithms using

reinforcement learning and a Self-supervised NeRF Image Selector (SNIS), which consists of a simple deep neural network structure. We employ a deep Q-network (DDQN) for reinforcement learning, adopting an approach that continually improves the model through iterative learning. As training progresses, rewards are assigned based on the model's performance, aiding the reinforcement learning agent in developing better image selection strategies. Meanwhile, the SNIS uses a self-supervised learning framework, generating new supervised learning labels that reflect the model's learning outcomes, akin to the rewards in reinforcement learning. This novel approach supports more effective model training by continually refining the image selection process based on the model's performance.

By comparing and evaluating the performance of these two methodologies, we aim to determine the most effective approach for optimizing NeRF training. Our goal is to develop a model that delivers superior performance by leveraging the strengths of both reinforcement learning and self-supervised learning. Through extensive experimentation and analysis, we seek to provide insights into the most effective strategies for image selection and to propose a robust algorithm that enhances the accuracy and efficiency of NeRF models. By exploring and implementing advanced image selection algorithms, we aim to enhance the model's ability to accurately reconstruct 3D scenes while minimizing the image numbers required to train the NeRF model for desired performance. This study ultimately contributes to the development of more efficient and effective NeRF models, with broad implications for various applications in AI and computer vision.

1.3 Purpose of Study

The objective of this research is to analyze the spatial distribution of the most effective image sets for training Neural Radiance Fields (NeRF) models and to propose an algorithm capable of predicting such distributions. Specifically, this study evaluates the combinations of images captured from various viewpoints and angles that enable NeRF models to achieve optimal performance. We aim to systematically analyze the impact of these image sets on model training, focusing on how sets with specific spatial distributions

contribute to maximizing the rendering performance of NeRF models. By understanding the influence of different image distributions, we seek to develop a predictive algorithm that can identify the most beneficial configurations for NeRF training.

To achieve this, our research leverages reinforcement learning (RL) and multi-layer perceptrons (MLP). The RL approach uses a deep Q-network (DDQN) to iteratively improve the model by optimizing image selection strategies through trial and error, guided by performance-based rewards. Meanwhile, the MLP-based method employs a self-supervised learning framework, where the model generates new learning labels that reflect its performance, akin to the reward system in reinforcement learning. By comparing and evaluating the performance of these two methodologies in the image selection process, we aim to determine the optimal approach. This involves conducting extensive experiments to assess how each method influences the efficiency and accuracy of NeRF model training. We explore various configurations, including different neural network architectures, hyperparameter settings, and the number of selected camera positions, to identify the critical factors that contribute to the optimization of NeRF training.

The ultimate goal of this research is to maximize the training efficiency of NeRF models. By developing and validating an algorithm that can predict the optimal spatial distribution of images, we hope to contribute to the advancement of 3D scene rendering technologies. Our findings are expected to provide valuable insights into the design of image selection strategies that can enhance the performance of NeRF models, making them more efficient and effective in various applications. In summary, this study aims to:

- Analyze the optimal distribution of image viewpoint (camera) sets for NeRF training.
- Develop a predictive algorithm for optimal image distribution.
- Compare the performance of reinforcement learning and Self-supervised Image Selector pipelines.
- Contribute to the development of efficient and effective image selection strategies for NeRF models.

By achieving these objectives, we aim to pave the way for more advanced and practical applications of NeRF in fields such as virtual reality, robotics, and computer vision.

2 Related Works

Neural implicit representation for 3D geometry is a method within a model to depict a scene, and it has attracted growing attention amid active research on representation methods utilizing neural networks. Neural rendering has gained significant acclaim as an implicit scene representation method, being actively employed in various applications of 3D computer vision, such as 3D reconstruction [1, 2, 3], 3D-aware generation [4, 5, 6]and novel view synthesis [7, 8, 9].

3D Reconstruction In recent research, neural implicit surface reconstruction techniques have surged in popularity for multi-view 3D reconstruction. Unlike conventional multi-view stereo methods, these approaches yield smoother and more comprehensive reconstructions, leveraging the inherent smoothness bias of neural networks. Although cutting-edge neural implicit methods have shown good performance in reconstructing straightforward scenes from numerous input views, their efficacy diminishes for large, complicated scenes and sparse viewpoints. Inspired by recent advancements in monocular geometry prediction, [3] focuses on intrinsic ambiguity in RGB reconstruction loss in enhancing neural implicit surface reconstruction. The authors demonstrate that depth and normal cues forecasted by general-purpose monocular estimators substantially enhance reconstruction quality and reduce optimization time. They scrutinize various design alternatives for representing neural implicit surfaces, encompassing monolithic MLP models, single-grid, and multi-resolution grid representations. More recently, [2] has addressed the limitations of neural implicit surface representation methods in handling texture-less planar regions commonly found in indoor scenes. Existing solutions rely on image priors derived from large annotated datasets. The authors propose a self-supervised super-plane constraint, leveraging geometric cues from predicted surfaces to improve plane region reconstruction without additional annotations. They introduce an iterative training scheme

involving pixel grouping to form super-planes and optimize the reconstruction network with a super-plane constraint. Surprisingly, models trained with super-planes outperform those using annotated planes, as super-planes cover larger areas and enhance training stability. Extensive experiments demonstrate that the self-supervised super-plane constraint significantly enhances 3D reconstruction quality, surpassing ground truth plane segmentation methods.

Research on 3D reconstruction of complex dynamic scenes is as actively studied as on static scenes. Achieving accurate 3D reconstruction in dynamic scenes with cameras has been difficult. To address this challenge, [10] proposes a neural invertible deforming network to represent and constrain the non-rigid deformations. Given the possibility of dynamic scene surface topology evolving, they utilize a strategy sensitive to topology changes to establish correspondence between the fused frames. Recently, [1] proposed R3D3, which integrates geometric estimation leveraging spatial-temporal information from multiple cameras with monocular depth refinement. It incorporates multi-camera feature correlation and dense bundle adjustment operators to improve geometric depth and pose estimates.

3D aware generation Generating content with an understanding of the three-dimensional structure, 3D-aware generation, involves considering the spatial relationships and characteristics of objects or scenes when creating images or videos. In the context of generative models like GANs (Generative Adversarial Networks), this approach allows for more realistic and accurate representations, particularly when viewed from different angles or in different poses. Recent efforts have aimed to make generative models 3D-aware, connecting the 2D image space with the 3D physical world. Some approaches incorporate NeRF into GANs to serve as a 3D prior, mapping 3D coordinates to pixel values. However, NeRF's implicit function has a limited receptive field, hindering global structure awareness. Basically, NeRF relies on volume rendering, which can be computationally expensive for high-resolution results, so it increases optimization complexity. To address these challenges, [6] introduces a framework for high-fidelity 3D-aware image synthesis. They train the model to learn a feature volume to represent structure, transformed into

a feature field resembling NeRF. This field is then converted into a 2D feature map for appearance synthesis using a neural renderer to facilitate independent control of shape and appearance. Also, [4] presents a generative adversarial network creating photorealistic full-body human images with consistent appearances across different view angles and body poses. Addressing representational and computational challenges, our novel generator architecture combines a 2D convolutional backbone with a 3D pose mapping network, leveraging 3D human mesh conditioning. In addition, [5] extends the capabilities of image-based 3D GANs to video editing and represents the input video in an identity-preserving and temporally consistent way, by introducing GAN inversion and optical flow-guided compositing. The GAN inversion jointly embeds in multiple frames and optimizes for the camera parameters to be tailored to 3D GANs. They also demonstrate the potential of intrinsic properties of 3D GANs with high-fidelity face edits showing novel views.

Novel view synthesis To create additional perspectives of scenes or objects, we can generate previously unseen views of a scene from a set of sparse input images taken from different viewpoints. To create realistic and coherent views that accurately depict how the scene or object would appear from different angles or viewpoints, many studies are being conducted. The neural radiance fields (NeRF) is one of the most representative models in this field, demonstrating impressive results in view synthesis for objects and specific spatial areas. However, there are remaining difficulties with "unbounded" scenes, where the camera can face any direction and objects may be located at any distance. Current methods often result in blurry or low-resolution images, slow training times, and artifacts due to reconstructing large scenes from limited image data. To tackle these issues, [9] present mip-NeRF 360, an extension of mip-NeRF [11], designed for 360-degree camera rotation. They parameterize nonlinear scenes and focus on distortion-based regularization to decrease the complexities of unbounded scenes. Also, Based on the need to synthesize novel views from a single input image and generalize across various object categories using a unified model, [7] propose combining global and local features to create a comprehensive 3D representation. In their method, a vision transformer learns global features, while a

2D convolutional network is trained for local features and a multi-layer perceptron (MLP) network is trained to generate volume rendering images. This method allows the network to reconstruct unseen regions without imposing constraints such as symmetry or standard coordinate systems.

To synthesize novel views from a single image without encountering issues, many methods project points to the input image plane and aggregate 2D features for volume rendering. However, in cases of severe occlusion, this projection fails to resolve uncertainty, leading to blurry renderings with insufficient details. In [8], The authors propose NerfDiff, which is a generative framework for single image view synthesis, by distilling the knowledge of a 3D-aware conditional diffusion model (CDM) into NeRF through synthesizing and refining a set of virtual views at test time. However, these methods necessitate multiple scene views during training, and their slow fine-tuning process hampers real-time usage. Moreover, there are many challenges like poor learning performance in deformable scenes and sparse viewpoints, and slow rendering speeds. To improve practical application efficiency, many recent research has addressed these issues to resolve the limitations of NeRF.

To address deformable scenes, several works [12, 13, 14] proposed optimizing the neural radiance field and extending it to a dynamic domain. These works aim to overcome the limitation of being only applicable to static scenes and empower the handling of unbounded scenes. NeRF demands an extended optimization time for volumetric representation. To reduce the rendering time complexity, [15] introduces a novel approach utilizing precomputed Neural Transmittance Functions to expedite neural reflectance field rendering, eliminating the need for laborious ray marching. [16] uses a single-network prediction of ray sample locations in view rays, utilizing a classification network and depth values to encode surface locations instead of directly estimating depth. [17] incorporates a view-dependent computing ordering scheme to alleviate inefficiencies to improve the speed of rendering images. More recently, [18] optimizes lightweight models suitable for resource-constrained devices where only one forward pass is executed on a ray to predict

pixel color. [19] extracts knowledge from a pre-trained dynamic NeRF model and handles non-rigid deformations by integrating hyperspace representation into a deformation field.

Neural radiance field (NeRF) [20] initially demonstrated its effectiveness with a large number of images in LLFF NeRF Real dataset [21]. Research aimed at effectively representing neural radiance fields even with sparse input and minimizing the reliance on dense scene coverage has been actively conducted. [22] estimates the structure shape by evaluating pairwise similarities and encoding stereo correspondence and enables the model to need only sparse views during test time. [23] introduces a novel representation for visibility prediction to enhance rendering quality by focusing on visible image features, along with a devised loss for refining visibility during fine-tuning on specific scenes. Focused on depth relationships, [24] leverages depth information and geometry information from predicted depth to maintain 3D consistency. [25] broadens NeRF generalization with sparse views to outdoor scenes, introducing a geometric correction module that normalizes rendered depth and integrates it with light directions, along with an appearance correction module to mitigate rendering artifacts resulting from changes in viewpoint. To supervise the NeRF's predicted depth, [26] propose a training pipeline where augmented models learn alongside the NeRF. Additionally, they design a mechanism to selectively choose reliable depth estimates for supervision to avoid using inaccurate augmented models for specific regions. With Structure from Motion (SfM), [27] employs a sparse depth prior sampled from volume points across local-to-global geometric regions to alleviate bias in the depth prior, instead of relying on direct depth supervision.

However, these works only focus on improving the NeRF model itself rather than addressing the selection of images for training. Ensuring the model's optimal performance with sparse viewpoints is indeed a critical endeavor; however, it is imperative to also optimize the data selection process. In this paper, we mainly investigate the viewpoint selection of the most essential scenes to effectively train the model with fewer images while maintaining consistency by utilizing reinforcement learning.

3 Preliminary

3.1 Neural Radiance Fields (NeRF)

In the realm of methods aimed at generating 3D rendering views from 2D perspectives, Neural Radiance Fields (NeRF) [20] stands out as a 3D rendering technique known for its efficient performance with effective memory utilization. By employing MLP networks, NeRF facilitates rapid 3D rendering from untrained viewpoints. The objective of NeRF is to render images for 3D scenes when viewed from arbitrary perspectives. This necessitates an understanding of spatial information, including acquiring information about the Z-axis, which is not visible in 2D.

To address this, it directly optimizes the parameters of 5D scene representation to minimize the error in rendering a set of captured images. Static scenes are represented in terms of radiance emitted in each direction (θ, ϕ) at each point (x, y, z) in space, alongside a density at each point regulating the accumulation of radiance by rays passing through (x, y, z). This density acts as a differentiable opacity, modulating the amount of radiance accumulated when rays pass (x, y, z). The method achieves this optimization by employing a deep fully-connected neural network without convolutional layers, regressing from a single 5D coordinate (x, y, z, θ, ϕ) to a single volume density and view-dependent RGB color.

The MLP network $F_\Theta : (x, d) \rightarrow (c, \sigma)$ approximates this continuous 5D scene representation and optimizes its weights Θ to map each input 5D coordinate to its corresponding volume density and directional emitted color. The inputs to this function are the 3D location $\mathbf{x} = (x, y, z)$ and the 2D viewing direction (θ, ϕ), while the outputs are the emitted color $\mathbf{c} = (r, g, b)$ and volume density σ. NeRF's 5D neural radiance field represents scenes in terms of volume density and directional emitted radiance at any point in space. It renders the color of any ray passing through the scene using principles from classical volume rendering. The volume density $\sigma(x)$ can be interpreted as the differential probability of a ray terminating at an infinitesimal particle at location x. The expected color $C(r)$ of a camera ray $r(t) = o + td$ with near and far bounds t_n and t_f is as follows:

$$C(r) = \int_{t_n}^{t_f} T(t)\sigma(r(t))c(r(t), d)dt, \quad \text{where} \quad T(t) = \exp\left(-\int_{t_n}^{t} \sigma(r(s))ds\right) \tag{1}$$

The function $T(t)$ represents the accumulated transmittance along the ray from t_n to t. Essentially, it denotes the probability that the ray travels from t_n to t without encountering other particles. Rendering a view from this continuous neural radiance field requires estimating the integral $C(r)$ for each pixel of the desired virtual camera's traced camera ray.

They employed quadrature to compute this continuous integral. By partitioning the interval $[t_n, t_f]$ into N evenly-spaced bins and employing a stratified sampling approach where one sample is uniformly drawn from each interval, we enhance the performance of the representation's resolution. Utilizing these samples, we can estimate $C(r)$ using the spherical law of cosines discussed in Max's volume rendering review. The sampling approach and the expression for estimating $C(r)$ are as follows:

$$t_i \sim U\left(t_n + \frac{i-1}{N}(t_f - t_n), t_n + \frac{i}{N}(t_f - t_n)\right) \tag{2}$$

$$[\hat{C}(r) = \sum_{i=1}^{N} T_i(1 - \exp(-\sigma_i\delta_i))c_i] \tag{3}$$

Here, $T_i = \exp\left(-\sum_{j=1}^{i-1} \sigma_j\delta_j\right)$, and $\delta_i = t_{i+1} - t_i$ represents the distance between adjacent samples. This function for computing $\hat{C}(r)$ from the set of (c_i, σ_i) values is straightforwardly differentiable and reduces to traditional alpha compositing transformed into $\alpha_i = 1 - \exp(-\sigma_i\delta_i)$.

Positional encoding When the network F_Θ operates directly on the input coordinates (x, y, z, θ, ϕ), it fails to properly capture high-frequency variations in color and geometry, as highlighted in recent research [28], which indicates a bias of deep neural networks towards learning low-frequency functions. To address this issue, significant performance improvement can be achieved by reconfiguring F_Θ as a composition of two functions,

$F_\Theta = F_0^\Theta \circ \gamma$. Here, γ represents a mapping from R to a higher-dimensional space R2L, while F_0^Θ remains a conventional MLP. The encoding function can be represented as follows:

$$\gamma(p) = \left(\sin^2(0\pi p), \cos^2(0\pi p), \ldots, \sin^2((L-1)\pi p), \cos^2((L-1)\pi p)\right) \tag{4}$$

The function $\gamma(\cdot)$is separately applied to each of the three coordinate values x, y, z and to the three components of the Cartesian viewing direction unit vector d. This is referred to as positional encoding, utilizing this function to map continuous input coordinates into a higher dimensional space, thereby enabling the MLP to easily approximate higher frequency functions.

Hierarchical volume sampling The rendering strategy of applying NeRF at N query points along each camera ray has the inefficiency of repeatedly sampling free space and occluded regions that do not contribute to the final rendered image. Hierarchical volume sampling allocates samples proportionally to their expected impact on the final rendering to overcome this inefficiency. Instead of using a single network to represent the scene, "coarse" and "fine" networks can be trained simultaneously.

Initially, hierarchical sampling is employed to sample a set of N_c locations, and the "coarse" network is evaluated at these positions according to equations 2 and 3. Considering the output of this "coarse" network, a more informed sampling of points along each ray can be generated in a direction that better samples the volume. To achieve this, the alpha-blended color values from the "coarse" network, as described in Equation 3, can be expressed as a weighted sum of all samples c_i as follows:

$$\hat{C}(r) = \sum_{i=1}^{N} T_i(1 - \exp(-\sigma_i\delta_i))c_i, \quad \text{where } T_i = \exp\left(-\sum_{j=1}^{i-1}\sigma_j\delta_j\right) \tag{5}$$

When normalizing these weights as $\hat{w}_i = w_i / \sum_{j=1}^{N_c} w_j$, it generates a piecewise constant probability density function along each ray. From this distribution, we sample the second set of N_f locations using inverse transform sampling, evaluate the "fine" network

at the union of the first and second set of samples, and compute the final rendered color of the ray using all $N_c + N_f$ samples. This procedure makes the rendering process more efficient by allocating more samples to regions expected to contain visible content.

3.2 Instant-NGP

Instant Neural Graphics Primitives (INGP) [29] is a novel approach to decrease the computational cost of training and evaluating neural graphics primitives. Through the adoption of a novel input encoding technique, Multiresolution Hash Encoding 3.2, the method enables the utilization of a smaller neural network without compromising quality. This is achieved by integrating a multiresolution hash table of trainable feature vectors into the network architecture. This design simplifies the structure and facilitates parallelization on modern GPUs. Implementation with fully-fused CUDA kernels minimizes wasted resources, resulting in a substantial acceleration of training high-fidelity neural graphics primitives within seconds and rendering them in milliseconds.

Multiresolution Hash Encoding Encoding inputs enhances the performance of our fully connected neural network $m(y; \Phi)$ by improving the quality of its inputs y through an encoding process $\text{enc}(x; \theta)$, which is adjustable to optimize approximation quality and training speed across various applications. In addition to the weight parameters Φ, our neural network incorporates trainable encoding parameters θ. These parameters are structured into L levels, each comprising up to T feature vectors with a dimensionality of F. Each level, as depicted in the figure by red and blue segments, operates independently and stores feature vectors at the vertices of a grid. The resolution of this grid is selected to follow a geometric progression ranging from the coarsest to the finest resolutions $[N_{\min}, N_{\max}]$. By determining proper value of $N_{\max}$, we can calculate the resolution for level l using the provided equations:

$$N_l := \left\lfloor N_{\min} \cdot b^l \right\rfloor \tag{6}$$

$$b := \exp\left(\frac{\ln N_{\max} - \ln N_{\min}}{L-1}\right) \tag{7}$$

The input coordinate $x \in \mathbb{R}^d$ is scaled by that grid resolution of each level l before rounding down and up by Equation 8. $\lfloor x_l \rfloor$ and $\lceil x_l \rceil$ span a voxel with $2d$ integer vertices in $\mathbb{Z}^d$. These array has the maximum length of T.

$$\lfloor x_l \rfloor := \lfloor x \cdot N_l \rfloor, \lceil x_l \rceil := \lceil x \cdot N_l \rceil \tag{8}$$

Each corner is mapped to the level's corresponding feature vector array. To produce y, which is the encoded input $\text{enc}(x;\theta)$ to the MLP $m(y;\Phi)$, feature vectors of each level have to be interpolated. The feature vectors of the corners are d-linearly interpolated within its hyper-cube, regarding the relative position of x. This process is independent for each L levels. At coarser levels where a denser grid requires fewer than T parameters, meaning the volume of the grid does not exceed T, the mapping corresponds one-to-one. However, for finer levels, a hash function $h : \mathbb{Z}^d \to \mathbb{Z}_T$ is used to navigate within the array, to essentially treat it as a hash table. Without explicit collision handling, optimization with the gradient strategically stores relevant sparse detail information. Any potential collisions are resolved by the subsequent neural network $m(y;\Phi)$. Here, $O(T)$ is the number of trainable encoding parameters θ.

The hash algorithm [30] involves calculating the XOR of unique, large prime numbers multiplied by the input coordinate x:

$$h(x) = \bigoplus_{i=1}^{d} x_i \pi_i) \mod T \tag{9}$$

where $\oplus$ means the bitwise XOR operation. XOR operation of the results of linear congruential permutations on each dimension's value reduces the correlation between dimensions in the hashed value. Achieving independence typically requires only $d-1$ of the d dimensions to be permuted. Conventionally, in Instant-ngp, the values of π are set as follows: $\pi_1 := 1$, $\pi_2 = 2,654,435,761$, and $\pi_3 = 805,459,861$.

Implicit hash collision resolution By using complementary various resolution levels, the encoding can retain sufficient scene information even in the presence of hash collisions. In coarser levels, there are no collisions, but only low-resolution versions of the scene can be represented. Conversely, finer levels can represent high-resolution versions but suffer from many collisions. This is because finer grid resolutions lead to more occurrences of different points hashing to the same table entry. When such collisions occur, the gradients of the training samples are averaged. Visually distinct features or surfaces with high visibility and density contribute significantly to the reconstructed image, causing substantial changes to the hash table entries. In contrast, points in empty space that refer to the same table entry have a much smaller weight in the image reconstruction. Consequently, the more important samples dominate the collision average, increasing the influence of their gradients, and the aliased table entry is naturally optimized to reflect the needs of the higher-weighted points. The multi-resolution of hash encoding covers the range from a guaranteed collision-free minimum resolution N_{min} to the required maximum resolution N_{max}. Therefore, all scales where meaningful learning can occur can be included.

3.3 Dual-DQN (DDQN)

One of the challenges in Q-Learning, and by extension DQN, is the overestimation of Q-values. Overestimation occurs because Q-Learning uses the maximum estimated Q-value for the next state to update the current state's Q-value. Mathematically, the Q-learning update rule is:

$$Q(s,a) \leftarrow Q(s,a) + \alpha \left(r + \gamma \max_{a'} Q(s',a') - Q(s,a) \right) \tag{10}$$

Here, r is the reward, γ is the discount factor, and $max_{a'} A(s',a')$ is the maximum Q-value for the next state s'. Since this update relies on the maximum estimated Q-value, it can lead to an upward bias, resulting in overoptimistic value estimates.

Dual-DQN [31], aims to mitigate the overestimation bias by decoupling the action selection and action evaluation processes. The core idea is to use two separate Q-value

estimations to reduce the bias in Q-value updates. In Dual-DQN, two sets of weights are maintained:

- Online Network: The primary network that is used to select actions.
- Target Network: A secondary network that is used to evaluate the Q-values of the selected actions.

The update rule in Dual-DQN is modified to use the action selected by the online network and the value evaluated by the target network:

$$Q(s,a) \leftarrow Q(s,a) + \alpha \left(r + \gamma Q_{\text{target}}(s', \arg\max_{a'} Q_{\text{online}}(s', a')) - Q(s,a) \right) \tag{11}$$

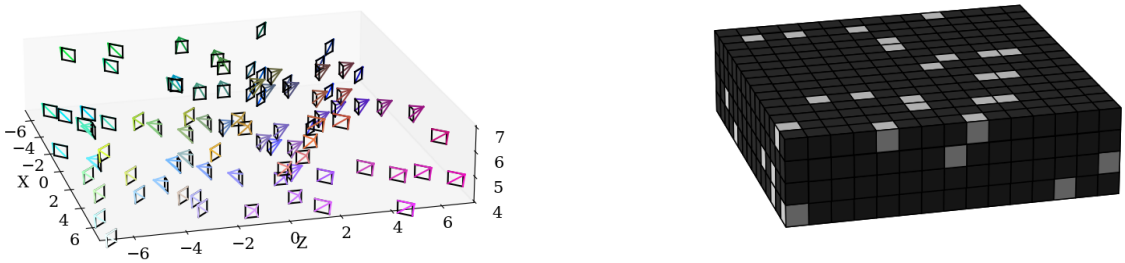
Here, Q_{online} represents the Q-values from the online network, and Q_{target} represents the Q-values from the target network. Dual-DQN offers several significant advantages over traditional DQN, making it a valuable improvement in the field of deep reinforcement learning. One of the primary benefits is the reduction of overestimation bias in Q-value predictions. By decoupling the action selection and action evaluation processes, Dual-DQN mitigates the inherent upward bias found in standard DQN, leading to more accurate and reliable value estimations. This is achieved by using the online network to select actions and the target network to evaluate them, ensuring that the Q-values are not overly optimistic. Additionally, this approach contributes to improved training stability. The target network, which is updated less frequently, provides a stable reference for the Q-value updates, preventing the fluctuations and divergence that can occur when using a single network. Empirical studies have demonstrated that Dual-DQN consistently outperforms DQN in various complex environments, including high-dimensional state spaces and tasks with intricate dynamics. The combination of reduced bias and enhanced stability enables more robust learning, ultimately leading to better performance and faster convergence.

4 Methodology

4.1 Dataset

Instead of directly manipulating robots, we plan to work in the Unity environment. To accomplish this, we will initially enclose a target object made of bricks and create a scene to contain this object. This target object will be used to simulate the leap of an airplane and will be represented as a cage. Additionally, we will position a camera within Unity to capture scene images and expect to obtain accurate camera coordinates and rotation matrices.

Our dataset consists of images obtained from coordinates divided into 1-unit increments, where x and z coordinates range from -7 to 7, and y coordinates range from 4 to 7. This method of acquiring the dataset will be visually demonstrated in Figure **??**. Consequently, the dataset comprises a total of 900 images. This coordinate setup ensures consistent division of the experimental space and provides various scenarios necessary for evaluating the results.



(a) Camera distribution in the Unity environment

(b) Embedding matrix of the camera distribution

Figure 1: An example of a camera distribution and its corresponding embedded form. The camera poses are embedded as a spatial arrangement of cameras within the grid. In this matrix, cells containing cameras are assigned a value of 1, while cells without cameras are assigned a value of 0.

In this baseline experiment, images are randomly selected from the pool of 900. Each

image is identified by a unique index number, and when selecting images randomly, a random selection of n numbers from 0 to 899 is made, followed by the selection of the corresponding images. Regarding reinforcement learning, the action space is also composed of discrete index numbers ranging from 0 to 899. Selecting one number corresponds to selecting the respective image. The order of these indices is arranged in ascending order of x and z coordinates to flatten spatial information into one dimension. Thus, this approach ensures the preservation of spatiotemporal information and ensures the consistency and effectiveness of the experiments.

4.2 Reinforcement Learning Agent and Environment

To implement a more stable and effective reinforcement learning algorithm, we chose DDQN, which is an improved version of Deep Q-Networks (DQN) that generally exhibits high performance in Q-learning-based reinforcement learning. Additionally, DDQN helps mitigate the overestimation bias issue present in DQN. The training steps for the DQN agent is as follows:

1. **Observation of environment and state**: Initially, the agent interacts with the environment and observes the current state. This state could be the pixel values of a game screen or the sensor data of a robot, for example. In our research, this state represents the information about which camera position is selected within the space corresponding to the Unity environment. We represent this information as a 15*15*4 grid, as shown in the Figure **??**. The grid consists of blocks where the value 1 indicates the coordinates where the camera is positioned, and the value 0 represents empty space in binary.

2. **Action selection**: Based on the observed state, the agent selects one of the possible actions. This is done while maintaining a balance between random exploration and optimal actions, for example, using the ϵ-greedy method. In our model, the agent selects one of the 900 images by choosing one integer from 0 to 899.

3. **Execution of action and acquisition of reward**: The selected action is applied

to the environment, and the agent receives a reward from the environment. At each step, the NeRF model is trained with the selected images, and rewards are obtained based on PSNR and SSIM scores. These rewards indicate the performance of taking a specific action in a given state, and the agent's goal is to maximize these rewards.

4. **Experience storage**: The agent stores the state, selected action, reward, and next state in memory. These experiences are used for future learning. In this experiment, the reward function is set to vary according to PSNR and SSIM scores.

5. **Mini-batch learning**: Periodically, the agent selects random mini-batches from memory and trains the network using the Q-learning algorithm. The double Q-learning technique, one of the key improvements of DQN, is applied, separating the Q-values of actions chosen in the current state and actions chosen in the next state to reduce overestimation bias.

6. **Target Q-value computation**: DDQN also computes target Q-values using a target network. This is used during network parameter updates to enhance the stability of learning.

7. **Network update**: The network parameters are updated using gradients obtained from mini-batch learning. This process is performed to more accurately estimate the value of actions in a given state.

8. **Iteration**: The above process is repeated, allowing the agent to accumulate experience, update Q-values, and adapt to the environment to learn better policies. This enables the agent to select optimal actions and obtain maximum rewards in a given environment.

4.3 Implementation

In this study, we employed Instant Neural Graphics Primitives (Instant NGP) with the Neural Radiance Field (NeRF) model. Instant NGP is particularly noted for its

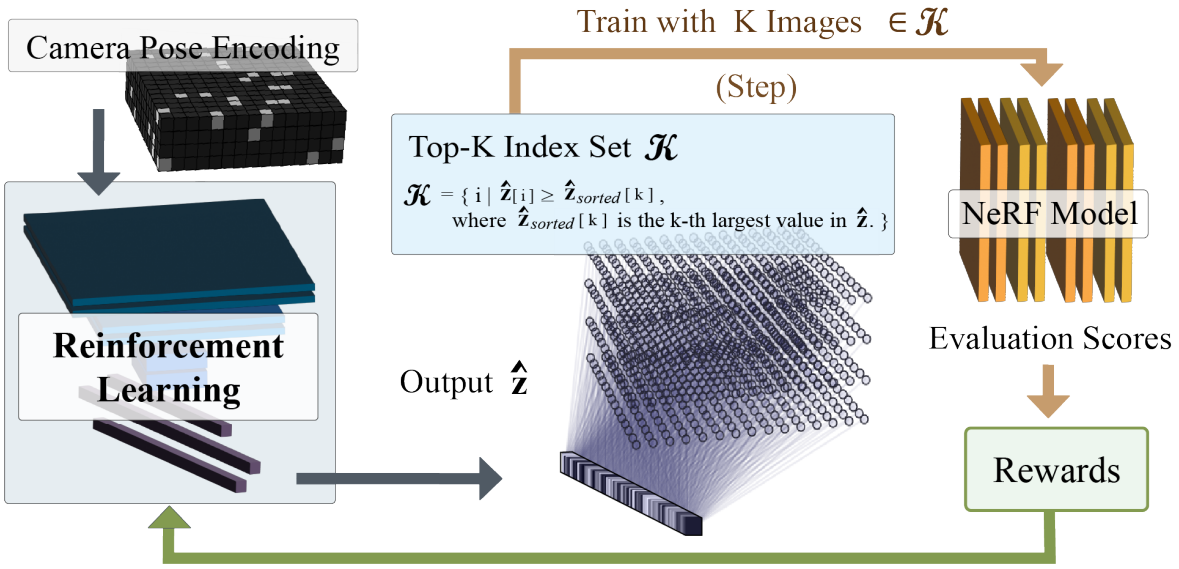


Figure 2: Overall process of reinforcement learning framework

rapid training times among NeRF models. Given that the reinforcement learning (RL) agent must train the selected image at each step and perform evaluations to compute rewards, we prioritized models with swift training capabilities. Instant NGP achieves comparable quality to state-of-the-art techniques after only a few seconds of training, facilitating rapid prototyping, iterative experiments, and offering significant benefits for real-time graphics applications. Despite its quick training time, Instant NGP maintains high visual quality, including high resolution and detailed local features, enabling more realistic environment reconstruction.

Furthermore, Instant NGP features an adaptive and efficient model using multi-resolution hash encoding, which allows for efficient data storage and processing, providing flexibility for various tasks. Utilizing a multi-layer perceptron (MLP), it demonstrates excellent performance even with a small model size, conserving storage space and reducing the resources required for model deployment and execution. In our experiments, training for 1,000 iterations takes approximately one minute, demonstrating remarkably fast learning as evidenced by the performance detailed in Section 4.

The overall flow of reinforcement learning (RL) training is depicted in Figure 2 and Algorithm 1. As shown in the figure, we encode the state of the camera distribution in space to create the model input as a matrix. By placing a grid around the target scene in the Unity environment and positioning cameras at these grid points, if a cell contains a

Algorithm 1 Step Procedure in the Environment

Input: The newly selected camera position chosen by the agent.
Output: State S and reward R

1: Update $S \leftarrow new\_pose$ ▷ Update the state matrix as shown in Figure **??**
2: Update $training\_dataset \leftarrow new\_pose$ ▷ Include the new pose in the training dataset
3: Train Instant-ngp with the updated $training\_dataset$
4: Validate Instant-ngp to obtain $PSNR$ and $SSIM$ metrics
5: $R \leftarrow reward\_function(PSNR, SSIM)$ ▷ Compute the reward using Equation **??**
6: **Output** S and R

camera (i.e., a camera is present at that location), the corresponding value in the matrix is set to 1; otherwise, it is set to 0. This model input is the State S as described in Algorithm 1. The reinforcement learning model processes this matrix input to generate the output $\hat{z}$ from the deep neural network (DNN) model. From $\hat{z}$, we extract k actions, where k is a predetermined hyperparameter representing the number of camera positions to be predicted at each step. We construct a set $\mathfrak{K}$ containing the indices of the top-k values in $\hat{z}$. The images corresponding to these indices are then used to train the NeRF model, from which we obtain the reward.

At each step, the model predicts k new positions, updates S, and computes the reward R. In this study, we design the reward function using PSNR and SSIM scores. These rewards are used to update the reinforcement learning model and evaluate the selected k camera positions. This approach leverages the efficiency and rapid training capabilities of Instant NGP to facilitate real-time reinforcement learning in dynamic environments. The adaptive nature of the model, combined with its ability to maintain high visual quality, makes it a robust solution for applications requiring frequent updates and evaluations of camera positions.

The proposed Self-supervised NeRF Image Selector (SNIS) is designed to apply the reward function learning paradigm of reinforcement learning to train a simple structure neural network. Rather than iteratively selecting images at each episode, our model predicts image poses in a single inference step utilizing an efficient neural network architecture. This methodology substantially mitigates computational complexity by employing a lightweight neural network that integrates convolutional neural network (CNN) and multilayer perceptron (MLP) architectures. The comprehensive process of SNIS is depicted

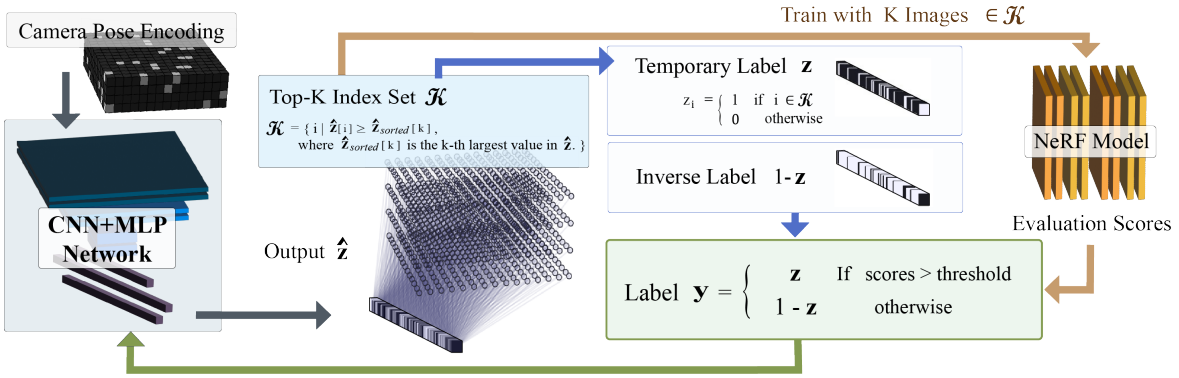


Figure 3: Overall process of SNIS framework

in Figure 3. The input to the SNIS model comprises the encoded camera distribution of the target scene, represented as a matrix where each position is assigned a zero in the absence of a camera. This matrix constitutes the foundational data from which the SNIS model generates its output, specifically designed to form the Top-k Index set $\mathfrak{K}$. This set identifies the most pertinent camera positions for subsequent processing.

To facilitate effective training of our network, we generate temporary labels z and their corresponding inverse labels $1 - z$. The values at the indices in the index set $\mathfrak{K}$ are set to 1, while all other positions are set to 0, creating a binary vector z. The inverse labels $1-z$ are binary vectors with values flipped from z. When the performance of the NeRF model, trained with the selected k images, satisfies the predefined criteria, the labels remain as z. Conversely, if the performance is suboptimal, the inverse labels $1 - z$ are utilized for training. This adaptive labeling strategy ensures the network consistently receives informative feedback, optimizing its learning process. The training of the SNIS network leverages the label y, with the objective function defined as $L(\hat{z}, y) = CrossEntropy(\hat{z}, y)$. This loss function is instrumental in fine-tuning the network, guiding it towards accurate predictions of optimal camera positions. By employing this training framework, SNIS offers a highly efficient and effective method for selecting image poses in NeRF models. This approach significantly reduces computational overhead, enabling rapid processing without sacrificing performance quality. Consequently, SNIS is well-suited for real-time applications and iterative experimentation, providing a robust solution for dynamic and high-demand environments.

5 Results and Discussion

In this section, we analyze and compare the NeRF training performance improvements achieved by applying the RL framework and the SNIS methodology. Specifically, we delve into the efficacy of each approach in enhancing the quality and efficiency of NeRF training. By systematically applying the RL framework, we assess its ability to iteratively optimize camera positions through reinforcement learning, evaluating its impact on both the training duration and the resultant image quality. On the other hand, we examine the SNIS methodology's capacity to streamline the image selection process using a single-step neural network inference, highlighting its potential to significantly reduce computational overhead while maintaining high visual fidelity. Furthermore, we explore a range of experimental configurations to identify the critical factors that contribute to the optimization of NeRF training. These experiments are designed to investigate various elements such as the effect of different neural network architectures, the influence of hyperparameter settings, and the impact of varying the number of selected camera positions on the overall performance of NeRF models. By comparing these approaches under diverse conditions, we aim to provide a comprehensive understanding of the mechanisms that drive NeRF training optimization.

5.1 Optimal Selection of Camera Positions

5.1.1 Reinforcement Learning for NeRF Training

To evaluate how well reinforcement learning selects optimal images and how beneficial the images selected by reinforcement learning are for training the NeRF model, we set two baselines and compare their performance with our approach. First, let us consider a NeRF model trained with n_0 initial images. We train the NeRF model over a predetermined number of iterations k, adding more images during each iteration. Thus, the NeRF model is ultimately trained with $n_0 + n_1 + \ldots + n_k = N$ images. The NeRF model trained with N randomly selected images at once is referred to as Baseline 1. For Baseline 2, we start with a NeRF model trained with n_0 images and then further train

(a) Reference Images (b) Samples from 5.2 (c) Samples from 5.1.1

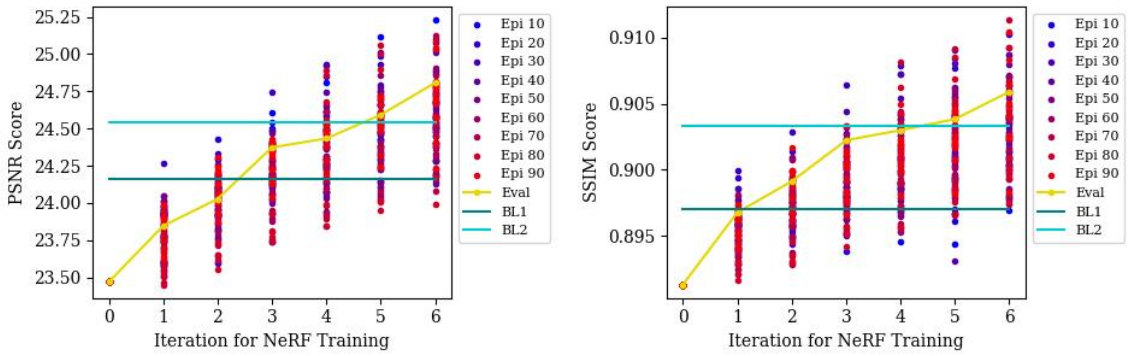
Figure 4: Sample evaluation result images. Column (b) shows the evaluation results from 5.2, which trained the NeRF network with 100 images with SNIS. Column (b) shows the evaluation results from 5.1.1, which trained the NeRF network with 130 randomly sampled images.

it with additional $n_1 + \ldots + n_k$ images in the same configuration as our reinforcement learning approach.

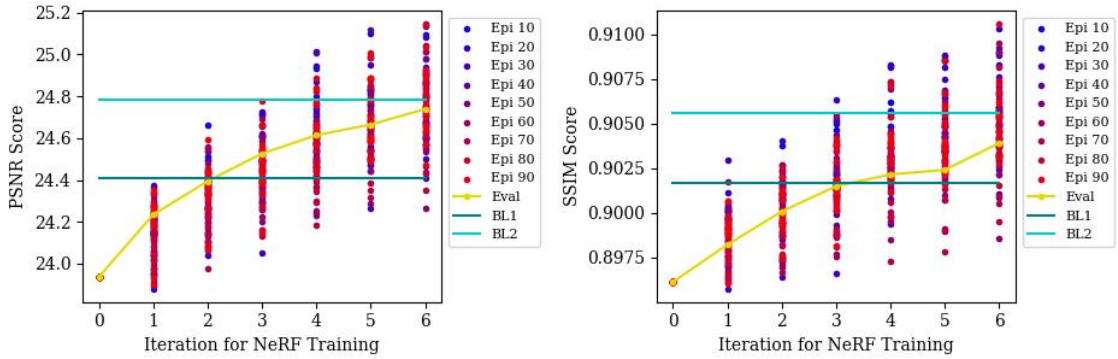
For each case, the number of iterations is fixed at 20,000. The final performance metric is the average of 100 training runs for each scenario. Given a model initially trained with n_0 random images, the results of using reinforcement learning to select and train with additional $n_1 + \ldots + n_k$ images are shown in Figure 5. The training was conducted over a total of 100 episodes, with the early episodes represented by blue points and the later episodes by red points. Since DDQN was used, the evaluations with the target network are shown with the yellow line. "All random BL" refers to Baseline 1, which is the model trained with N random sample images at once, and "Ini + addi BL" refers to Baseline 2, which is the model trained initially with n_0 images followed by additional training with the remaining images. Sample evaluation result of Baseline 1 with $N = 130$ is represented in Figure 4.

In the graph, the x-axis represents the iteration for NeRF training. In this experiment, $k = 10$, meaning the image selection and performance improvement processes for the NeRF model were carried out over 10 iterations. A large reward was given when Baseline 2 was exceeded, and moderate rewards were given whenever there were slight performance improvements.

The results indicate that while Baseline 1 is easily surpassed, Baseline 2 proves to be a more challenging benchmark. It can be observed that the models struggle to consistently outperform Baseline 2. This is especially evident in the evaluations shown with the yellow line, where the performance consistently falls short of exceeding Baseline 2. Despite surpassing Baseline 1 with relative ease, the difficulty in surpassing Baseline 2 highlights the increased challenge posed by this benchmark. The yellow line evaluations, which represent the target network evaluations using DDQN, consistently fail to outperform either of the baselines, demonstrating the significant difficulty in achieving superior performance compared to Baseline 2. The results for the experiment where k is reduced to 5 are shown in the Figure 6. In this scenario, the number of camera poses predicted by the RL agent at each step is increased to 10, with $n_1 = n_2 = \ldots = n_5 = 10$. In this



(a) 80 randomly sampled images + 50 images selected by RL

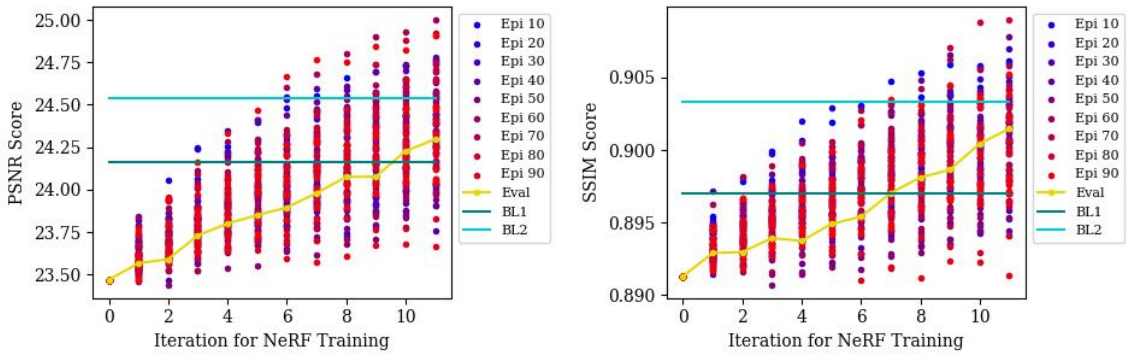


(b) 100 randomly sampled images + 50 images selected by RL

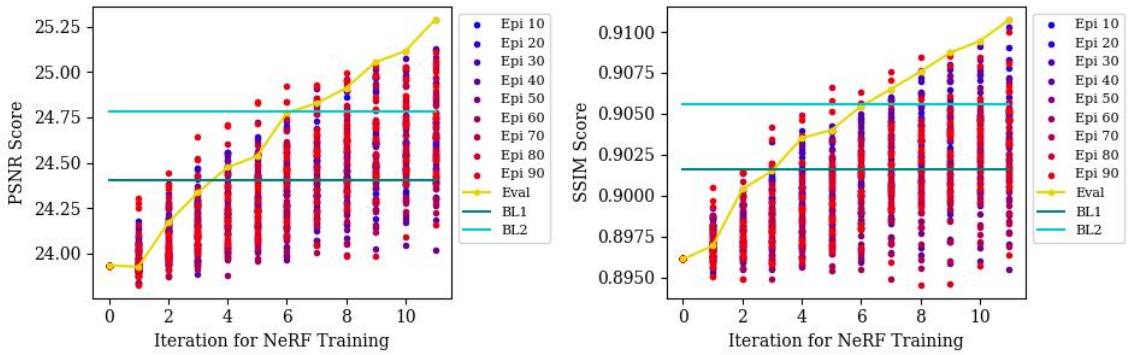
Figure 5: Performance comparison of training with RL (our method) and randomly sampled images, when k is 10.

case, the model ultimately fails to surpass Baseline 2 as well consistently.

To enhance the effectiveness of the reinforcement learning (RL) agent's training, an experiment was conducted in which episodes were terminated with a substantial reward once a predetermined criterion was met. The results of this experiment with $k = 10$ are illustrated in Figure 7. As in previous figures, the red lines represent more recent performance, while the blue lines indicate initial performance. The yellow line represents evaluation performance. When these lines exceed the Baseline 2 score, the episode terminates. As observed, the impact on the agent's learning efficiency remains modest. By setting the episodes to conclude upon exceeding a specific threshold, it was anticipated that this approach would significantly improve learning outcomes. However, the results did not meet these expectations. As depicted in Figure 7, the strategy of terminating



(a) 80 randomly sampled images + 50 images selected by RL



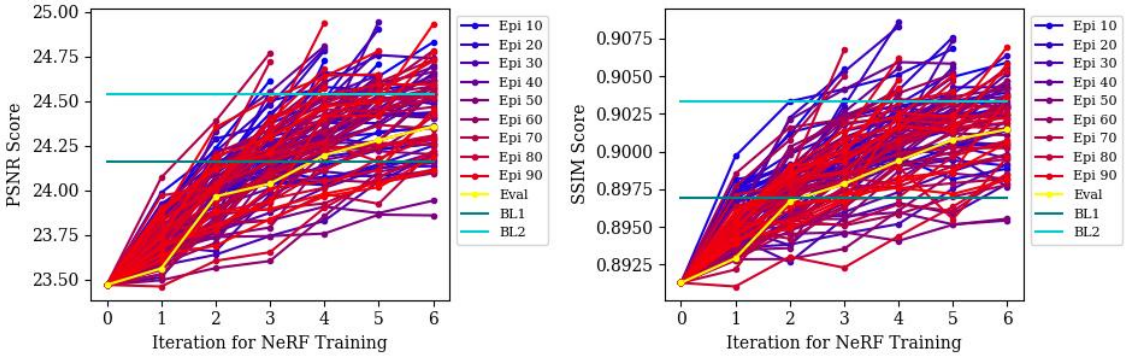
(b) 100 randomly sampled images + 50 images selected by RL

Figure 6: Performance comparison of training with RL (our method) and randomly sampled images, when k is 5.

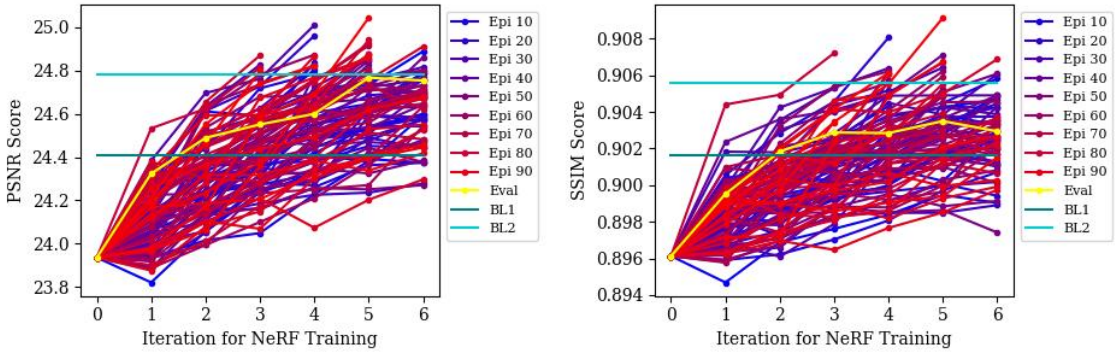
episodes upon reaching the defined criterion did not lead to a notable improvement in the agent's performance. This suggests that concluding episodes when the RL agent surpasses a set threshold may not substantially enhance learning efficiency.

5.1.2 Self-supervised NeRF Image Selector (SNIS)

We also proposed a self-supervised SNIS with a simple neural network structure. In this case, since images are not selected iteratively, $k = 1$. Therefore, when a NeRF model is already trained with n_0 images, SNIS predicts n_1 camera positions. We experimented with $n_1 = 50$ for the cases where n_0 is 80, 100, and 150. The results are shown in Figure 8. In the graph, we chose Baseline 1 from the previous section as the baseline, which has the same number of randomly selected images and the same iteration number configuration.



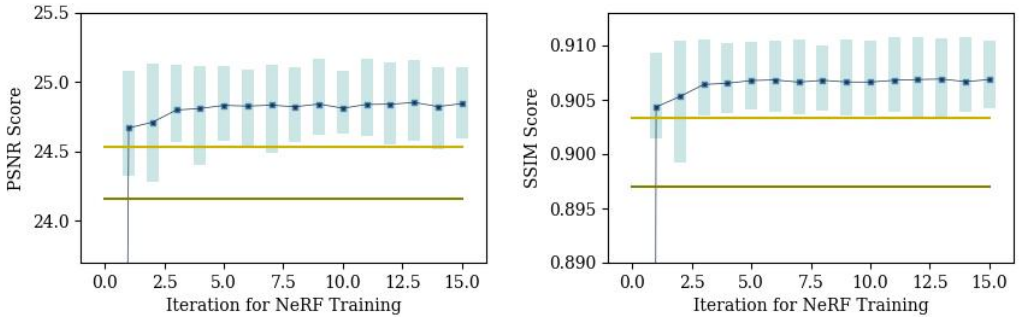
(a) 80 randomly sampled images + 50 images selected by RL



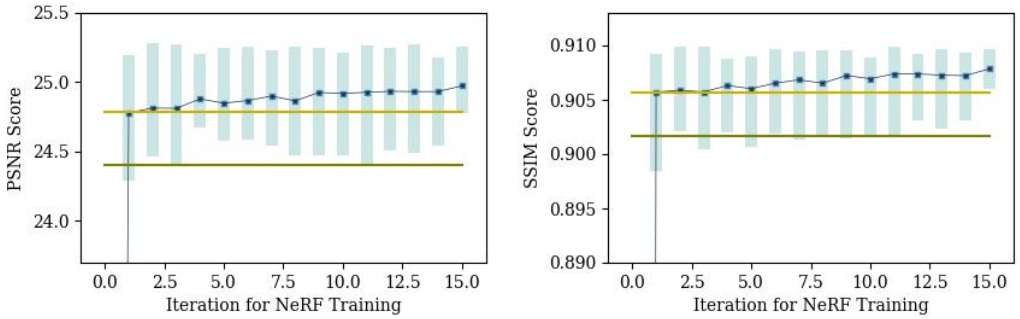
(b) 100 randomly sampled images + 50 images selected by RL

Figure 7: Performance comparison of training with RL where the episode ends once scores exceed the baseline.

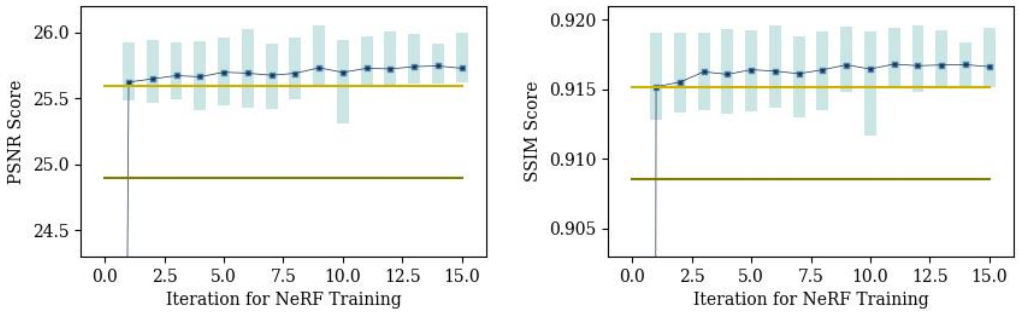
Given that NeRF training is highly susceptible to random noise, we conducted experiments using various random seeds. The graph illustrates that, in the majority of cases, the performance of SNIS training surpasses the baseline before stabilizing within five iterations of image selection. This stability is attributed to the learning method where, upon selecting an optimal camera pose, the loss function is structured to favor the selection of similar camera poses, thereby updating the SNIS weights accordingly. Even if the performance is suboptimal during the initial iterations, once a good result is achieved, the SNIS weights are updated to reflect this improvement, allowing the model to consistently surpass the baseline performance. These results demonstrate the effectiveness of the SNIS approach in selecting optimal camera poses, highlighting the improved training efficiency and performance of NeRF models. Particularly in scenarios where iterative selection and



(a) 80 randomly sampled images + 50 images selected by RL



(b) 100 randomly sampled images + 50 images selected by RL



(c) 150 randomly sampled images + 50 images selected by RL

Figure 8: Performance comparison of training with RL (our method) and randomly sampled images. The yellow line shows the baseline performance in which images are randomly selected. The brighter yellow line and the darker one indicate BL2 and BL1, respectively. Blue bars represent the ranges of 15 different random seed experiments.

training can be costly or impractical, SNIS can be a promising alternative for achieving good results with low computational requirements.

The reinforcement learning (RL) approach exhibited suboptimal performance, consistently failing to surpass the baseline. Furthermore, it required a substantial number of iterations, with each step necessitating a fixed number of iterations for training. This extensive computational demand resulted in prolonged training times, making the RL approach less efficient and more resource-intensive. In contrast, the SNIS approach demonstrated the capability to surpass the baseline with significantly fewer iterations, maintaining stable and reliable performance throughout. This efficiency, coupled with its ability to achieve superior results in a shorter time frame, underscores the advantages of SNIS over RL.

Given these findings, we have decided to shift our focus toward SNIS for further experimentation. By concentrating our efforts on this method, we aim to explore its full potential and further validate its effectiveness in optimizing camera pose selection and enhancing the performance of NeRF models.

5.2 SNIS Performance of Training from Scratch

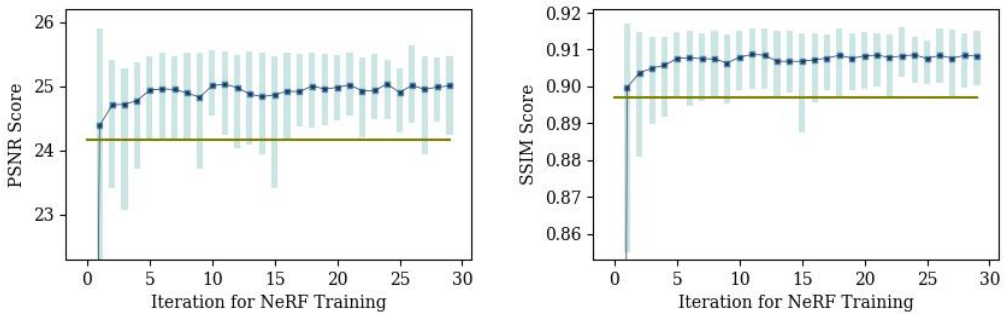


Figure 9: Performance of training from scratch with 130 images. The yellow line shows the baseline performance, where 130 images are randomly selected. Each blue bar represents the range of 15 different random seed experiments.

In addition to experiments with initial images already present, we also conducted experiments where no existing cameras are available, setting $n_0 = 0$ and $n_1 = N$, $N \in 100, 150, 200$. In this scenario, the SNIS input is simply an empty zero matrix. This setup

height Image Number	Random Selection		SNIS Mean after iter=10		Max performance	
	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
100	23.7269	0.8899	24.0438	0.8976	24.912	0.9074
150	24.4078	0.9016	25.3959	0.9149	25.911	0.9199
200	24.896	0.9086	25.786375	0.9208	26.519	0.9273

Table 1: Training results for different image number sets. *'Mean after iter=10'* indicates the average performance score after the SNIS prediction has stabilized. *'Max performance'* represents the highest performance scores achieved.

allows us to evaluate whether SNIS can achieve performance that surpasses the baseline without referencing the existing camera distribution.

As shown in Figure 9 and Table 1, even when starting from scratch, the image selection using SNIS significantly outperforms the random image selection method. When $n_1 = 100$, the sample images are visualized in Figure 4. This improvement is due to the ability of SNIS to identify and prioritize optimal camera poses from the outset, thereby facilitating more efficient and effective training. From these results, it is evident that SNIS is capable of providing substantial performance enhancements over random selection, even in scenarios where no initial camera data is available. This highlights the robustness and versatility of the SNIS approach in optimizing NeRF model training.

5.3 Investigation for Image Number Configuration

To determine the optimal extent of existing camera distribution for SNIS to achieve the best performance relative to the baseline, we conducted experiments by varying n_0 and n_1. We fixed $n_0+n_1 = N = 150$ and tested scenarios with $n_0 = 0, 5, 10, 20 \ldots, 120, 140, 145, 150$ and $n_1 = N - n_0$. The results of these experiments, showing the average performance over 5 trials at 30 iterations, are summarized in the Figure 10.

The bars represent the minimum and maximum performance values after 5 iterations of training, reflecting the range of performance stabilization. The solid line graph represents the average performance values across 5 trials at the conclusion of the training phase. From this graph, it is evident that our model achieves its highest performance when $n_0 = 30$ and $n_1 = 120$. However, there is a noticeable decline in performance for the

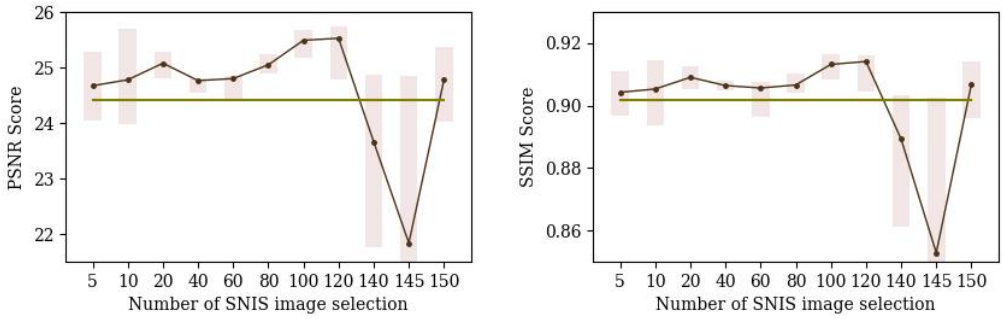


Figure 10: Performance of training from scratch.

n_1	PSNR	SSIM	n_1	PSNR	SSIM
0	24.4077	0.9016452	80	25.047	0.9065955
5	24.6704	0.9042506	100	25.489	0.913193
10	24.7776	0.9053156	120	25.5255	0.9141115
20	25.076	0.9090505	140	23.6584	0.8892766
40	24.7625	0.9064165	145	21.8358	0.8528506
60	24.7995	0.905633	150	24.7765	0.9067615

Table 2: Detailed scores of PSNR and SSIM of each n_1 case. Each score is the average of 5 trials. The total training Image number is $n_0 + n_1 = N = 150$.

case of $n_1 = 140$ and 145. This decline can be attributed to the specific characteristics of the NeRF model utilized in our experiments (Instant-NGP). The Instant NGP tends to exhibit poor learning progression when an excessively small number of images are used during the initial training phase. In this case, the model was initially trained using only 10 and 5 random images. Subsequently, the training continued with 140 and 145 images predicted by SNIS. The inadequate initial training with a few images hindered the model's ability to learn effectively in later stages, resulting in diminished performance. Conversely, for the case of $n_0 = 150$, the model was trained directly with the images predicted by SNIS, without prior training on a smaller set of images. This approach led to a significantly better performance compared to the case of $n_0 = 145$. The detailed numeric scores are represented in Table 2

By adjusting the values of n_0 and n_1, we aimed to identify the optimal balance that allows SNIS to maximize its effectiveness. The table provides a clear comparison of the performance metrics, offering insights into how different initial distributions of camera poses influence the efficiency and accuracy of the SNIS approach.

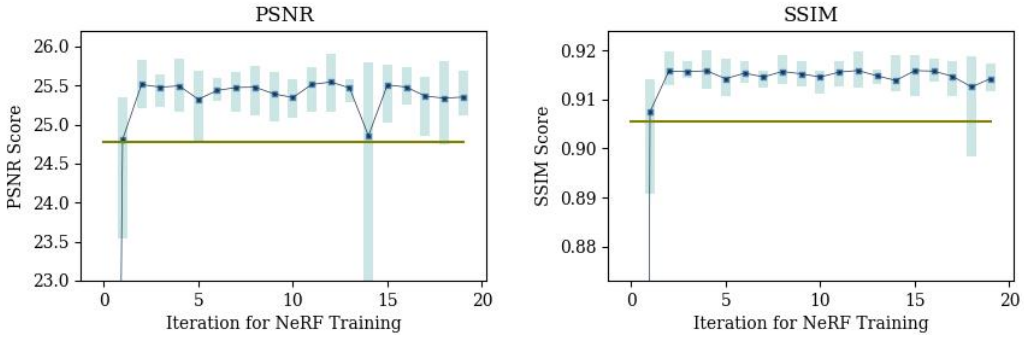
5.4 Label Coefficient Exploration

To further optimize SNIS, we introduced a parameter called alpha when creating the jet vector. Alpha is defined and utilized in the following equation to adjust the training process of SNIS: $y = \alpha * z$ if NeRF training score > history scores. This equation implies that during training, if the NeRF training score reaches a new peak, the corresponding label is given additional weight. We conducted experiments with alpha values set to 1, 2, 5, and 20. When alpha is 1, the equation simplifies to $y = z$, meaning no additional weight is applied. The graph in Figure 11 illustrates the results of these experiments. This outcome suggests that introducing additional weights to the labels when the NeRF training achieves the new best score does not necessarily enhance the learning process of SNIS. Instead, it indicates that small values of α lead to comparably stable training results.

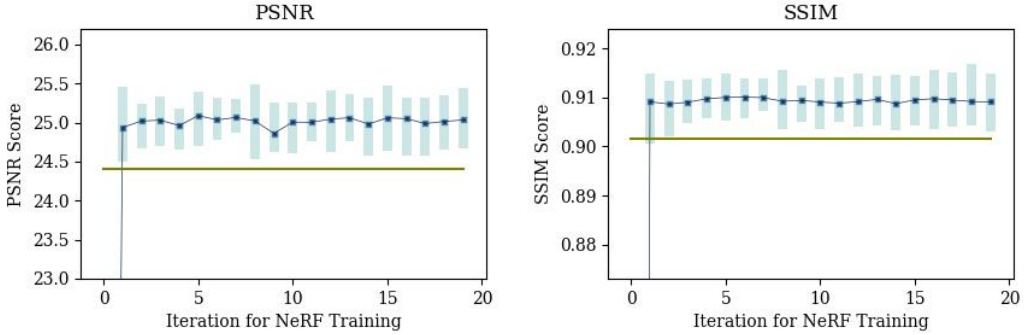
From these findings, we can infer that adding complexity through the alpha parameter may not always be beneficial. This insight is crucial for developing more efficient training strategies for SNIS, highlighting the importance of simplicity in optimizing the training process. In conclusion, our experiments demonstrate that while it is possible to adjust the training process of SNIS by introducing parameters such as alpha, the simplest approach—where $y = z$—proves to be the most effective. This discovery underscores the value of simplicity in the training algorithms and suggests that further research should focus on refining straightforward methods to enhance SNIS performance.

6 Conclusion

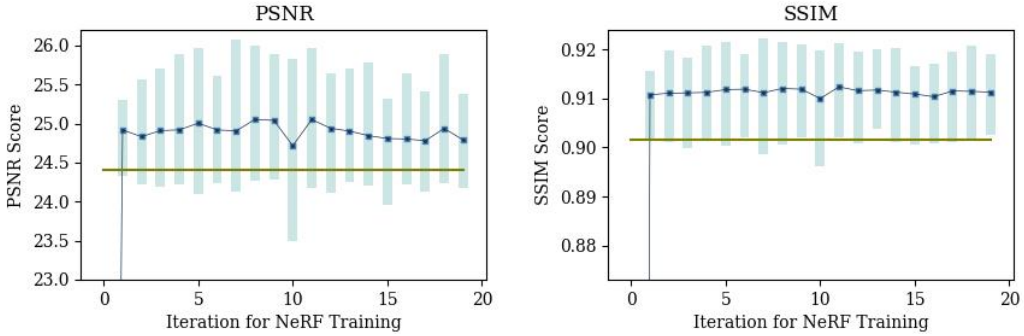
In this study, we conducted a comprehensive analysis of the NeRF training performance improvements achieved by applying both reinforcement learning (RL) and the Self-supervised NeRF Image Selector (SNIS) methodology. Our investigation focused on the efficacy of each approach in enhancing the quality and efficiency of NeRF training, considering various experimental configurations and performance metrics. The RL framework was designed to iteratively optimize camera positions through reinforcement



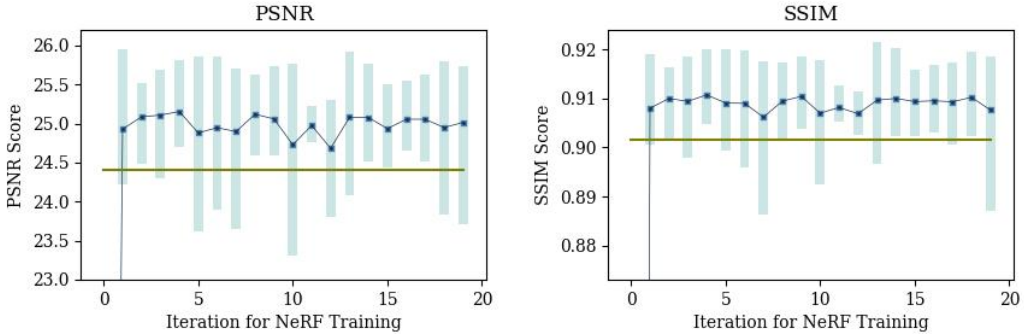
(a) $\alpha = 1$



(b) $\alpha = 2$



(c) $\alpha = 5$



(d) $\alpha = 20$

Figure 11: Performance comparison of training with various value of α where label $y = \alpha * z$.

learning, which indicated that this approach had a modest impact on learning efficiency. Specifically, using RL for the training NeRF model where episodes were terminated with substantial rewards upon meeting a predetermined criterion did not yield the anticipated improvements. This suggests that the strategy of selecting new images by training RL agent may not significantly enhance the efficiency of reinforcement learning in NeRF training.

In contrast, the Self-supervised NeRF Image Selector (SNIS) methodology demonstrated a remarkable capacity to streamline the image selection process using a single-step neural network inference. The experiments showed that when compared to both Baseline 1 and Baseline 2, SNIS consistently achieved superior performance. The stability of SNIS performance, even after a small number of training iterations, highlights its robustness and efficiency. Our experiments also explored the impact of different configurations, such as varying the number of initial images (n0) and the additional images predicted by SNIS (n1). We found that the model achieved its highest performance when n0 was set to 30 and n1 to 120. However, performance declined when n1 was set to 140 and 145, likely due to the inadequate initial training with few images. Conversely, direct training with 150 images predicted by SNIS, without prior training on a smaller set, led to significantly better performance.

Moreover, we evaluated the SNIS performance from scratch, setting n0 to 0 and n1 to N. The results were compelling, as SNIS significantly outperformed random image selection, demonstrating its ability to identify and prioritize optimal camera poses from the outset. This highlights the versatility and robustness of SNIS in optimizing NeRF model training, even in the absence of initial camera data.

To sum up, the reinforcement learning approach exhibited suboptimal performance and required extensive computational resources. The need for a substantial number of iterations and the fixed number of iterations per step resulted in prolonged training times, making this approach less efficient and more resource-intensive compared to SNIS. Future research will delve deeper into the optimization of SNIS parameters and explore its integration with other advanced neural network architectures. Additionally, we plan to

investigate the potential of combining SNIS with other machine learning techniques to further enhance its performance. By continuing to refine and validate the SNIS methodology, we hope to contribute to the development of more efficient and effective NeRF training frameworks, ultimately advancing the state-of-the-art in 3D scene reconstruction and rendering technologies.

References

[1] Aron Schmied, Tobias Fischer, Martin Danelljan, Marc Pollefeys, and Fisher Yu. R3d3: Dense 3d reconstruction of dynamic scenes from multiple cameras. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3216–3226, 2023.

[2] Botao Ye, Sifei Liu, Xueting Li, and Ming-Hsuan Yang. Self-supervised super-plane for neural 3d reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 21415–21424, 2023.

[3] Zehao Yu, Songyou Peng, Michael Niemeyer, Torsten Sattler, and Andreas Geiger. Monosdf: Exploring monocular geometric cues for neural implicit surface reconstruction. *Advances in neural information processing systems*, 35:25018–25032, 2022.

[4] Zhuoqian Yang, Shikai Li, Wayne Wu, and Bo Dai. 3dhumangan: 3d-aware human image generation with 3d pose mapping. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 23008–23019, 2023.

[5] Anna Frühstück, Nikolaos Sarafianos, Yuanlu Xu, Peter Wonka, and Tony Tung. Vive3d: Viewpoint-independent video editing using 3d-aware gans. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4446–4455, 2023.

[6] Yinghao Xu, Sida Peng, Ceyuan Yang, Yujun Shen, and Bolei Zhou. 3d-aware image synthesis via learning structural and textural representations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18430–18439, 2022.

[7] Kai-En Lin, Yen-Chen Lin, Wei-Sheng Lai, Tsung-Yi Lin, Yi-Chang Shih, and Ravi Ramamoorthi. Vision transformer for nerf-based view synthesis from a single input image. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 806–815, 2023.

[8] Jiatao Gu, Alex Trevithick, Kai-En Lin, Joshua M Susskind, Christian Theobalt, Lingjie Liu, and Ravi Ramamoorthi. Nerfdiff: Single-image view synthesis with nerf-guided distillation from 3d-aware diffusion. In *International Conference on Machine Learning*, pages 11808–11826. PMLR, 2023.

[9] Jonathan T Barron, Ben Mildenhall, Dor Verbin, Pratul P Srinivasan, and Peter Hedman. Mip-nerf 360: Unbounded anti-aliased neural radiance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5470–5479, 2022.

[10] Hongrui Cai, Wanquan Feng, Xuetao Feng, Yan Wang, and Juyong Zhang. Neural surface reconstruction of dynamic scenes with monocular rgb-d camera. *Advances in Neural Information Processing Systems*, 35:967–981, 2022.

[11] 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.

[12] Keunhong Park, Utkarsh Sinha, Jonathan T. Barron, Sofien Bouaziz, Dan Goldman, Steven Seitz, and Ricardo Martin-Brualla. Deformable neural radiance fields. *https://arxiv.org/abs/2011.12948*, 2020.

[13] Albert Pumarola, Enric Corona, Gerard Pons-Moll, and Francesc Moreno-Noguer. D-NeRF: Neural radiance fields for dynamic scenes. *https://arxiv.org/abs/2011.13961*, 2020.

[14] Zhengqi Li, Qianqian Wang, Forrester Cole, Richard Tucker, and Noah Snavely. Dynibar: Neural dynamic image-based rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4273–4284, 2023.

[15] Mohammad Shafiei, Sai Bi, Zhengqin Li, Aidas Liaudanskas, Rodrigo Ortiz-Cayon, and Ravi Ramamoorthi. Learning neural transmittance for efficient rendering of reflectance fields, 2021.

[16] Thomas Neff, Pascal Stadlbauer, Mathias Parger, Andreas Kurz, Joerg H. Mueller, Chakravarty R. Alla Chaitanya, Anton S. Kaplanyan, and Markus Steinberger. DONeRF: Towards Real-Time Rendering of Compact Neural Radiance Fields using Depth Oracle Networks. *Computer Graphics Forum*, 40(4), 2021.

[17] Chaojian Li, Sixu Li, Yang Zhao, Wenbo Zhu, and Yingyan Lin. Rt-nerf: Real-time on-device neural radiance fields towards immersive ar/vr rendering. *IEEE/ACM International Conference on Computer-Aided Design (ICCAD 2022)*, 2022.

[18] Junli Cao, Huan Wang, Pavlo Chemerys, Vladislav Shakhrai, Ju Hu, Yun Fu, Denys Makoviichuk, Sergey Tulyakov, and Jian Ren. Real-time neural light field on mobile devices. *arXiv preprint arXiv:2212.08057*, 2022.

[19] Heng Yu, Joel Julin, Zoltan A Milacski, Koichiro Niinuma, and László A Jeni. Dylin: Making light field networks dynamic. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12397–12406, 2023.

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

[21] 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.

[22] Julian Chibane, Aayush Bansal, Verica Lazova, and Gerard Pons-Moll. Stereo radiance fields (srf): Learning view synthesis for sparse views of novel scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7911–7920, 2021.

[23] Yuan Liu, Sida Peng, Lingjie Liu, Qianqian Wang, Peng Wang, Christian Theobalt, Xiaowei Zhou, and Wenping Wang. Neural rays for occlusion-aware image-based

rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7824–7833, 2022.

[24] Shoukang Hu, Kaichen Zhou, Kaiyu Li, Longhui Yu, Lanqing Hong, Tianyang Hu, Zhenguo Li, Gim Hee Lee, and Ziwei Liu. Consistentnerf: Enhancing neural radiance fields with 3d consistency for sparse view synthesis. *arXiv preprint arXiv:2305.11031*, 2023.

[25] Liang Song, Guangming Wang, Jiuming Liu, Zhenyang Fu, Yanzi Miao, et al. Sc-nerf: Self-correcting neural radiance field with sparse views. *arXiv preprint arXiv:2309.05028*, 2023.

[26] Nagabhushan Somraj, Adithyan Karanayil, and Rajiv Soundararajan. Simplenerf: Regularizing sparse input neural radiance fields with simpler solutions. In *SIGGRAPH Asia 2023 Conference Papers*, pages 1–11, 2023.

[27] Zelin Gao, Weichen Dai, and Yu Zhang. Hg3-nerf: Hierarchical geometric, semantic, and photometric guided neural radiance fields for sparse view inputs. *arXiv preprint arXiv:2401.11711*, 2024.

[28] 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.

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

[30] Matthias Teschner, Bruno Heidelberger, Matthias Müller, Danat Pomerantes, and Markus H Gross. Optimized spatial hashing for collision detection of deformable objects. In *Vmv*, volume 3, pages 47–54, 2003.

[31] Hado Van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 30, 2016.