# Exploring and Recreating the Neural Radiance Field (NeRF) Model

Gehna Jain, Rimika Dhara, Yubin Hwang, and Skye Gagnon

***Abstract*—This project aims to recreate the Neural Radiance Field (NeRF) model, as described in the original paper by Mildenhall et al. (2020). The primary objective is to develop the NeRF model from scratch, ensuring faithful adherence to the architecture and methods presented in the original paper. This includes implementing core components such as positional encoding, hierarchical volume sampling, ray marching, and the multilayer perceptron model. Key learning outcomes from this project include understanding and extending theoretical concepts to generate novel views of a world scene, gaining hands-on experience in implementing foundational concepts in computer vision and deep learning, and navigating the practical challenges of training NeRF models, particularly the computational challenges and intricacies involved in optimizing such models for high-quality 3D scene reconstruction. Through this work, we aim to contribute to the understanding of NeRF's application in fields like 3D modeling, virtual reality, and robotics. This report covers the approach we followed while implementing the NeRF model from scratch, the experimental results obtained, and the contributions of each team member.**

***Index Terms*—Neural Radiance Fields, 3D Scene Reconstruction, Photorealistic Rendering, Ray Marching, Computer Vision, Deep Learning, Model Training, Volume Rendering.**

## I. Introduction and Motivation

THE main objective of this project was to recreate the Neural Radiance Field (NeRF) model, as described in the original NeRF paper by Mildenhall et al. (2020). This model has revolutionized 3D scene representation by using deep learning to generate photorealistic 3D images from multiple novel 2D views. NeRF employs volume rendering techniques and utilizes a neural network to map 3D coordinates and viewing directions into RGB color values and volume density, enabling it to generate high-quality images. This method has vast applications in fields such as computer graphics, augmented reality (AR), virtual reality (VR), and robotics, offering detailed 3D reconstructions from a small set of 2D images.

This work is important for us because it allows us to expand our knowledge in the realm of Computer Vision and explore a new topic hands-on. It also addresses the limitations of traditional 3D scene reconstruction methods, which often require large numbers of 2D images or depth maps. NeRF, on the other hand, can generate accurate 3D models even from sparse 2D views. The key motivation for this project was to reproduce the NeRF model from scratch and explore the challenges associated with training and implementing such a model. The rest of the report provides an overview of related work, explains the approach we used, presents our experiments and results, and highlights our individual contributions.

### A. Outline of the Report

This report is organized as follows:

- **Section I: Introduction and Motivation** outlines the problem statement and the incentive behind the project
- **Section II: Related Work** reviews various NeRF-based alternative methods, highlighting their advantages and limitations
- **Section III: Approach** details our methodology, including the datasets, NeRF architecture, and key components
- **Section IV: Experiments and Results** presents the experimental setup, quantitative results (e.g., metrics like PSNR and SSIM), and qualitative evaluations
- **Section V: Contributions** provides a summary of the contributions made by each team member
- **Section VI: Bonus Features** discusses out-of-scope features explored, such as, mesh generation and SSIM metric for training
- **Section VII: Conclusions** summarizes the key findings, discusses limitations, and proposes future directions for NeRF applications

## II. Related Work

### A. Overview of Approaches

Neural Radiance Fields (NeRF) have established themselves as a significant breakthrough in synthesizing novel views of 3D scenes from sparse 2D images. The original NeRF paper by Mildenhall et al. [1] introduced a method that leverages multilayer perceptrons (MLPs) with positional encoding to map spatial coordinates and viewing directions to RGB values and densities. This foundational approach has inspired a multitude of extensions and adaptations aimed at improving efficiency and applicability.

Among these, Sat-NeRF [6] focuses on learning multi-view satellite photogrammetry, enabling the generation of realistic 3D models from satellite images. Another notable work, S3-NeRF [**?**], incorporates shading and shadow modeling to improve 3D reconstructions in single-viewpoint scenarios. The Tiny-NeRf model is a compact version of NeRF with a significantly reduced model size, optimization techniques to reduce model parameters, and downsampled image resolution. These approaches demonstrate the versatility of NeRF in addressing a variety of challenges in computer vision and remote sensing.

### B. Details and Advantages of Similar Methods

Methods that build upon the original NeRF architecture, such as Sat-NeRF and S3-NeRF, emphasize improving realism

and fidelity. Sat-NeRF adapts the NeRF framework to handle transient objects and shadow modeling, achieving more accurate photogrammetry in dynamic environments. S3-NeRF, on the other hand, enhances 3D reconstructions by modeling reflectance and visibility fields, enabling better handling of complex lighting conditions. Tiny-NeRF trades-off high resolution output images with faster inference and training times, thus proving to be effective for applications with limited computational resources.

These approaches share the common advantage of leveraging the fundamental strengths of NeRF, such as its differentiable volume rendering and efficient encoding of spatial and view-dependent details. By introducing domain-specific modifications, these methods extend the utility of NeRF to specialized applications such as urban planning and lunar terrain mapping.

### C. Details and Advantages of Different Methods

Alternative methods diverge from NeRF's reliance on dense sampling and high computational demands. For example, Light Field Networks (LFN) [11] utilize light field representations to approximate radiance fields, significantly reducing the computational overhead while maintaining high-quality rendering. LB-NeRF (Light Bending NeRF) [4] incorporates physics-based modeling to simulate transparent media and refraction, addressing limitations in scenes involving glass or water.

These methods excel in scenarios where computational resources are constrained or where physical accuracy is critical. While they may sacrifice some generality compared to NeRF, their tailored designs enable targeted solutions for unique challenges in 3D scene reconstruction.

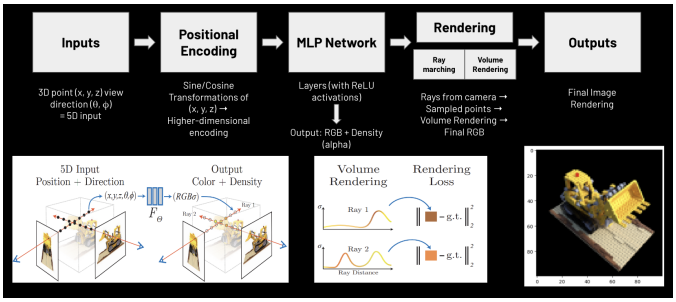
## III. Approach

### A. Problem Definition

The main objective of this project is to implement the Neural Radiance Field (NeRF) model, which allows for photorealistic 3D scene reconstruction from sparse 2D images. NeRF generates accurate 3D representations by learning the interaction of light in the scene from minimal 2D viewpoints. The inputs to the model consist of 5D coordinates that include the 3D position of a point in space $(x, y, z)$ and the viewing direction $(\theta, \phi)$ from the camera. The output of the model is the rendered 3D scene, which is produced by ray marching and volume rendering techniques. The model uses these inputs to compute RGB values and densities for each point along a camera ray, which are then used to generate the final 3D reconstruction.

### B. Network Architecture

The architecture of our NeRF model is built around the concept of encoding 5D positional inputs, followed by processing these through a multi-layer perceptron (MLP) network, and finally using ray marching and volume rendering techniques to generate a 3D rendered image.




**Inputs:** The model takes as input a 5D vector:

- 3D coordinates $(x, y, z)$ that represent the spatial location of points in the scene.
- The view direction $(\theta, \phi)$, which defines the camera's perspective for each 3D point.

**Positional Encoding:** The 5D inputs undergo a positional encoding step, where sine and cosine transformations of the coordinates map them into a higher-dimensional space. This step is crucial as it enables the model to capture high-frequency details, essential for photorealistic rendering.

**MLP Network:** After encoding the 5D inputs, they are passed through an MLP consisting of multiple layers with ReLU activations. The network outputs color (RGB) and density (alpha) values for each sampled point in the 3D space, which represent the appearance and opacity at each point.

**Rendering:** The model uses ray marching to cast rays from the camera and samples points along these rays. These points are then rendered using volume rendering, which accumulates the RGB and density values along the ray to compute the final color. The result is the predicted image, which is compared to the ground truth using a rendering loss function.

### C. Definitions of Mathematical Terms and Methodologies

**Ray Marching:** Ray marching is a technique used to sample points along each ray cast from the camera. This method helps the model determine the appearance and opacity at each point along the ray, crucial for generating realistic 3D reconstructions.

**Volume Rendering:** Volume rendering accumulates color and density values from sampled points along the ray to compute the final pixel color. It simulates the way light interacts with materials in the scene, allowing for the creation of realistic images from sparse views. The expected color $C(r)$ of the ray $r$, can be expressed as an integration of the transfer function, $T(t)$, the density function, $\sigma(r(t))$ and the color component, $c(r(t), d)$ for all the points sampled along the ray.

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

$$\text{where } T(t) = \exp\left(-\int_{t_n}^{t} \sigma(r(s))ds\right)$$

**Loss Function:** The key loss function used in NeRF is the rendering loss, which compares the predicted RGB values to the ground truth. The loss is defined as:

$$L_c = \sum_{r \in R} \|C_r - \hat{C}_r\|_2^2$$

where $C_r$ is the ground truth color, and $\hat{C}_r$ is the predicted color for a given ray $r$. The goal of training is to minimize this loss, improving the model's accuracy in generating realistic 3D reconstructions.

**Stratified Sampling:** Stratified sampling is an important component of NeRF because it allows the model to sample points along rays in a non-uniform way, focusing more on regions with higher density and detail. Instead of sampling uniformly, we break the ray into smaller intervals, and sample in a way that prioritizes regions likely to have more data, improving efficiency. We also perform this stratified sampling along with perturbations to introduce randomness in the data, increasing generalization capabilities. This technique reduces computational complexity while maintaining high rendering quality. It ensures that we focus our computational resources on parts of the scene that are most important for rendering.

**Hierarchical Volume Sampling:** Hierarchical volume sampling builds upon stratified sampling by introducing a two-stage approach with a coarse model and a fine model. The coarse model first gives a rough estimation of the scene, and then, based on these predictions, the fine model samples more densely in areas where higher detail is required. This hierarchical approach enables us to get high-quality results without spending computational resources on unnecessary parts of the scene.

### *D. Datasets and Selection Criteria*

We utilized several datasets for training and testing our NeRF model, each chosen for its ability to provide controlled and diverse conditions to test the model's performance.

**Tiny NeRF Dataset:** This dataset consists of 106 images of a synthetic Lego bulldozer captured from different angles. Each image includes camera poses and focal lengths, which are essential for accurately reconstructing the 3D scene. The dataset is 'tiny' as each image has a resolution of 100x100x4 and is therefore computationally efficient, making it ideal for initial experimentation and model development. Using this smaller dataset allowed for faster training and testing, ensuring that we could iterate on the model quickly without the computational burden of larger datasets.

**Synthetic 360 Dataset:** This dataset was introduced in the NeRF's original paper consisting of 8 world scenes rendered on Blender. We pick the Lego bulldozer dataset which consists of 106 images, each of resolution 800x800x4, captured from different angles. The transforms.json files include metadata about the images, including a 3x4 transformation matrix from the camera to the world coordinates, camera intrinsic parameters such as image width, height, and focal length, and information about near and far bounds for the view frustum. We downsample the images to 400x400x4 resolution, and halve the width, height and focal length associated with each image respectively.

**Cybertruck SketchFab Dataset:** For real-world testing, we used a dataset of 105 images of the Tesla Cybertruck. The images come with camera poses and focal lengths, similar to the Tiny NeRF dataset. This dataset was chosen to explore the model's ability to reconstruct more complex, real-world objects.

The choice of these datasets allows us to quickly iterate on the model while testing both synthetic and real-world objects. The smaller size of the Tiny NeRF dataset ensures faster training and debugging, while the larger dataset ensures high-fidelity and they offer a variety of complexities.

### *E. Evaluation Metrics*

To evaluate the performance of the NeRF model, we used two primary evaluation metrics: Peak Signal to Noise Ratio (PSNR) and the loss throughout training.

**Peak Signal to Noise Ratio (PSNR):** PSNR is a widely used metric to quantify the distortion between the predicted image and the ground truth. It is calculated as:

$$PSNR(I) = 10 \cdot \log_{10}\left(\frac{\text{MAX}(I)^2}{\text{MSE}(I)}\right)$$

where $\text{MSE}(I)$ is the mean squared error between the predicted and true images. A higher PSNR indicates less distortion and better image quality.

**Loss Throughout Iterations:** The loss function used for training NeRF measures the difference between the predicted and ground truth images. The loss decreases as the model improves, capturing more details of the scene with each iteration. This reduction in loss is a key indicator of the model's training progress and convergence.

## IV. Experiments and Results

We evaluated the NeRF model using a variety of datasets and performed several experiments to validate the model's performance. The key results can be summarized as follows:

### *A. Epoch Progression*

**NeRF-Synthetic 360**:
Due to computational power limitations, we were able to run the model on the larger dataset, i.e., NeRF-synthetic 360, for only 5000 iterations/epochs. The model is able to perform well on the test dataset, giving a PSNR value of approximately 23 (see fig. 1). Next, the model is tested with novel camera poses


Fig. 1. Results for test set image using the model trained for 6000 iterations

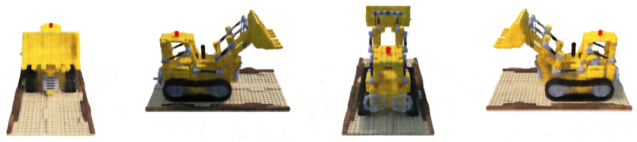


Fig. 2. Novel views synthesized for 4 novel camera poses along the spherical axis

to generate unseen views of the model. (see fig. 1). This was also carried out with 20 different novel camera poses, and a video of the result can be found here.

**Tiny-NeRF**:
The Tiny-NeRF dataset was used to train the model for 10,000 iterations and the results with a test image is demonstrated in fig. 3. The PSNR values reach a peak of approximately 27 during training, and 24 during testing.

**Cybertruck SketchFab**:
The Cybertruck SketchFab dataset was used to train the NeRF model for over 50 iterations, due to time limitations. The result for a test image is depicted in fig. 4.

### *B. Quantitative Metrics*

We plotted the PSNR graph values for the Tiny-NeRF dataset at epochs 0, 10, 5000 and 10,000 respectively. (see fig. 5). A similar visualization was carried out for the model

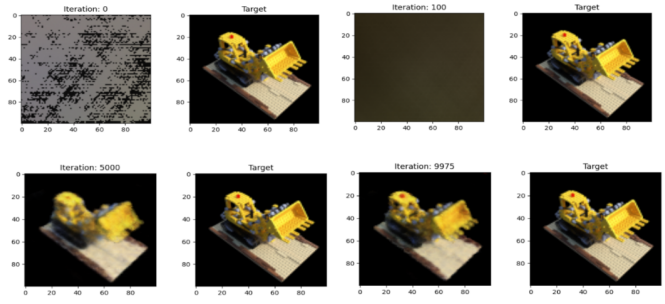



Fig. 3. Results for test set image using the model trained for 10,000 iterations

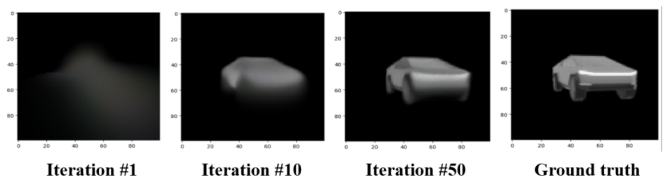



Fig. 4. Results for test set image using model trained for 50 iterations

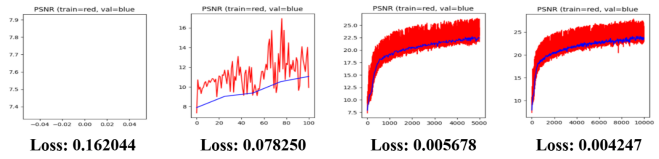



Fig. 5. Train and validation PSNRs, along with loss during iterations 0, 10, 5000 and 10,000 for model trained on Tiny-NeRF dataset.

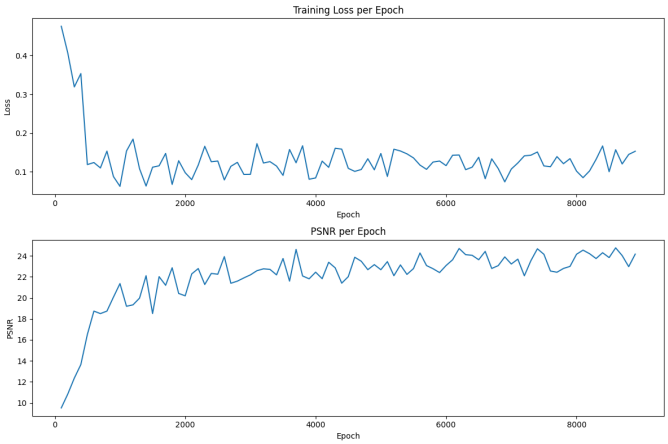



Fig. 6. Train PSNR and loss during every 100th iteration for model trained on NeRF-synthetic 360.

trained on NeRF-synthetic 360 dataset for 6000 iterations and the results are represented in fig. 6

### *C. 360-Degree Video Rendering*

To showcase the NeRF model's ability to generate realistic 3D reconstructions from multiple viewpoints, we rendered a 360-degree video. The process involves capturing frames from novel camera angles, creating a smooth transition around the 3D scene. The following steps were performed to generate the video:

We iterated through 4-20 different camera angles ranging from 0 to 360 degrees using spherical coordinates. For each angle, we computed the camera's pose and cast rays through the scene, generating the corresponding RGB, depth, and accumulated opacity (acc) values using the model's rendering pipeline. This was achieved by calling the 'render_rays()' function, which computes the colors and depth values for each ray cast from the camera to the scene.

Next, we collected each frame and saved it in an array. Afterward, we used the 'imageio' library to compile the frames into a video file. The resulting video is stored as 'video.mp4', with a frame rate of 30 fps. This allows us to visualize how the scene looks from different angles as the camera moves around the object. The two figures (fig. 7 and fig. 8 ) show frames from the video generated by the model trained on Tiny-NeRF and the one trained on NeRF-Synthetic 360. The complete video can be found here.

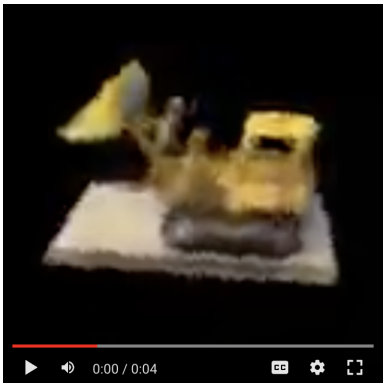



Fig. 7. 360-Degree Video Visualization using the model trained on Tiny-NeRF for 10,000 iterations.

Fig. 8. 360-Degree Video Visualization using the model trained on NeRF-Synthetic 360 for 6000 iterations.

### *D. 3D Mesh Visualization (Bonus)*

In this experiment, we generated 3D meshes from the NeRF model's density and RGB predictions. First, we extracted a 3D volume from the model's output, representing the scene as an occupancy grid. Then, we used the Marching Cubes algorithm to extract the surface of this volume, defining the scene's geometry. The extracted mesh, composed of vertices and triangles, was visualized using the Trimesh library, allowing for an interactive 3D representation of the scene. Additionally, we rendered a 360-degree turntable video of the mesh, providing a dynamic view of the 3D structure.

The generated 3D mesh and the accompanying video demonstrated the model's ability to reconstruct accurate geometric representations of the scene. Visual inspection of the mesh revealed that the NeRF model effectively captured the complex structure and fine details, providing a realistic 3D reconstruction based on the sparse 2D inputs.

## V. Summary of Contributions

- **Gehna Jain**: I was mainly responsible for implementing the NeRF's model architecture, and training and evaluating the performance of the said architecture on the NeRF Synthetic 360 dataset, along with Rimika. I also spent some time playing around with the "fern" images from the real-world NeRF LLFF dataset, but experienced challenges processing the images into the pipeline due to unclear NDC

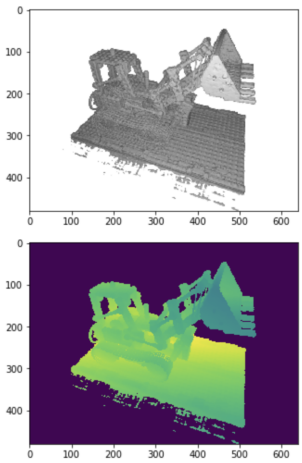



Fig. 9. 3D Mesh Visualization 1

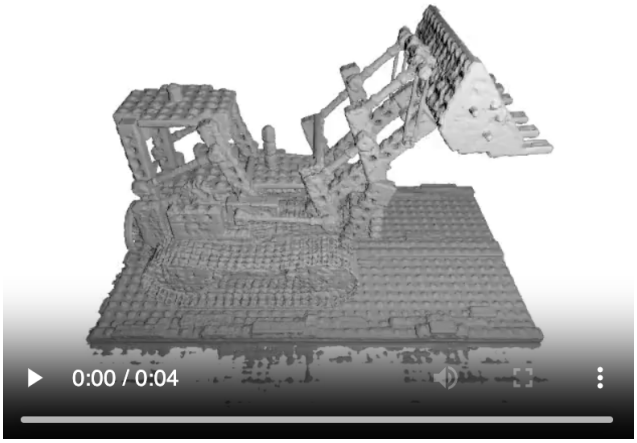



Fig. 10. 3D Mesh Visualization 2 - Rendered Video

transformations.

I began by thoroughly understanding the fundamental concepts used in the paper, and creating the multi-layer perceptron as well as positional encoding modules with Rimika. Next, we worked on the entire pipeline from input to the novel view synthesis. This started with reading image data and associated camera intrinsics and extrinsics information, following which I created novel view camera poses using spherical coordinates to test the model's inference with new unseen data (camera angles). Next, the images were transformed into rays "marching" towards the scene using the get_rays() function, followed by hierarchical and stratified sampling (Fine, coarse models and sample_pdf()), and volumetric rendering. After training, the model uses the novel camera angles to create views!

Apart from technical contributions, I co-authered all the reports and presentations and was one of the primary contributors to all of them. This project had an extremely steep learning curve, and difficulties extending concepts into higher dimensional space. Although, it was computationally challenging to achieve the results described on the paper, our project traces the trajectory to similar results with much

smaller training iterations and compute power. Overall, the project deepened my interest and fascination in computer vision concepts, and enhanced my understanding of NeRFs. My contribution and code can be found here.

- **Rimika Dhara**: I was primarily responsible for implementing the core architecture of the NeRF model and integrating it with the training pipeline alongside Gehna. I began by coding the essential components of the NeRF model, including positional encoding and the MLP network for predicting color and density values. I also worked extensively on the rendering pipeline, implementing ray marching and volume rendering techniques to generate realistic 3D scenes. In terms of data collection, I researched for and prepared the Cybertruck dataset, handling data preprocessing and ensuring that camera poses and focal lengths were properly integrated into the model for accurate ray tracing. I also worked with my teammates to test the model across different data splits and debugged computational inefficiencies during training to improve performance. Additionally, I was responsible for setting up the infrastructure to generate 360-degree videos from the rendered images, ensuring smooth transitions and high-quality visual outputs. I also conducted research on 3D mesh visualization and explored several methods to implement this functionality. I was able to generate a 3D mesh visualization using marching cubes.

Beyond technical contributions, I played a key role in ensuring smooth team collaboration. I led the creation and submission of the intermediate check-in document, providing updates on progress and setting milestones. I also authored a significant portion of the final report and was the primary contributor of the final presentation along with Gehna.

I developed a deep understanding of NeRF's architecture and how it combines machine learning techniques with 3D scene reconstruction. Through the implementation of positional encoding and ray tracing, I gained hands-on experience in simulating light interaction with 3D environments. I also learned about the optimization process in NeRF, where the model iteratively minimizes the loss between predicted and observed data. The process of debugging, adapting code, and overcoming setup issues reinforced my skills in troubleshooting and collaborating effectively in a team environment. These experiences have expanded my technical toolkit in the fields of deep learning, computer vision, and 3D reconstruction. My contribution and code can be found here.

- **Yubin Hwang**: I contributed to this project in both technical and collaborative capacities. On the technical side, I implemented the code for three important evaluation metrics: Peak Signal-to-Noise Ratio (PSNR), Structural Similarity Index Measure (SSIM), and Learned Perceptual Image Patch Similarity (LPIPS). These metrics provide a comprehensive assessment of image quality. Building upon Rimika's existing work, I further integrated the SSIM metric into the model's training process(see the evaluation metrics in the table of content). This modification aimed to improve the model's ability to generate perceptually realistic images. In terms of collaboration, I have been consistently involved in all stages of the project. I actively participated in all group meetings (with the exception of the final presentation day) and contributed to the writing of all project reports, ensuring cohesive and comprehensive documentation of our work.
Through implementing PSNR, SSIM, and LPIPS, I gained a deeper understanding of how image quality is evaluated in the context of NeRF. I learned about the strengths and weaknesses of different metrics, from traditional methods like PSNR to more perceptually aligned metrics like SSIM and learned metrics like LPIPS. This highlighted the importance of choosing appropriate evaluation metrics to assess the visual fidelity of NeRF-generated images. By integrating the SSIM metric into the training loop of the PSNR-based NeRF model, I gained practical experience in how loss functions affect the learning process. This demonstrated how incorporating a perceptual loss like SSIM can guide the model towards generating images with improved structural similarity and visual quality, a key objective in NeRF research.

- **Skye Gagnon**: Much of the work that I accomplished was that of self-learning and writing contributions to the project. As the Professor knows of the significant drawbacks and loss that I've been going through in my personal life much of my work was solitary but never the less significant. `< Code >` I grew my understanding of Neural Radiance Fields by learning and coding its foundational concepts and architecture. Beyond technical work, I authored the majority of first draft of the final report (about 5 pages over a span of a week), contributed to approximately half of the slides in the first presentation and added slides five supporting slides to the final presentation, conducted extensive research paper searches, and reached out to teammates to coordinate meetings. Additionally, I worked to debug and edit my teammate Yubin's code to enable proper training, though I was ultimately unable to get the code to run. I also attempted to set up and run the repository `nerf_pl` to integrate COLMAP with our own dataset but encountered Python versioning issues in Google Colab. I explored using a Miniconda virtual environment to resolve these conflicts, but the approach was unsuccessful. Furthermore, I faced significant library versioning challenges, which required me to rewrite portions of the original `train.py` file to align with updated dependencies and ensure compatibility. These tasks underscored the complexity of managing and troubleshooting modern machine learning pipelines while contributing to the technical robustness of the project.

While learning about NeRFs, I was able to transition from basic 3D reconstruction methods to NeRF, which synthesizes novel views from image datasets using simple MLPs. I was able to write code for NeRF's positional encoding and skip connections in PyTorch, ensuring stability and accuracy through careful attention to initialization practices and activation functions. I also modeled a pinhole camera and simulated ray tracing to calculate ray origins and directions, a foundational step for rendering 2D images from a 3D scene.

Additionally, I explored how NeRF's optimization pipeline parallels solving linear systems like $Ax = B$, where NeRF minimizes the loss between predicted and observed image data through iterative optimization techniques. While $Ax = B$ represents a simpler, linear system, NeRF's reconstruction process similarly seeks to align predicted outputs with known observations, leveraging neural networks to approximate a high-dimensional mapping from input rays to color and density values. This connection helped frame NeRF as a sophisticated extension of traditional mathematical optimization methods for solving inverse problems in computer vision.

## VI. Bonus Features (Optional)

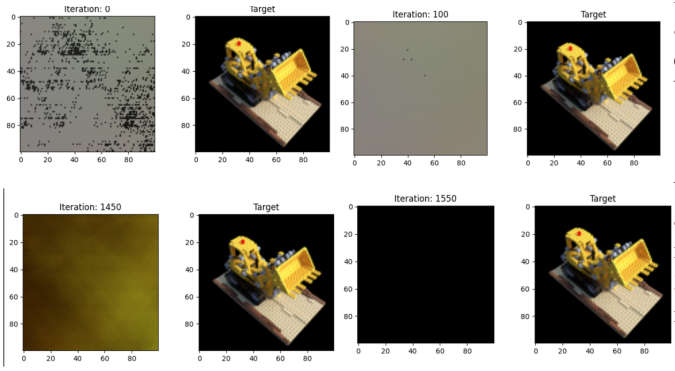
We generated 3D meshes using the NeRF model's density and RGB predictions and the Marching cubes algorithm. This was initially out of scope, and gave us insight into their significance in AR/VR, game development and interactive environments.

Additionally, we updated the train function of the model trained on Tiny-NeRF for 10,000 iterations to use the Structural Similarity Index Measure (SSIM) instead of PSNR for the training model. SSIM metric evaluates the quality of digital images and videos by measuring the similarity between two images. The equation that is implemented to calculate the similarity is the following:

$$SSIM(x, y) = \frac{(2\mu_x\mu_y + C_1)(2\sigma_{xy} + C_2)}{(\mu_x^2\mu_y^2 + C_1)(\sigma_x^2\sigma_y^2 + C_1)}$$

Using SSIM, we expected the new train() function to potentially provide a more accurate and meaningful optimization objective for the neural-radiance field (NeRF) model during the training process.

Unfortunately, the loss of the SSIM calculation was around 0.95 among all iterations, and the generated images did not get closer to the target image throughout the iterations.




These are results of iteration 0, 100, 1450 and 1550. All images generated after the 1550 iteration are the same as 1550, which is a plane black image.

The reason why the train function of the SSIM version does not perform better than the PSNR could be an incorrect loss function, but we did not manage to find it.

## VII. Conclusions

In this report, we explored the implementation and application of Neural Radiance Fields (NeRF) for 3D scene reconstruction and novel view synthesis. Starting with the foundational components of NeRF, including multi-layer perceptrons, positional encoding, and volume rendering, we successfully constructed a framework capable of synthesizing high-quality views of 3D scenes from sparse 2D image datasets.

Our experiments demonstrated the effectiveness of NeRF in achieving detailed reconstructions, as evidenced by quantitative metrics like PSNR and qualitative evaluations of rendered outputs. We also implemented advanced features, such as stratified and hierarchical sampling, to improve rendering quality and efficiency. The results of these experiments highlight the potential of NeRF to address longstanding challenges in computer vision, particularly in scenarios requiring high-fidelity 3D reconstructions with limited input data.

Despite these successes, the project faced several challenges. Restricted computing power on Google Colab limited the efficiency of model training and testing, with the high computational demand of NeRF exacerbating this issue. We also encountered difficulty finding an all-encompassing dataset that wasn't computationally expensive. Additionally, there was a steep learning curve for understanding and implementing NeRF, and data limitations required reliance on synthetic data for evaluating the model. The complexity of the results made it difficult to identify specific areas for improvement, further complicating the refinement process.

Looking ahead, there are several avenues for future work. Enhanced view synthesis could be explored by leveraging NeRF-based novel view synthesis with challenging viewing angles. Optimizing the model for scenes with varying lighting conditions or reflective surfaces could also significantly improve its applicability. Further training the model for additional epochs may lead to improved performance, and continued work on 3D mesh generation and visualization will be crucial for expanding NeRF's capabilities.

Overall, this work demonstrates the practicality and versatility of NeRF for applications in augmented reality, robotics, and photorealistic image synthesis. It also provides valuable insights into the nuances of implementing neural models for 3D scene representation, laying the groundwork for future innovations in this rapidly evolving field.

## Acknowledgment

We would like to thank Prof. Volkan Isler and teaching assistants, Qingyuan Jiang and Mason Hawver, for their guidance and support throughout this project. We are also grateful to the contributions in this field that preceded our work and the research community whose work provided the foundation for our implementation.

## REFERENCES


[1] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng, "NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis," in *Proc. European Conference on Computer Vision (ECCV)*, 2020, pp. 405–421. [Online]. Available: https://arxiv.org/abs/2003.08934

[2] W. Yang, G. Chen, C. Chen, Z. Chen, and K.-Y. K. Wong, "$S^3$-NeRF: Neural Reflectance Field from Shading and Shadow under a Single Viewpoint," *arXiv preprint arXiv:2210.08936*, 2022. [Online]. Available: https://arxiv.org/abs/2210.08936

[3] M. Toschi, R. De Matteo, R. Spezialetti, D. De Gregorio, L. Di Stefano, and S. Salti, "ReLight My NeRF: A Dataset for Novel View Synthesis and Relighting of Real World Objects," *arXiv preprint arXiv:2304.10448*, 2023. [Online]. Available: https://arxiv.org/abs/2304.10448

[4] T. Fujitomi, K. Sakurada, R. Hamaguchi, H. Shishido, M. Onishi, and Y. Kameda, "LB-NERF: Light Bending Neural Radiance Fields for Transparent Medium," in *2022 IEEE International Conference on Image Processing (ICIP)*, Bordeaux, France, 2022, pp. 2142-2146. doi: 10.1109/ICIP46576.2022.9897642

[5] L. Radl, A. Kurz, M. Steiner, and M. Steinberger, "Analyzing the Internals of Neural Radiance Fields," *arXiv preprint arXiv:2306.00696*, 2024. [Online]. Available: https://arxiv.org/abs/2306.00696

[6] R. Marí, G. Facciolo, and T. Ehret, "Sat-NeRF: Learning Multi-View Satellite Photogrammetry With Transient Objects and Shadow Modeling Using RPC Cameras," in *CVPR Workshop Proceedings*, 2022. [Online]. Available: https://centreborelli.github.io/satnerf

[7] M. Hansen, C. Adams, T. Fong, and D. Wettergreen, "Analyzing the Effectiveness of Neural Radiance Fields for Geometric Modeling of Lunar Terrain," in *2024 IEEE Aerospace Conference*, DOI: 10.1109/AERO58975.2024.10521163

[8] M. Partha, S. Gupta, and G. Gao, "Neural City Maps: A Case for 3D Urban Environment Representations Based on Radiance Fields," Stanford University, Technical Report, 2023.

[9] A. Dai, S. Gupta, and G. Gao, "Neural Elevation Models for Terrain Mapping and Path Planning," Stanford University, Technical Report, 2024.

[10] A. S. A. Rabby and C. Zhang, "BeyondPixels: A Comprehensive Review of the Evolution of Neural Radiance Fields," *Journal of ACM*, vol. 37, no. 4, Article 111, Aug. 2023. DOI: https://arxiv.org/abs/2306.03000

[11] V. Sitzmann, S. Rezchikov, W. T. Freeman, J. B. Tenenbaum, and F. Durand, "Light Field Networks: Neural Scene Representations with Single-Evaluation Rendering," *arXiv preprint arXiv:2106.02634*, 2022. [Online]. Available: https://arxiv.org/abs/2106.02634