https://doi.org/10.2352/EI.2025.37.15.AVM-114

A Comparative Study of NeRF and 3D Gaussian Splatting for Automotive and Edge Applications

Mary Raymond [1] ***Ganesh Sistu*** [1]
Valeo, Tuam, Co.Galway, Ireland [1]

Abstract

This paper presents a comparative study of Neural Radiance Fields (NeRF) and 3D Gaussian Splatting (3DGS) models within the context of automotive and edge applications. Both models demonstrate potential for novel view synthesis but encounter challenges related to real-time rendering, memory limitations, and adapting to changing scenes. We assess their performance across key metrics, including rendering rate, training time, memory usage, image quality for novel viewpoints, and compatibility with fisheye data. While neither model fully meets all automotive requirements, this study identifies the gaps that need to be addressed for each model to achieve broader applicability in these environments.

Introduction

Novel 3D scene reconstruction models, such as NeRF (Neural Radiance Fields) [5] and 3DGS (3D Gaussian Splatting) [6], have shown significant promise in automotive and edge applications due to their ability to render novel views from trained scenes. However, applying these models in such contexts presents several key challenges, as discussed below:

- **Real-time Rendering**: Many automotive applications such as parking assistance systems require real-time rendering of novel views to ensure seamless user experience. These systems rely on a high *model rendering rate*, defined as the speed at which a trained model can generate images from new viewpoints, .
- **On-vehicle Rendering**: In automotive systems, novel views often need to be rendered directly on embedded hardware within the vehicle. This introduces the challenge of *model memory requirements*, i.e., the amount of memory necessary to load and operate the model on specific hardware platforms.
- **Changing Scenes**: As vehicles navigate through dynamic environments, the surrounding scenes change continuously. Therefore, the 3D reconstruction model must be capable of retraining/adapting on new scene data, emphasizing the importance of *training time*. If training occurs on the vehicle's embedded hardware, the *training data memory requirements* also become a critical consideration.
- **Novel View Rendering**: In many automotive scenarios, the required novel viewpoints may lie outside the range of the original dataset, such as a bird's-eye view or a viewpoint not mounted on the vehicle. This requires careful consideration of *model rendering quality* for both *validation datasets* and novel viewpoints beyond the dataset's original range.
- **Fisheye Cameras**: Vehicles and robotic systems frequently utilize fisheye cameras due to their wide field of view. However, fisheye lenses introduce significant optical distortion and add complexity to the camera model, raising the question of how effectively 3D models can accommodate *fisheye camera inputs*.

This study evaluates and compares NeRF and 3DGS models based on these criteria to determine their suitability for automotive and edge applications. Additionally, it highlights the current limitations that need to be addressed for these models to fully realize their potential in these contexts. While both NeRF and 3DGS learn 3D scene representations from images and associated camera poses, they differ fundamentally in how they represent and train these scenes. The following section provides a brief comparison of these key differences as they pertain to this study.

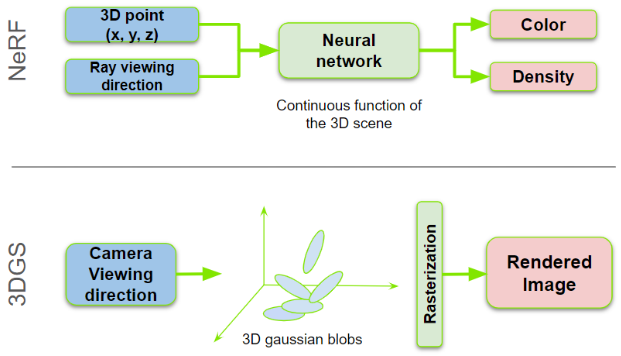

Figure 1: NeRF vs 3DGS model represenation

NeRF	3DGS
- Uses a neural network model to learn a continuous function that represents the density and color of every point in space.	- Uses a number of Gaussian blobs (millions) placed throughout the scene to represent color and density.
- Is an implicit representation of the 3D scene.	- Is an explicit representation of the 3D scene.
- Is a continuous representation of the space.	- Is a discrete representation of the space.
- Ground truth used for training NeRF is the RGB pixel value.	- Ground truth used for training 3DGS is the undistorted image.

Table 1: Brief comparison between NeRF and 3DGS

Prior Art

NeRF (Neural Radiance Fields)[5]: Introduced in 2020 by Mildenhall et al., NeRF uses neural networks to learn continuous volumetric scene representations. It maps 3D coordinates and viewing directions to RGB colors and densities, enabling high-quality novel view synthesis. **3DGS (3D Gaussian Splatting)**[6]: Proposed in 2023 by Bernhard Kerbl et al., 3DGS accelerates rendering by representing scenes with discrete Gaussian blobs. It offers greater efficiency in real-time rendering by using explicit scene representation and rasterization techniques.

Since the original NeRF paper, numerous improvements have been made to enhance NeRF's rendering time. One notable development is **Instant-NGP** [7](2022), by Thomas Müller et al., which achieved real-time rendering on NVIDIA GPUs. A key feature of Instant-NGP is its *fully fused MLP*, a single-kernel CUDA-optimized implementation that minimizes the time required for transferring data between neural network layers. However, while this approach is effective on NVIDIA GPUs, it lacks portability for other embedded platforms used in automotive applications. Another improvement is **FastNeRF**[9] (2021), by Stephan J. Garbin et al., which precomputes a cache after training the NeRF model to render novel views efficiently. While FastNeRF achieves real-time performance, the cache size required (e.g., 7GB for n=512) is impractical for embedded systems, limiting its use in applications with memory constraints.

Although significant advancements have been made in NeRF, direct comparisons between NeRF and 3DGS for 3D reconstruction remain relatively scarce. These comparisons are crucial for understanding the trade-offs between the two technologies, particularly in real-world applications such as automotive and robotics. One recent comparative study, *"Evaluating Modern Approaches in 3D Scene Reconstruction: NeRF vs Gaussian-Based Methods"* [10](2024) by Yiming Zhou et al., contrasts NeRF and Gaussian Splatting within SLAM systems. The study evaluates mapping accuracy, localization precision, and real-time adaptability but does not address the unique challenges faced in **automotive and edge applications**, such as memory limitations, on-target training, and handling fisheye data. This leaves a gap for more application-specific research, which this paper aims to address.

Methodology

This study utilizes the *Nerfstudio* framework [1] as the foundation for experiments, extending it where necessary to incorporate additional metrics. The *Nerfacto* model [2], with its PyTorch implementation, is used as the NeRF-based model, while the *Splatfacto* model [3] serves as the 3DGS-based model. The datasets employed include the *Nerfstudio* dataset [1] for rectilinear images and the open-source *KITTI 360* automotive dataset [4] for outdoor morning scenes for fisheye images use case. The study encompasses multiple experiments designed to comprehensively analyze and compare NeRF-based and 3DGS-based solutions in real-world automotive and edge computing scenarios. The comparison focuses on capturing key metrics for each model, as outlined below:

1. **Model Training Time:** Measured over 30,000 iterations using the Nerfstudio training log, with time recorded in minutes.
2. **Model Rendering Rate:** Assessed for the trained models via the Nerfstudio evaluation script, with rendering rate measured in frames per second (FPS). The resolution of the rendered images is 270×480 (portrait) and 480×270 (landscape)
3. **Model Memory Requirements:** Monitored at various checkpoints, with memory usage calculated based on the number of parameters in each model, assuming 4-byte floating points. The metric is recorded in megabytes (MB).
4. **Training Data Memory Requirements:** Evaluated for the rectilinear dataset, assessing the memory footprint for both Nerfacto and Splatfacto training data samples, captured in megabytes (MB).
5. **Validation Subset Image Quality:** Model rendering quality is evaluated on a validation subset using Structural Similarity Index (SSIM) [11], Learned Perceptual Image Patch Similarity (LPIPS) [12], and PSNR metrics.
6. **Novel Viewpoint Rendering Quality:** Analyzed for viewpoints outside the training dataset range. For the rectilinear dataset, this involves a virtual camera path slightly elevated from the training path. For the fisheye automotive trace, a virtual camera path simulates a front camera view. The Fréchet Inception Distance (FID) [8] [13] metric is used for quality measurement.
7. **Use of Fisheye Dataset:** Due to the rectilinear differentiable renderer used in 3DGS training, undistorted images are required for training the Splatfacto model. For the fisheye use case, images are undistorted before being processed with Splatfacto, while the original fisheye images are used in the NeRF-based model (Nerfacto), utilizing the OpenCV fisheye camera model for ray casting. Image quality for both validation and novel view rendering is assessed using SSIM, LPIPS, and PSNR for the validation subset, with FID applied to novel viewpoint rendering.

Experiment Setup

- **Software:** Nerfstudio framework [1], Git revision 0e889f7, with additional metrics and KITTI 360 Dataset [4] integrated for this study.
- **Hardware:** PC: Lenovo ThinkPad 21FBS1MX0A, GPU: NVIDIA RTX 3500 Ada Generation Laptop GPU (12 GB RAM).

Experiment Configuration

- **Nerfacto Model Configuration:** Default configuration with the following modifications:
  - Image resolution scale = 0.5
  - Model implementation = PyTorch
  - Number of rays per chunk for evaluation = 24,576
  - Number of saves between iterations = 5,000
- **Splatfacto Model Configuration:** Default configuration with the following modification:
  - Image resolution scale = 0.5
  - Number of saves between iterations = 5,000

Datasets

- **Nerfstudio Dataset [1]:** For the rectilinear use case, the used scenes include *poster*, *library*, *redwoods2*, *bww en-*

trance, and *Egypt*.

- **KITTI 360 Dataset [4]:** For the fisheye use case, the specific trace used is *2013_05_28_drive_0000_sync*, with camera IDs 2 and 3 (two side fisheye cameras). The dataset includes 200 frames, starting at frame 950 and ending at frame 1250.

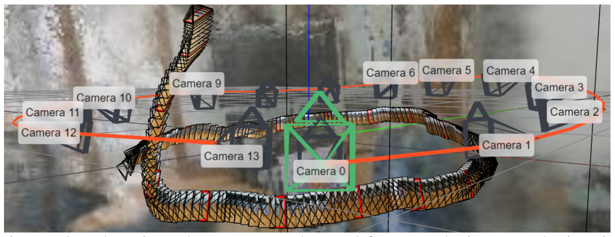

Figure 2: The virtual camera path used for novel view rendering in comparison with the training data for the poster rectilinear trace.

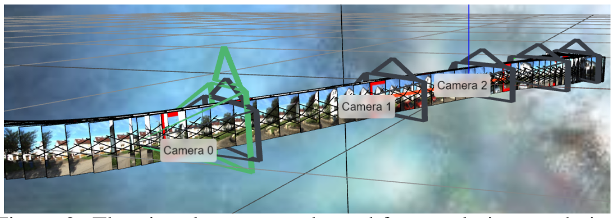

Figure 3: The virtual camera path used for novel view rendering in comparison with the training data for the fisheye data.

Results

1. **Training time**

Model	Poster	Library	Redw.	BWW	Egypt	Avg.
Nerfacto	89.65	88.53	90.54	95.23	88.28	90.45
Splatfacto	13.71	14.69	16.29	14.38	26.86	17.19

Table 2: Dataset Scene Training Time (minutes)

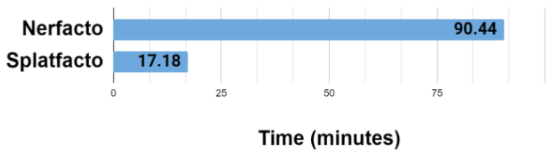

Figure 4: Avg. training time in minutes Nerfacto vs. splatfacto

2. **Rendering rate**

Model	Poster	Library	Redw.	BWW	Egypt	Avg.
Nerfacto	0.39	0.39	0.39	0.39	0.39	0.39
Splatfacto	26.19	24.22	19.36	22.57	19.6	22.38

Table 3: Dataset Scene rendering rate (fps)

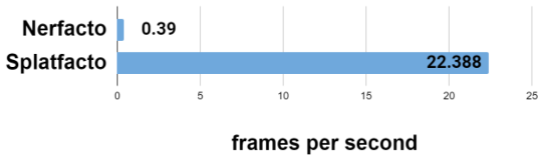

Figure 5: Avg. rendering rate in fps Nerfacto vs. splatfacto

3. **Model memory requirements**

Model	Iter.	Poster	Library	Redw.	BWW	Egypt
Nerfacto	5k	74.0	74.0	74.0	74.0	74.0
	10k	74.0	74.0	74.0	74.0	74.0
	15k	74.0	74.0	74.0	74.0	74.0
	20k	74.0	74.0	74.0	74.0	74.0
	25k	74.0	74.0	74.0	74.0	74.0
	30k	74.0	74.0	74.0	74.0	74.0
Splatfacto	5k	47.7	113.2	138.9	99.1	426.4
	10k	51.3	160.5	204.5	128.2	463.9
	15k	49.8	184.4	213.3	132.3	561.3
	20k	41.2	141.2	167.4	105.3	415.7
	25k	37.8	127.5	154.1	96.5	368.9
	30k	36.1	119.9	148.2	92.1	347.1

Table 4: Dataset Scene Memory Size (Mbytes) by Iterations

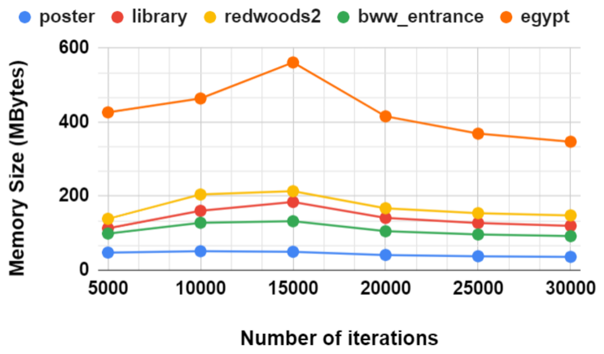

Figure 6: Model memory requirements (Mbytes) for splatfacto for different scenes per training iteration

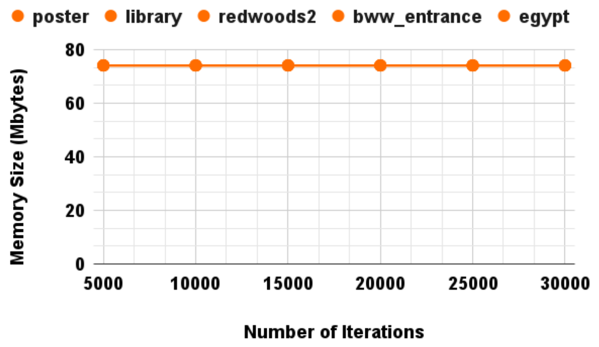

Figure 7: Model memory requirement (Mbytes) for Nerfacto for different scenes per training iteration

4. **Training data memory requirements**
Nerfacto:

$$\text{Training data size} = (5 \times S + 3) \times N_r \times b$$

where 5 represents the input features per spatial point (x, y, z, θ, and ϕ); 3 corresponds to the ground truth color values per pixel (R, G, and B); N_r is the number of rays per batch (with each ray representing one pixel); S is the number of samples taken along each ray; and b is the number of bytes per data type.
Splatfacto:

$$\text{Training data size} = ((H \times W \times 3) + 2) \times b \times B$$

where 3 represents the pixel color values (R, G, and B), 2 represents the network input features (θ and ϕ), H is the image height, W is the image width, b is the number of bytes in the datatype used and B is the batch size.

Model	Poster	Library	Redw.	BWW	Egypt
Nerf. (1 ray)	9.2e-4	9.2e-4	9.2e-4	9.2e-4	9.2e-4
Nerf. (4K rays)	3.79	3.79	3.79	3.79	3.79
Splatfacto	5.93	5.93	5.93	5.93	5.93

Table 5: Training data memory requirements (MBytes). The datatype used for the measurements is float 4 bytes, for Nerfacto the number of samples per ray used is 48 samples, for Splatfacto the undistorted image resolution is 540x960 and the batch size for Splatfacto is 1

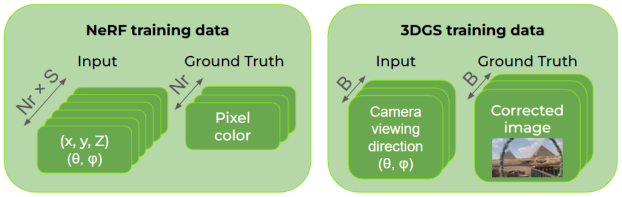

Figure 8: Comparison of the training data content between NeRF and 3DGS models. N_r represents the number of rays per batch, S represent the number of samples per ray and B is the batch size.

5. Validation subset image quality

model	Poster	Library	Redw.	BWW	Egypt	Average
			SSIM			
Nerfacto	0.84	0.9	0.84	0.77	0.77	0.82
Splatfacto	0.97	0.93	0.93	0.955	0.7	0.89
			LPIPS			
Nerfacto	0.11	0.04	0.08	0.08	0.12	0.08
Splatfacto	0.03	0.07	0.05	0.03	0.22	0.08
			PSNR			
Nerfacto	20.46	27.64	24.21	22.62	23.55	23.69
Splatfacto	35.85	24.77	28.59	31.02	19.7	27.98

Table 6: Nerfacto vs spaltfacto quantitative results for validation subset

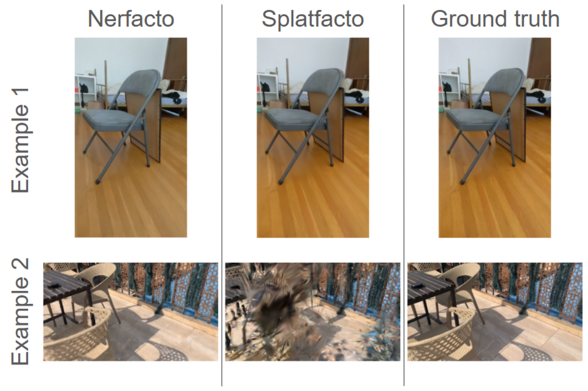

Figure 9: Nerfacto vs splatfacto qualitative results for validation subset

6. Novel view image quality

			FID		
Model	**Poster**	**Library**	**Redw.**	**BWW**	**Egypt**
Nerfacto	195.22	181.84	109	194.55	176.83
Splatfacto	212.06	220.15	117.34	188.81	183.92

Table 7: Nerfacto vs spaltfacto quantitative results (FID) for novel view rendering

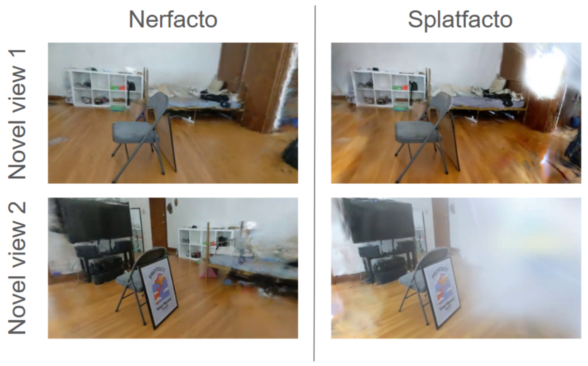

Figure 10: Nerfacto vs spaltfacto qualitative results for novel view rendering

7. Fisheye dataset

	Validation Metrics			NV Metric
Model	**SSIM**	**LPIPS**	**PSNR**	**FID**
Nerfacto	0.66	0.2	18.35	187.05
Splatfacto	0.71	0.24	18.5	377.18

Table 8: Nerfacto vs spaltfacto quantitative result for the fisheye scene for the validation subset and the novel view rendering

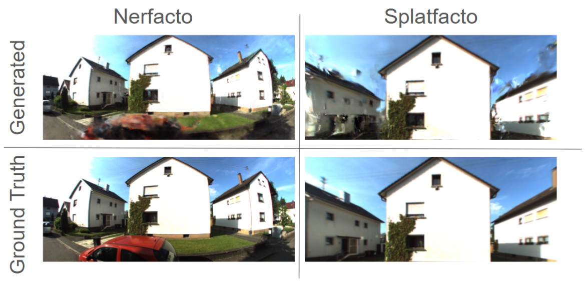

Figure 11: Nerfacto vs spaltfacto qualitative results for fisheye scene validation subset

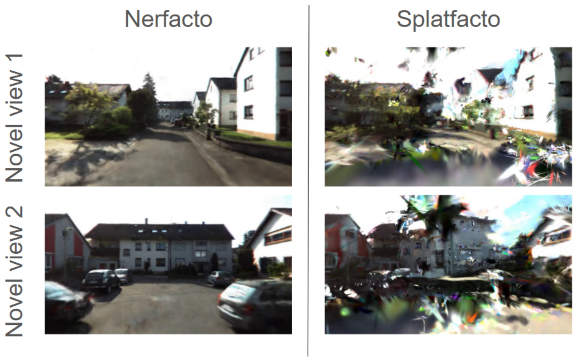

Figure 12: Nerfacto vs spaltfacto qualitative results for fisheye scene novel view rendering

Discussion

1. **Training Time**
   The results indicate that the NeRF (Nerfacto) model requires around five times the training time compared to the Gaussian Splatting (Splatfacto) model. This difference is a critical factor, particularly when on-device training is necessary.
2. **Rendering Speed**
   NeRF: Rendering with NeRF involves sampling the neural network multiple times for each pixel to obtain density and color values for points in space along the camera ray cast through the pixel, which are then integrated to determine the pixel's RGB value. Consequently, the rendering rate for the Nerfacto model is very slow, as shown in the results. **3DGS**: Due to its explicit scene representation, Gaussian Splatting can employ rasterization techniques for rendering. As a result, the 3DGS model achieves near real-time rendering, as demonstrated in the results.
3. **Model Memory Requirements**
   NeRF: NeRF models typically have a fixed memory requirement determined by the architecture of the neural network. This requirement remains consistent regardless of the number of training iterations or scene complexity, which is advantageous for embedded systems with limited memory that need predefined allocation to avoid dynamic adjustments. **3DGS**: Instead of a neural network, 3DGS represents the environment as a set of 3D Gaussian blobs, optimizing parameters such as the mean and covariance of each Gaussian during training. The number of Gaussians used can vary to better represent the scene, leading to dynamic memory requirements that fluctuate with the number of iterations and/or the scene's complexity (as illustrated in fig. 6). A potential solution for fixing a maximum value for the memory requirements is to cap the number of Gaussians, though this would likely impact the quality of scene representation.
4. **Training Data Memory Requirements**
   NeRF: Training in NeRF involves casting rays through each pixel and sampling points along the ray, predicting color and density values for each point. These values are then integrated to produce a pixel prediction, which is compared to the ground truth pixel value. This setup allows some flexibility in adjusting memory requirements by changing the number of rays per batch and the number of samples per ray, though reductions could compromise model quality and training time. **3DGS**: 3DGS uses undistorted images as ground truth, with the camera viewing angle as input. Even with the minimum batch size, memory must accommodate at least one image and its input data, making 3DGS's training data memory requirements somewhat inflexible and more demanding compared to NeRF.
5. **Validation Subset Quality**
   Qualitative Results: In many cases, 3DGS (Splatfacto) produces vibrant colors and sharper details where Gaussians are accurately placed. However, when Gaussian placement is inaccurate, the rendered images can appear distorted or unrecognizable (as shown in fig. 9). **Quantitative Results**: Quantitatively, 3DGS generally outperforms NeRF across most metrics, providing better scores on various image quality measures.
6. **Novel View Rendering Quality**
   Qualitative Results: NeRF (Nerfacto) tends to perform better with unseen viewpoints, while 3DGS (Splatfacto) often displays significant artifacts when viewed from new angles (as shown in fig. 10). **Quantitative Results**: The quantitative data aligns with the qualitative observations, with Nerfacto achieving superior metrics compared to Splatfacto for novel view rendering.
7. **Fisheye Dataset**
   NeRF: NeRF models, which use pixel-level data, can directly utilize fisheye camera models, allowing the entire field of view to be used without modification. This capability enables NeRF to fully leverage the information in fisheye images. **3DGS**: In contrast, 3DGS requires undistorted images as ground truth, meaning fisheye images must be corrected and undistorted prior to training. This process introduces several issues:
   - **Field of View Reduction**: The undistortion process necessitate cropping parts of the image, particularly in the corners where distortion is most severe, leading to data loss.
   - **Geometry Alteration and Resolution Loss**: Edge pixels must be stretched to fit a rectilinear image format, which can distort objects and reduce resolution at the edges.
   - **Artifacts**: Depending on the correction method, artifacts such as aliasing or blurring may occur.

   Validation Subset Image Quality: Quantitative measures indicate that Splatfacto generally achieves higher image quality. However, these results should be interpreted carefully, as 3DGS uses undistorted ground truth images for the validation subset missing substantial areas of information compared to the fisheye images data used by Nerfacto.
   Novel View Rendering Quality: Since undistorted images have a smaller field of view than the original fisheye images, 3DGS struggles with rendering novel views, especially in occluded or unseen areas. This issue highlights the limitations of the discrete nature of 3DGS, with both qualitative and quantitative results clearly reflecting this shortfall.

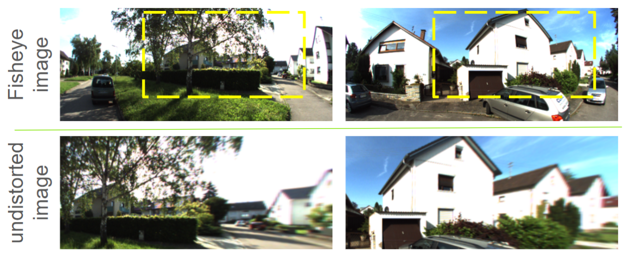

Figure 13: Fisheye images (used in nerfacto) vs undistorted images (used in splatfacto)

Conclusion

Both NeRF-based and 3DGS-based models offer distinct strengths and limitations for automotive and edge applications. **3DGS** excels in rendering speed and training time, making it ideal for scenarios requiring instantaneous feedback, such as parking assistance and autonomous driving visualization. It also produces high-quality images for viewpoints within the training range, making it suitable when virtual camera deviations are minimal.

In contrast, **NeRF** outperforms **3DGS** in novel viewpoint rendering due to its continuous representation, effectively handling occlusions and unseen areas. This makes **NeRF** more suitable for applications requiring significant virtual camera movement, such as complex parking scenarios.

In terms of memory, **3DGS** faces challenges due to its dynamic memory requirements, which fluctuate with scene adaptation, limiting its use in memory-constrained systems. **NeRF**, with its constant and predictable memory usage, is better suited for applications with strict memory constraints. Moreover, the high memory usage per training sample in **3DGS** restricts its effectiveness for on-target training in adapting to changing scenes, while **NeRF** remains flexible in such environments.

A key distinction is their handling of fisheye data. **NeRF** can natively process fisheye images without field-of-view loss or distortion. In contrast, **3DGS** requires pre-processing into rectilinear images, leading to quality degradation and reduced field of view.

Model	**Rendering Rate**	**Training Time**	**Model Mem. Req.**	**Training Mem. Req.**	**N.V. Quality**	**FE data**
NeRF			x	x	x	x
3DGS	x	x				

Table 9: NeRF vs 3DGS: metric-based performance comparison

While neither model fully meets the requirements of automotive applications, their complementary strengths suggest potential for a **hybrid model**. Adapting **3DGS** to handle **fisheye data** directly through a differentiable fisheye renderer, the use of augmented datasets (e.g., bowl-view images), or improved fisheye correction, could enhance image quality by reducing field-of-view loss and artifacts. Alternatively, a pre-trained **NeRF** model could model the continuous **background**, while **3DGS** adapts dynamically to **scene changes**, improving novel view rendering. These strategies highlight the potential of combining these technologies to advance automotive applications and bridge the gap between research and real-world deployment. We hope this work provides valuable insights into the strengths, limitations, and future improvements of NeRF and 3DGS.

Acknowledgments

The authors would like to thank Louis Gallagher (Maynooth University) for his valuable contribution in integrating the KITTI 360 dataset into the Nerfstudio framework.

References

[1] Matthew Tancik, Emily Weber, Eric Ng, Ruicheng Li, Boyuan Yi, Tzu-Mao Wang, Andreas Kristoffersen, Jason Austin, Kiana Salahi, Ananya Ahuja, and David McAllister, "Nerfstudio: A modular framework for neural radiance field development," in ACM SIGGRAPH 2023 Conference Proceedings, 2023, pp. 1–12.

[2] NeRF Studio, "Nerfacto: A Method for Neural Radiance Fields," available at: https://docs.nerf.studio/nerfology/methods/nerfacto.html, 2023. Accessed: 2023-06-19.

[3] NeRF Studio, "Splat: A Method for Neural Radiance Fields," available at: https://docs.nerf.studio/nerfology/methods/splat.html, 2023. Accessed: 2023-06-19.

[4] Yiyi Liao, Jun Xie, and Andreas Geiger, "KITTI-360: A Novel Dataset and Benchmarks for Urban Scene Understanding in 2D and 3D," in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 45, no. 3, pp. 3292-3310, 1 March 2023.

[5] 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, vol. 65, no. 1, pp. 99–106, 2021.

[6] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis, "3D Gaussian Splatting for Real-Time Radiance Field Rendering," ACM Transactions on Graphics, vol. 42, no. 4, pp. 1–14, 2023.

[7] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller, "Instant neural graphics primitives with a multiresolution hash encoding," ACM Transactions on Graphics, vol. 41, no. 4, pp. 1–15, 2022.

[8] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter, "GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium," in Advances in Neural Information Processing Systems, vol. 30, pp. 6626–6637, 2017.

[9] Stephan J. Garbin, Marek Kowalski, Matthew Johnson, Jamie Shotton, and Julien Valentin, "Fastnerf: High-fidelity neural rendering at 200fps," in Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 14346–14355, 2021.

[10] Yiming Zhou, Zixuan Zeng, Andi Chen, Xiaofan Zhou, Haowei Ni, Shiyao Zhang, Panfeng Li, Liangxi Liu, Mengyao Zheng, and Xupeng Chen, "Evaluating modern approaches in 3D scene reconstruction: NeRF vs Gaussian-based methods," arXiv preprint arXiv:2408.04268, 2024.

[11] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image Quality Assessment: From Error Visibility to Structural Similarity," IEEE Transactions on Image Processing, vol. 13, no. 4, pp. 600–612, April 2004.

[12] R. Zhang, P. Isola, A. A. Efros, E. Shechtman, and O. Wang, "The Unreasonable Effectiveness of Deep Features as a Perceptual Metric," in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 586–595, 2018.

[13] Mary Raymond, Ganesh Sistu, and Louis Gallagher, "No-reference Image Quality Metric for NeRF (Neural Radiance Fields) Rendering in Automotive Applications," in Proceedings of the Irish Machine Vision and Image Processing Conference (IMVIP), 2024, forthcoming.

Author Biography

Mary Raymond earned her Master's degree in Artificial Intelligence from the University of Galway in 2022 she has over 12 years of experience in automotive vision applications, with a focus on low-level development and hardware optimization. Currently a Lead Software Engineer in the Valeo Brain division, she specializes in AI research. Since 2023, her work has centered on developing AI techniques for novel view rendering and exploring their applications in the automotive industry.

Ganesh Sistu, Principal AI Architect at Valeo Ireland and Adjunct Assistant Professor at the University of Limerick, Ireland. He is leading multiple global research teams working on automated driving and parking. With over 14 years in computer vision and machine learning, he has contributed to 35+ publications in top-tier conferences like ICCV and ICRA. He also plays a pivotal role in guiding the future of AI education as an industry board member for Science Foundation Ireland's Data Science PhD and the National MSc in AI programs at the University of Limerick, blending academic insight with industry expertise.