Analyzing 3D Gaussian Splatting and Neural Radiance Fields: A Comparative Study on Complex Scenes and Sparse Views

Cordell Blanchard, Lakshya Gupta, Shailesh Nanisetty

Abstract—This paper delves into state-of-the-art methods in 3D image reconstruction and rendering, spotlighting Neural Radiance Fields (NeRF) and 3D Gaussian Splatting as transformative methods. While NeRF introduces a neural network for high-fidelity scene reconstruction, its computational demands led to the development of variants like InstantNGP, prioritizing faster processing times. Concurrently, 3D Gaussian Splatting emerges as a fast alternative, employing Gaussian's for efficient scene reconstruction. This study conducts a comparative analysis of Gaussian Splatting and InstantNGP, focusing on their performance in rendering complex scenes with reflective and transparent surfaces, as well as their efficacy in reconstructing high-fidelity images from sparse views. This research contributes valuable insights into which of the two models performs better under complex circumstances with faster training time than current state-of-the-art methods.

1 Introduction

In recent years, the field of 3D image reconstruction and rendering has made significant strides, with groundbreaking techniques like Neural Radiance Fields (NeRF) and 3D Gaussian Splatting emerging as prominent methods. These advancements have substantially enhanced our ability to render complex scenes, especially those involving challenging material types such as reflective and transparent surfaces.

NeRF represents a paradigm shift in high-fidelity 3D scene reconstruction. It employs a deep neural network to encode a volumetric scene function that maps 5D coordinates (including 3D position and 2D viewing direction) to color and density, allowing for detailed rendering of intricate scenes. However, the high computational demands of NeRF models pose significant challenges in terms of resource and time requirements. To address these limitations, variants like InstantNGP have been developed, offering faster processing times while still delivering quality outputs. InstantNGP represents a leap in efficiency, using innovative data structuring and neural network optimizations to enhance speed.

Parallel to these developments, 3D Gaussian Splatting has been gaining attention as a fast and efficient alternative. Distinguished by its use of Gaussian kernels to project 3D points onto a 2D plane, this method has been reported to be up to 50 times faster than traditional NeRF models. Its ability to handle sparse inputs and efficiently reconstruct scenes with limited data points makes it an attractive option for comparison against fast NeRF models like InstantNGP.

This paper aims to undertake a comparative analysis of 3D Gaussian Splatting and InstantNGP, focusing on two critical aspects:

1) Assessing the performance of both models in rendering scenes with reflective and transparent surfaces, challenging scenarios for most 3D reconstruction algorithms.
2) Examining the efficacy of Gaussian Splatting in reconstructing high-fidelity images from sparse views and compare its performance against that of InstantNGP.

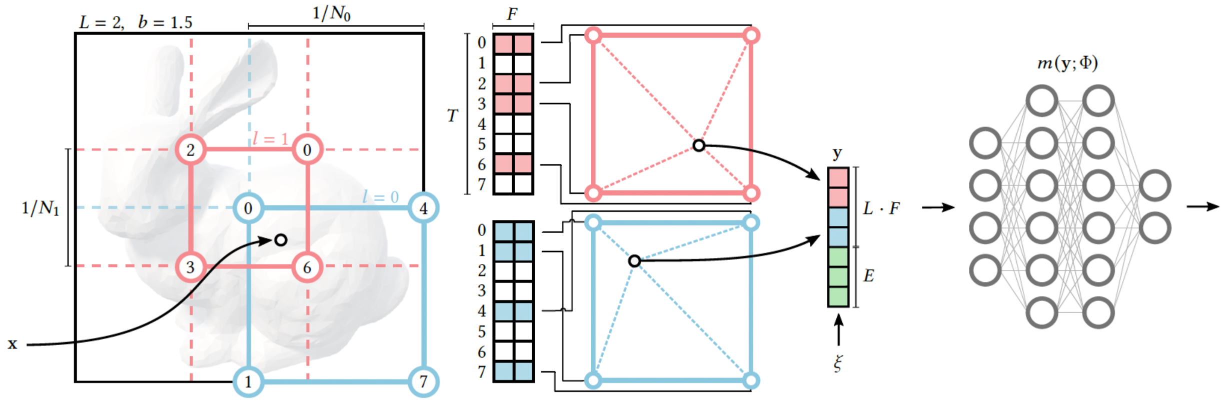
This study is significant as it sheds light on the trade-offs between processing speed and rendering accuracy in advanced 3D reconstruction techniques. By concentrating on specific challenging datasets, it provides valuable insights that could guide future research and practical applications in fields that demand rapid, high-quality 3D rendering.

2 Related Work

2.1 Structure from Motion and Multi-view Stereo

The introduction of Structure-from-Motion (SfM) by Snavely et al. [1] opened up a new realm where a set of photos could be utilized to generate novel views. SfM initially establishes a sparse point cloud during camera calibration, primarily for basic 3D space visualization. Subsequent advancements in multi-view stereo (MVS), as demonstrated by Goesele et al. [2], led to remarkable progress in full 3D reconstruction algorithms. This progress, in turn, facilitated the development of various view synthesis algorithms involving re-projecting and blending input images into a novel view camera, leveraging geometry for guidance.

While these approaches yielded impressive results in many cases, they often struggled to fully recover from unreconstructed regions or instances of ”over-reconstruction” when MVS generated non-existent geometry. In contrast, recent advancements in neural rendering algorithms, exemplified by Tewari et al. [3], have significantly mitigated such artifacts. These neural rendering methods also circumvent the resource-intensive task of storing all input images on the GPU, surpassing traditional techniques in various aspects.



(1) Hashing of voxel vertices **(2)** Lookup **(3)** Linear interpolation **(4)** Concatenation **(5)** Neural network

Fig. 1. Illustration of multiresolution hash encoding in 2D [4]

2.2 Neural Rendering and Radiance Fields

Neural Radiance Fields (NeRFs), as introduced by Mildenhall et al. [5], incorporated importance sampling and positional encoding to enhance rendering quality but at the expense of speed due to a large Multi-Layer Perceptron. The success of NeRF spurred a proliferation of subsequent methods aiming to balance quality and speed, often through the introduction of regularization strategies. The current leading-edge in image quality for synthesizing novel views is represented by Mip-NeRF360, as proposed by Barron et al. [6]. While Mip-NeRF360 achieves exceptional rendering quality, the associated training and rendering times remain exceedingly high.

Recent advancements in methodologies prioritize accelerated training and/or rendering, primarily leveraging three design decisions: employing spatial data structures to store (neural) features that are later interpolated during volumetric ray-marching, utilizing various encodings, and adjusting the Multi-Layer Perceptron (MLP) capacity. For instance, InstantNGP, as presented by Müller et al. [4], exploits a hash grid and an occupancy grid to hasten computation. Additionally, it employs a smaller MLP to depict density and appearance. This method relies on Spherical Harmonics for the direct representation of directional effects.

2.3 Point-Based Techniques

Point-based techniques are adept at efficiently rendering disconnected and unstructured geometry samples, such as point clouds, as demonstrated by Gross and Pfister [7]. 3D Gaussians offer a more versatile representation of scenes, eliminating the necessity for Multi-View Stereo (MVS) geometry and enabling real-time rendering through a tile-based rendering algorithm for the projected Gaussians. This technique is taken advantage of in the 3D Gaussian Splatting Model [8]

3 Methods

3.1 Instant Neural Graphics Primitives (Instant NGP)

The aim of Instant NGP [4] is to enhance approximation quality and accelerate the training speed of a fully connected neural network denoted as m(y;ϕ). The focus lies on optimizing an encoding of its inputs y = enc(x;θ) to achieve superior performance across diverse applications, all while minimizing any notable impact on computational efficiency. The neural network comprises trainable weight parameters ϕ and introduces an additional set of trainable encoding parameters θ. These are arranged into L levels, each containing up to T feature vectors with dimensionality F.

Figure 1 illustrates the steps performed in the multiresolution hash encoding. Every level, depicted in the figure by two instances highlighted in red and blue, operates independently. In a conceptual sense, these levels store feature vectors positioned at the vertices of a grid. The resolution of this grid is selected to follow a geometric progression spanning from the coarsest to the finest resolutions [Nmin,Nmax]:

$$N_l = [N_{min}.b^l] \tag{1}$$

$$b := exp(ln(N_{max}) - ln(N_{min})/L - 1) \tag{2}$$

N_{max} is chosen to match the finest detail in the training data. Due to the large number of levels L, the growth factor is usually small.

The input coordinate x ϵ R^d is scaled by that level's grid resolution before rounding down and up. Each corner is mapped to an entry in the level's respective feature vector array, which has fixed size of atmost T. For coarse levels where a dense grid requires fewer than T parameters, i.e. $(N_l + 1)^d$<=T, this mapping is 1:1. A hash function h: $Z^d - > Z_T$ is used at finer levels,to index into the array, utilizing it as an without explicit collision handling.Gradient-based optimization is used instead to embed relevant sparse details in the array. The neural network m(y;ϕ) is then employed for resolving collisions. The count of trainable encoding parameters θ is consequently O(T), constrained by TLF. A spatial hash function Techsner et al. [9] is used:

$$h(x) = (\bigoplus_{i=1}^{d} x_i \pi_i) modT \tag{3}$$

where $\bigoplus$ denotes the bit-wise XOR operation and π_i are unique, large prime numbers. Lastly, the feature vectors at each corner are d-linearly interpolated according to the relative position of x within its hypercube, i.e. the interpolation

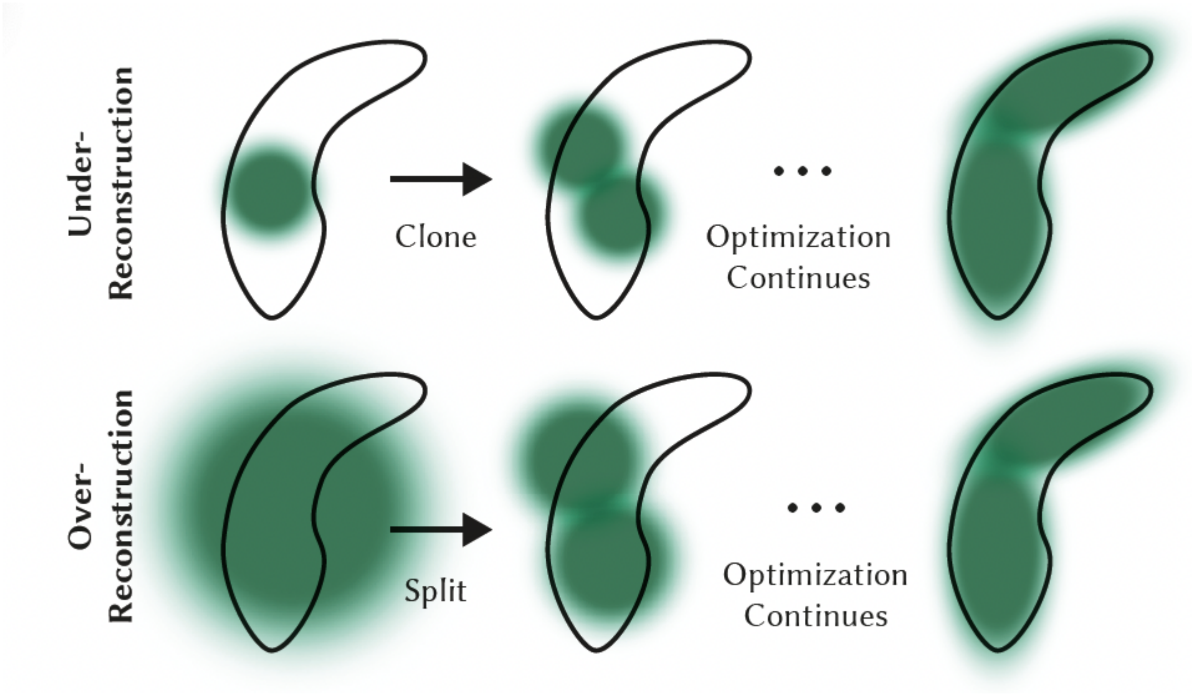


Fig. 2. Adaptive Gaussian Densification Scheme.Top Row (under-reconstruction) and Bottom Row (Over-reconstruction) [8]

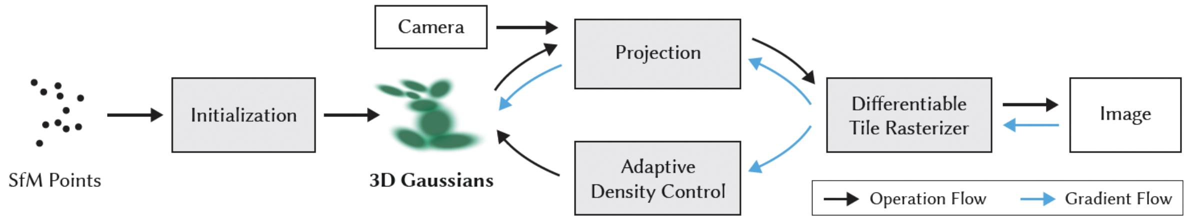


Fig. 3. 3D Gaussian Splatting Pipeline [8]

weight is $w_l = x_l - [x_l]$. Recall that this process takes place independently for each of the L levels. The interpolated feature vectors of each level, as well as auxiliary inputs $\zeta\epsilon R^E$ (such as the encoded view direction and textures in neural radiance caching), are concatenated to produce $y\epsilon R^{LF+E}$, which is the encoded input enc(x;θ) to the MLP m(y;ϕ).

3.2 3D Gaussian Splatting

The input to 3D Gaussian Splatting is a set of images of a static scene, together with the corresponding cameras calibrated by SfM Schönberger and Frahm [10] which produces a sparse point cloud as a side-effect. 3D gaussians are differentiable and can be easily projected to 2d SPLATS allowing fast α-blending for rendering. From these points, a set of 3D Gaussians is created, defined by a position (mean), covariance matrix and opacity α, that allows a very flexible optimization regime. The Gaussians are defined by a full 3D covariance matrix Σ defined in world space Zwicker et al. [11] centred at point (mean) μ:

$$G(x) = e^{-1/2(x)^T \Sigma^{-1}(x)} \tag{4}$$

This Gaussian is multiplied by α in the blending process. Given a viewing transformation W the covariance matrix $\Sigma^{'}$ in camera coordinates is given as follows:

$$\Sigma^{'} = JW\sigma W^T J^T \tag{5}$$

where J is the Jacobian of the affine approximation of the projective transformation. The core approach is the optimization step, which creates a dense set of 3D Gaussians accurately representing the scene for free-view synthesis.

The optimization of the parameters is interleaved with steps that control the density of the Gaussians to better represent the scene.

The loss function is L1 combined with a D-SSIM term:

$$L = (1-\lambda)L1 + \lambda L_{D-SSIM} \tag{6}$$

The adaptive control of the Gaussians populates empty areas. It focuses on regions with missing geometric features (“underreconstruction”), but also in regions where Gaussians cover large areas in the scene (which often correspond to “over-reconstruction”).

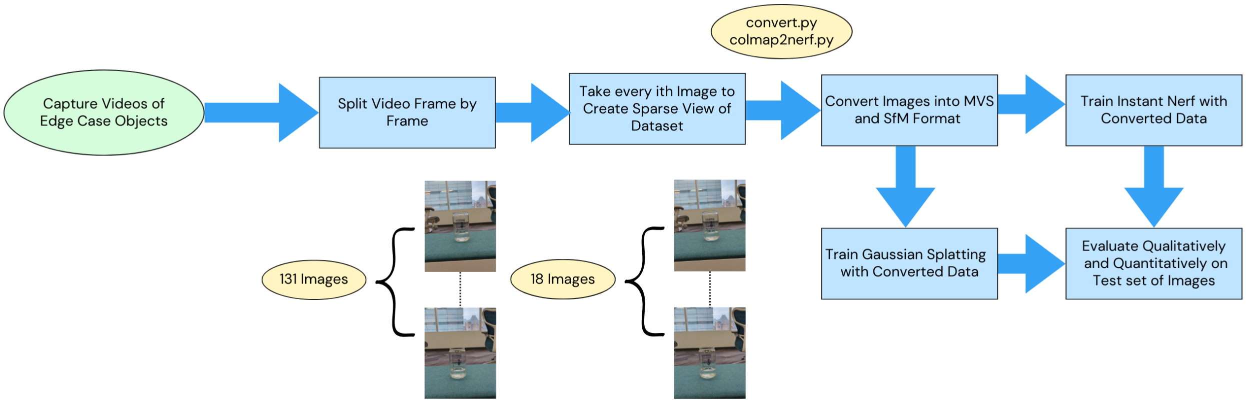

convert.py
colmap2nerf.py
Capture Videos of Edge Case Objects
Split Video Frame by Frame
Take every ith Image to Create Sparse View of Dataset
Convert Images into MVS and SfM Format
Train Instant Nerf with Converted Data
Train Gaussian Splatting with Converted Data
Evaluate Qualitatively and Quantitatively on Test set of Images
131 Images
18 Images

Fig. 4. Sparse View Experiment Pipeline

To achieve fast overall rendering and fast sorting to allow approximate α-blending, a tile-based rasterizer for Gaussian Splats is employed inspired by software rasterization techniques Lassner and Zollhofer [12].This is done to pre-sort primitives for an entire image at a time, avoiding the expense of sorting per pixel that hindered previous α-blending solutions Kopanas et al. [13]. The rapid rasterizer enables effective backpropagation across numerous blended Gaussians with minimal additional memory usage, demanding only a consistent per-pixel overhead.

4 Experiments

4.1 Data Generation and Preprocessing

In order to create our datasets for both the reflective object and the transparent object, we captured 360-degree videos to obtain a complete view of each object. As depicted in Figure 4, we then developed a script to split the videos frame by frame. This resulted in a total of 182 and 132 views for the reflective and transparent datasets respectively.

One of the main objectives of our analysis is to evaluate how 3D Gaussian Splatting handles Sparse View inputs in comparison to Instant NGP. Consequently, we generated two sets of sparse views for each dataset by excluding every i^{th} image from the full dataset, reducing the number of views while preserving the 360-degree perspective. For the reflective dataset, we created two sparse view datasets containing 61 and 31 views, while for the transparent dataset, we crafted sparse view datasets with 58 and 18 views.

Once all the datasets for comparison were created, they needed to be converted into two formats for 3D Gaussian Splatting and Instant NGP. For Gaussian Splatting, we had to create calibrated cameras with Structure-from-Motion (SfM) and initialize the set of 3D Gaussians with a sparse point cloud produced from the SfM process [8]. This process was carried out using COLMAP [14] and the Gaussian Splatting source code. Specifically, we had to use the provided function named convert.py to adapt our custom datasets to the correct format. This data was then used to train the 3D Gaussian Splatting Model.

Concerning Instant NGP, we again utilized COLMAP. This time to recover the camera poses and other intrinsic parameters of the input images of our datasets. To accomplish this, we used Instant NGP's script called colmap2nerf.py. The converted datasets were then used to train Instant NGP's model.

4.2 Experiment Design

We trained both the 3D Gaussian Splatting [8] and Instant NGP [4] models from scratch on one Tesla T4 GPU. The implementation for both models was provided by the respective researchers on GitHub. When training the models, we used their default hyperparameters. The implementation of the Gaussian Splatting model withheld every 8th image in the input dataset for evaluating metrics on novel view synthesis. Instant NGP did not do this, so we modified the source code in instant-ngp/scripts/run.py to withhold every 8th image in that dataset for evaluation of metrics as well. We also added the calculation of the LPIPS metrics, as it was not implemented in the source code. This was done to provide a systematic and proper approach to comparing the two methods. All metrics represent the average of the withheld test set mentioned above. For Gaussian Splatting, we trained for 10,000 iterations for all our datasets, and for Instant NGP, we used 50,000 and 30,000 train iterations on the transparent and reflective datasets, respectively. This was done to assess how Instant NGP would perform on our complex datasets given a similar amount of training time as Gaussian Splatting.

5 Results

5.1 Evaluation and Analysis

After training was finished, we took the averaged quantitative metrics on the test sets and an example render from the test set, so that we could analyze the result as shown in Table 1, Table 2, Figure 5 and Figure 6. It is very important to note the number of input views shown in the results are the training views. This means they are the total set of views with every 8th image withheld for the test set. This is why the number of views is smaller than initially described in section 4.1 as that was the total number of images in the whole set. We use the standard PSNR, L-PIPS, and SSIM accuracy metrics that are commonly used in 3D scene reconstruction literature.

The overall results depicted in Tables 1 and 2, indicate a general decrease in quality for both 3D Gaussian Splatting and Instant NGP, as shown in the metrics. It is noteworthy

TABLE 1
Quantitative Comparison on Reflective Test Set Images

Method	3D Gaussian Splatting			Instant NGP		
Train Views	159	53	27	182	53	27
Train Time (h:m)	0:29	0:27	0:16	0:30	0:27	0:26
Train Iterations	10k	10k	10k	30k	30k	30k
PSNR	31.23	25.86	19.06	16.53	13.68	8.98
LPIPS	0.086	0.149	0.259	0.342	0.518	0.629
SSIM	0.959	0.907	0.805	0.672	0.599	0.310

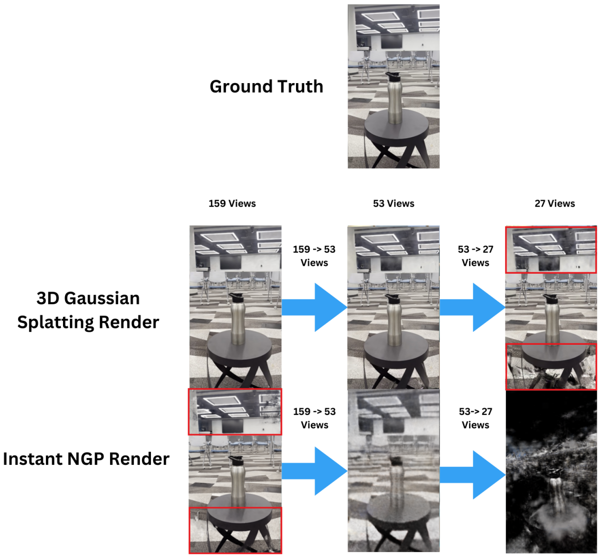


Fig. 5. Qualitative Comparison on Reflective Test Set Image

TABLE 2
Quantitative Comparison on Transparent Test Set Images

Method	3D Gaussian Splatting			Instant NGP		
Train Views	114	50	15	132	50	15
Train Time (h:m)	1:01	0:54	0:52	0:48	0:47	0:44
Train Iterations	10k	10k	10k	50k	50k	50k
PSNR	28.32	26.56	22.88	15.35	14.42	15.01
LPIPS	0.247	0.271	0.323	0.428	0.489	0.474
SSIM	0.886	0.850	0.801	0.522	0.463	0.512

that Table 2 reveals a slight difference where the metrics are superior at 15 views compared to 50 views for Instant NGP. This discrepancy may be attributed to the smaller number of images in the test set for the 15-view experiment. However, 3D Gaussian Splatting consistently outperforms Instant NGP in all of the experiments. One of the reasons behind this is because Instant NGP excels in scenes with high geometric detail but sacrifices quality when handling complex datasets with reflections or transparent objects. This trade-off is associated with the much smaller MLP required in Instant NGP to achieve the training speedup, as discussed in [4]. The findings underscore that 3D Gaussian Splatting exhibits greater potential for delivering high-quality results with fast training time, even in the face of complex datasets and sparse views. This aspect stands in contrast to the compromise between quick training and quality for NeRF models such as Instant NGP.

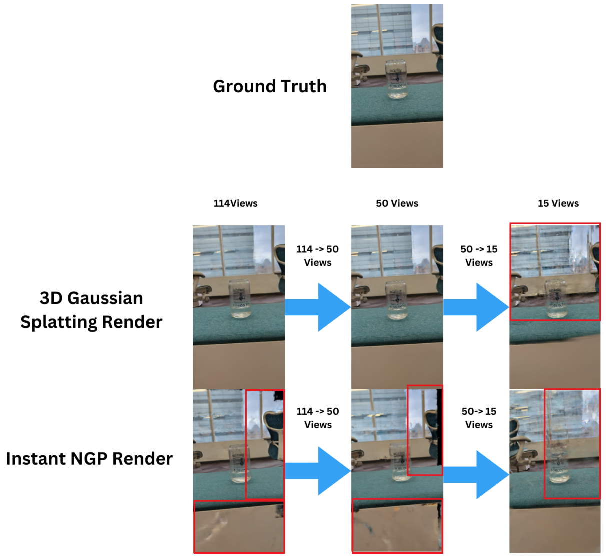


Fig. 6. Qualitative Comparison on Transparent Test Set Image

In Figures 5 and 6, we highlight some of the areas where both models seem to struggle the most. In Figure 6, you can observe that Gaussian Splatting does not obviously fail anywhere until we reach the 15-view experiment, where it is evident that it encounters challenges with the background in the top half of the photo. This may be attributed to insufficient information to generate a proper novel view of the background at that angle or the complexity of the scene in the background. Turning to the transparent qualitative comparison for Instant NGP, we observe that even with a full view the model encounters difficulty rendering a high-quality image. In all cases, it struggles to create the bottom half and the right half of the image. Similar to Gaussian Splatting, it also faces challenges in rendering the background. The Instant NGP model also struggles at creating a novel view, providing a render not at the same exact angle as the ground truth, which was verified to be what the model was trying to render. This may be due to the complexity of the scene or instant NGP struggles more with novel view synthesis than 3D Gaussian Splatting. In Figure 5, 3D Gaussian Splatting excels at rendering the reflective object in all view cases but encounters challenges with more of the background and the bottom of the image. This difficulty may be attributed to the non-uniformity and curvature of the stool's legs shown in the bottom half of the reflective image set. Instant NGP performs extremely poorly in Figure 5, which we discuss in more detail in the next section of our paper.

6 Discussion

One of the intriguing aspects of this study was the differential performance of InstantNGP NeRF on reflective versus transparent surfaces. The relatively poorer performance on reflective surfaces could be attributed to the inherent complexity in modeling specular reflections. Reflective surfaces often require a more nuanced understanding of light interaction, which can be challenging for NeRF-based models like

InstantNGP to capture accurately, particularly under sparse view conditions. In contrast, transparent materials, while complex, may present less variability in light interaction, allowing NeRF models to approximate their properties with greater accuracy even with limited data.

The compute resources employed in this study played a crucial role in our experimental setup and outcomes. The initial use of an i5 4th gen processor, 8 GB RAM, and an Nvidia MX 150 GPU for running COLMAP for SfM (Structure from Motion) and MVS (Multi-View Stereo) set certain limitations, particularly in terms of pre-processing speed. This hardware configuration, while adequate for basic processing, might not fully exploit the potential of more advanced NeRF models. However, the subsequent use of Google Colab's Tesla T4 GPU for training provided significant computational power, enabling more robust training of the models. With the focus on comparing the model's performances against each other, the pre-processing time was not highlighted in the Results section, which helped in a fairer comparison.

The superior performance of Gaussian Splatting in sparse view scenarios, particularly in comparison to InstantNGP, can be attributed to its operational methodology. Gaussian Splatting employs techniques like leveraging anisotropic covariance in the optimization of 3D Gaussians and a fast visibility-aware rendering algorithm. These methods facilitate more accurate scene representation and faster rendering, which are crucial in sparse view conditions. The anisotropic covariance optimization helps in better capturing the scene's details, even with fewer data points. The visibility-aware rendering algorithm supports this by accelerating both the training process and real-time rendering, further enhancing the performance of Gaussian Splatting in scenarios with limited view availability.

6.1 Future Work

In future research, it would be valuable to focus on enhancing the computational efficiency of Gaussian Splatting, aiming to further reduce the training time. This could involve optimizing the current PyTorch-based optimizer and refining GPU usage for more efficient processing. Additionally, experimenting with the 3D Gaussian representation itself presents an interesting avenue. Exploring different distribution types, such as skewed distributions, could potentially improve scene representation, especially in challenging scenarios like sparse views. Finally, a comparative analysis of a sparse view specialized variant of Gaussian Splatting and Sparse-NeRF, would provide deeper insights into the efficacy of various approaches when dealing with limited data. These areas of investigation could significantly contribute to the advancement of 3D rendering techniques, particularly in optimizing performance under constrained conditions.

7 Conclusion

This comparative study of 3D Gaussian Splatting and InstantNGP reveals that Gaussian Splatting significantly outperforms InstantNGP in rendering sparse views, especially in scenarios involving reflective and transparent surfaces. It maintains higher image quality, as evidenced by superior PSNR values, within the same training duration. Meanwhile, InstantNGP demonstrates notable challenges with reflective surfaces under sparse view conditions. These findings highlight the potential of Gaussian Splatting for efficient and high-quality 3D rendering and underscore areas for improvement in NeRF-based models like InstantNGP.

References

[1] S. M. S. Noah Snavely and R. Szeliski, "Photo tourism: exploring photo collections in 3d," *SIGGRAPH*, 2006.

[2] B. C. H. H. Michael Goesele, Noah Snavely and S. M. Seitz., "Multi-view stereo for community photo collections." *ICCV*, 2007.

[3] B. M. P. S. E. T. W. C. L. V. S. R. M.-B. Ayush Tewari, Justus Thies and S. Lombardi, *Advances in neural rendering*. Wiley Online Library, 703-735, 2022.

[4] T. Müller, A. Evans, C. Schied, and A. Keller, "Instant neural graphics primitives with a multiresolution hash encoding," *ACM Trans. Graph.*, vol. 41, no. 4, pp. 102:1–102:15, Jul. 2022. [Online]. Available: https://doi.org/10.1145/3528223.3530127

[5] 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," 2020.

[6] D. V. P. P. S. Jonathan T. Barron, BenMildenhall and P. Hedman., "Mip-nerf 360: Unbounded anti-aliased neural radiance fields." *CVPR*, 2022.

[7] M. Gross and H. E. Pfister, *Point-based graphics*. Elsevier, 2011.

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

[9] M. M. D. P. Matthias Teschner, Bruno Heidelberger and M. Gross, "Optimized spatial hashing for collision detection of deformable objects," in *In Proceedings of VMV'03, Munich, Germany. 47–54*, 2003.

[10] J. L. Schönberger and J.-M. Frahm, "Structure-from-motion revisited," *In Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[11] J. V. B. Matthias Zwicker, Hanspeter Pfister and M. Gross, "Ewa volume splatting." in *Visualization, 2001. VIS'01. IEEE, 29–538.*, 2001.

[12] C. Lassner and M. Zollhofer, "Pulsar: Efficient sphere-based neural rendering," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). 1440–1449*, 2021.

[13] T. L. Georgios Kopanas, Julien Philip and G. Drettakis, "Point-based neural rendering with per-view optimization," *Computer Graphics Forum*, vol. 40, no. 4, pp. 29–43, 2021. [Online]. Available: https://doi.org/10.1111/cgf.14339

[14] J. L. Schönberger, E. Zheng, J.-M. Frahm, and M. Pollefeys, "Pixelwise view selection for unstructured multi-view stereo," in *Computer Vision – ECCV 2016*, B. Leibe, J. Matas, N. Sebe, and M. Welling, Eds. Cham: Springer International Publishing, 2016, pp. 501–518.