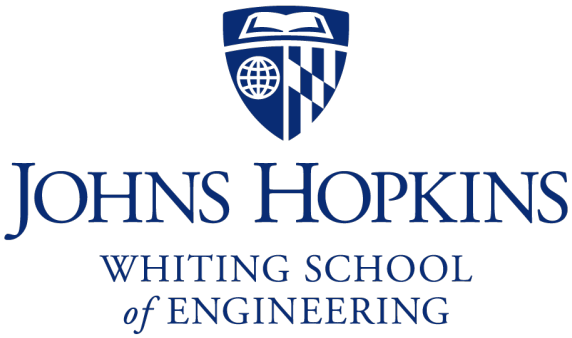
Project 2: NeRF and COLMAP



520.465/665

Machine Perception

Monday, Dec 16th, 2024

Rama Chellappa, PhD. & **Prabhakar Kathirvel**, PhD.

Fall 2024

Seyi R. Afolayan, Sameer Khan, Lavinia Kong

Contents

List of Figures

1 Introduction

Neural radiance fields (NeRFs) represent a breakthrough in 3D scene reconstruction, enabling high-quality novel view synthesis from a set of input images. In this project, we explore NeRF and related deep learning models to understand their practical performance characteristics. Our main objective is to evaluate how varying the number of input images and models affect both the training process and visual output quality. Specifically, we compare two implementations from the Nerfstudio framework - `Nerfacto` (the default model) and `Splatfacto` (a GPU-optimized 3D Gaussian splatting model) - across three different temporal sampling rates. Through these experiments, we aim to provide insights into the trade-offs between computational requirements and reconstruction quality that could guide future applications of these models.

2 Methods

2.1 COLMAP

COLMAP is an open-source Structure-from-Motion (SfM) and Multi-View Stereo (MVS) software that enables 3D reconstruction of a scene from multiple images. In the context of NeRF, it is used as a preprocessing SfM program to take our raw video or image and calculate the extrinsic and intrinsic parameters for each frame/image. These parameters are necessary for NeRF as an input and cannot be run otherwise.

2.2 Nerfacto

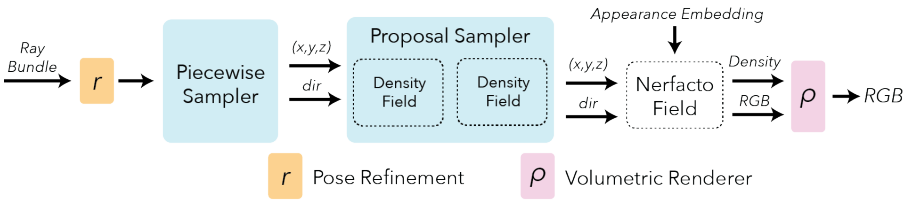


Figure 1: Nerfacto pipeline (Source)

Nerfacto is the default model in Nerfstudio to construct captures of static scenes; it is relatively fast and lightweight, having 6 GB of memory. It is a combination of many published methods, and the full pipeline is described in Figure 1.

The basis of *Nerfacto* is NeRF. NeRF is a deep learning model that aims to solve the plenoptic function using the coordinates (x, y, z) and the view angle (θ, ϕ) as input and outputs the emitted color (R, G, B) and volume density σ. For every 3D location, it seeks to generate all possible views. To render the output, camera rays are marched through the scene to generate 3D point samples. The 3D samples and their corresponding 2D viewing directions are inputs to the neural networks to output a set of colors and densities, which are then rendered into a 2D image. The neural network learns by minimizing the error between each rendered image and each observed image and its corresponding camera pose and intrinsic parameters. This encourages the network to predict a coherent model of

the scene by assigning high volume densities and accurate colors to the locations that contain true scene content. High-resolution scenes because a positional encoding enabled the MLP to represent higher frequency functions and a hierarchical sampling procedure reduces the number of queries needed to accurately represent the scene. Finally, the result is rendered by compositing the color of each pixel as a camera travels through the scene from the desired viewpoint.

On top of NeRF, *Nerfacto* is augmented in several ways. Although the camera poses are already calculated by COLMAP, Nerfacto first refines this estimation by backpropagating the loss gradients to the input pose calculations. Next, a Piecewise sampler is used to produce the initial samples of the scene, with half of the sample being sampled uniformly in proximity to the camera, and the other half are distributed so that the step size increases with each sample. This way, the samples are more densely sampled closer to the camera, while those further away from the camera are still represented.

Subsequent samples are produced by a proposal sampler that consolidates the sample at regions with the most impact to the final render. The density function to the sampler is a small fused MLP (multilayer perceptron) with a hash encoding to provide a balance between accuracy and speed. Two density functions are chained together to the sampler to further consolidate sampling.

Nerfacto uses a small fused MLP with hash encoding to query the scene efficiently; further efficiency is achieved by decreasing the encoding dictionary size and the number of feature levels.

2.3 Splatfacto

Splatfacto is another NeRF method in the Nerfstudio library. The underlying technique of *Splatfacto* is 3D Gaussian splatting. Gaussian splatting is a volume rendering technique that deals with the direct rendering of volume data without converting the data into surface or line primitives by describing the image using rasterized gaussians. The rasterization consists of projecting each gaussian into 2D from the camera perspective, sorting the gaussians by depth. For each pixel, blend the gaussians together front-to-back.

The method uses the 3D point cloud of the scene as input, which can be calculated from SfM programs like COLMAP. Each point is converted into a Gaussian, described by its position, covariance, color, and opacity. The neural network performs stochastic gradient descent and minimizes the error between each rasterized images and ground truth images, and the Gaussian parameters are adjusted based off of the L1 loss and structural similarity index measure. The gaussians are also adjusted or prune if their opacity falls below a certain threshold, or if the gradient is sufficiently large. This procedure creates a dense set of 3D Gaussians that represent the scene as accurately as possible.

The only difference compared to the original paper is that *Splatfacto* utilizes CUDA, making it more time and memory efficient. The *Splatfacto* model also takes 6 GB of memory.

3 Experiment

3.1 Experimental Overview and Data Collection

Our experiments utilized video data captured from *Malone Hall*. From the original video footage, we employed three distinct temporal sampling to generate three different datasets. We evaluated two state-of-the-art NeRF architectures on each temporally sampled dataset discussed in the Methods section. It is imperative to mention that we ran the sampled dataset through COLMAP. The generated output was then ran through the two architectures-`splatfacto` & `nerfacto`. The three different datasets collected are itemized below:

(a) `room_vid_T10_S1`

(b) `room_vid_T20_S1`

(c) `room_vid_T5_S1`

Additionally, the training pipeline was implemented using the Nerfstudio framework with the following configuration:

```
Model: nerfacto/splatfacto
Downscale Factor: 1
Websocket Port: 7007
Data Directory: data/nerfstudio/[dataset_name]
```

3.2 Training Protocol

Each model was trained on all three temporal variations of the room video data to get a good understanding of performance. The training process maintained consistent parameters across the experiments:

(a) Full resolution image processing (`--downscale-factor 1`)

(b) Real-time visualization capabilities through websocket connection (ran through `Xterm`)

(c) Interactive monitoring through Nerfstudio's built-in viewer (Viser).

3.3 Evaluation Metrics

The models were evaluated based on the following metrics:

(a) Training time

(b) COLMAP processing time

(c) Rendering time

(d) Output sharpness and accuracy

4 Results

For each of the six experiments, we captured custom rendered videos and images using nerfstudio and documented the COLMAP processing time and the nerfstudio training time. The original 3000 frame video was temporally downsampled by 5,10, and 20 to create 3 new videos with 600, 300, and 150 frames respectively. We will refer to the 600 frame video as High-frame, the 300 frame video as Medium-frame, and the 150 frame video as Low-frame, for conciseness in the following section. The original video can be found here

4.1 Experiment 1: High-frame video on Nerfacto

The full video can be found here.

4.1.1 Sample Visualization

Figure 2: Sample image from High-frame video trained on nerfacto

4.2 Experiment 2: High-frame video on Splatfacto

The full video can be found here.

4.2.1 Sample Visualization

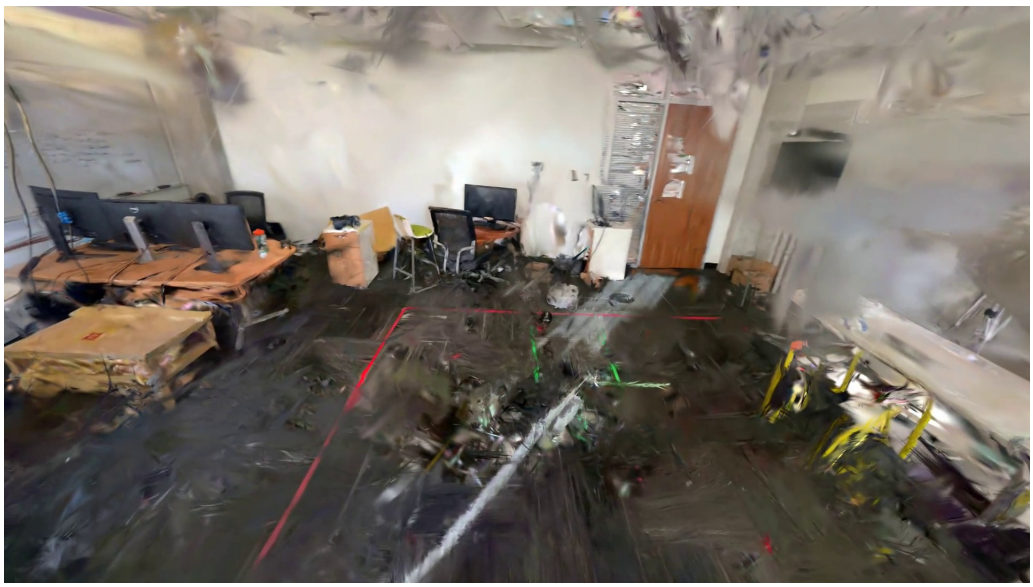


Figure 3: Sample image from High-frame video trained on splatfacto

4.3 Experiment 3: Medium-frame video on Nerfacto

The full video can be found here.

4.3.1 Sample Visualization

Figure 4: Sample image from Medium-frame video trained on nerfacto

4.4 Experiment 4: Medium-frame video on Splatfacto

The full video can be found here.

4.4.1 Sample Visualization

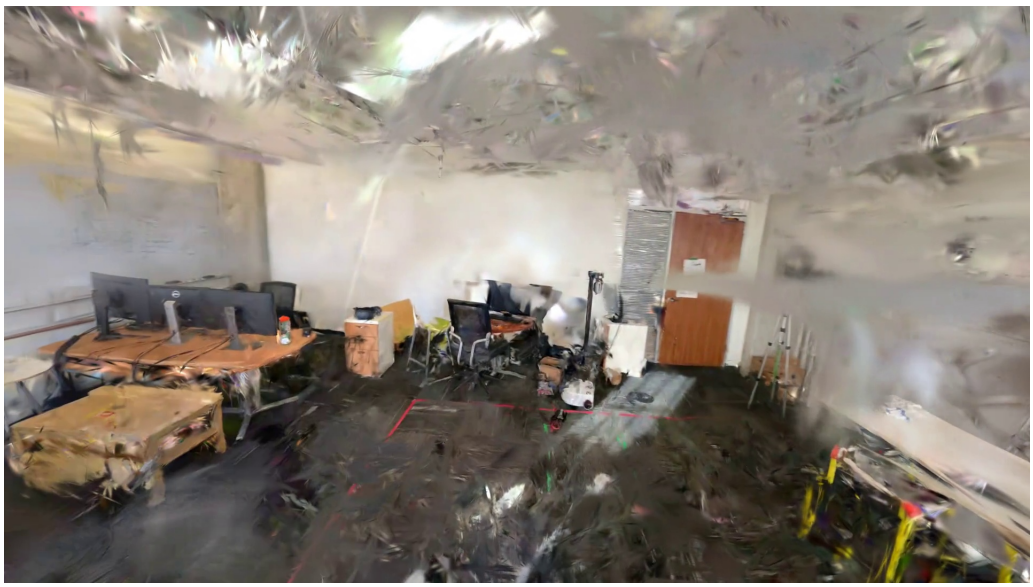


Figure 5: Sample image from Medium-frame video trained on splatfacto

4.5 Experiment 5: Low-frame video on Nerfacto

The full video can be found here.

4.5.1 Sample Visualization

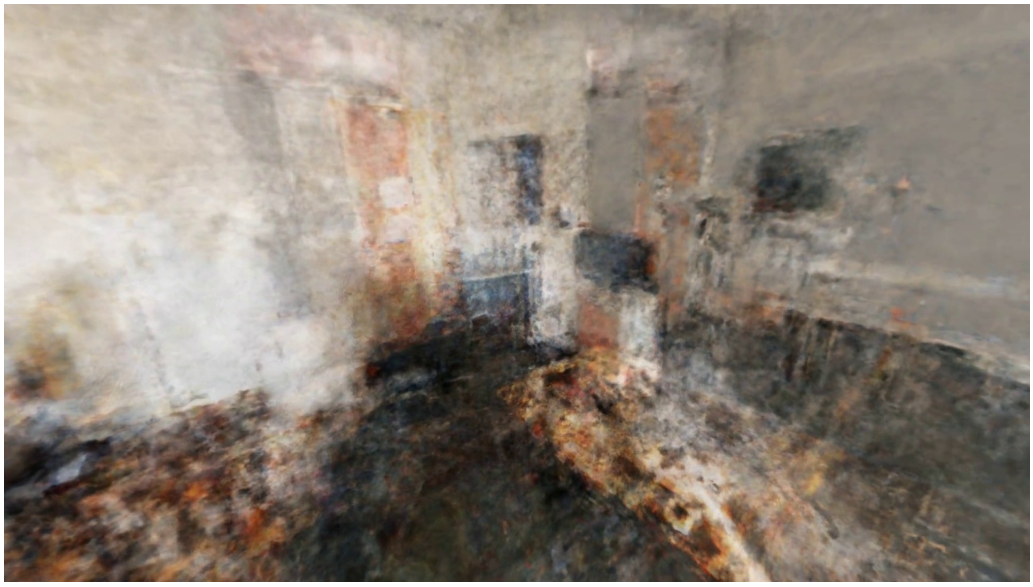


Figure 6: Sample image from Low-frame video trained on nerfacto

4.6 Experiment 6: Low-frame video on Splatfacto

The full video can be found here.

4.6.1 Sample Visualization

Figure 7: Sample image from Low-frame video trained on splatfacto

4.7 Time Metrics

We ran COLMAP and nerfstudio on a compute unit with an NVIDIA RTX 4090. The values from Table 1 are all measured using said compute unit.

Metrics	Exp.1	Exp.2	Exp.3	Exp.4	Exp.5	Exp.6
COLMAP Processing Time (in mins)	30	30	5	5	2	2
Training Time (in mins)	11	12	9	11	9	9
Video Render Time (in mins)	7	0.1	7	0.1	7	0.1

Table 1: Table with evaluation metrics for each experiment

5 Conclusion

The objective of our experiments is to explore the differences in model results when using different numbers of input images and different types of model architectures.

We found that using different numbers of images has a few key outcomes. With larger number of input images to COLMAP and the NeRF model, the output visualization is qualitatively better and closer to the actual video and real life object/scene. Higher number of input images also comes with exponentially longer COLMAP reconstruction times and slightly increased NeRF training times which makes it a tradeoff between quality and time efficiency.

We also tested two distinct Nerfstudio models, *Nerfacto* and *Splatfacto*, with drastically different outputs. Qualitatively, the *Nerfacto* models looked sharper and had more detail with less extraneous errors and artifacts, especially in the Low-frame example in

Experiment 6. The benefit to *Splatfacto* comes with its over 40 times faster rendering speed when compared to *Nerfacto*, which makes using the trained model much quicker and more feasible for real-time applications.

Overall, we believe the Medium-frame video trained on *Nerfacto* had the best qualitative look for the amount of processing and training time taken. Rendering for NeRF models is still slow and if that is important to the project at hand, the Medium-frame video trained on *Splatfacto* is also relatively good when traversing through the trained model.