Stanford Architecture Benchmark

George Nakayama and Rishi Agarwal
Stanford University

Abstract

Learning 3D scene representations from 2D images has become popular recently, especially since the community has found efficient ways of representing 3D information such as Nerf and computationally quicker ways of rendering the 3D information such as Gaussian Splatting. In this paper, we contribute a novel dataset to the community which was captured using an iPhone under a fairly casual setting with no control over lighting that includes unbounded outdoor scenes. We use this dataset to benchmark existing techniques on the task of novel-view synthesis similar to the approach taken by other authors in this line of work. Our experiments suggest that Gaussian Splatting continues to outperform Nerf for these new scenes. Our data and code can be found at `https://github.com/georgeNakayama/CS231A`.

1. Introduction

It has been a long-standing area of research to learn 3D models from 2D images - some works focus on learning it from a single 2D image such as [20], some focus on multiple 2D images [22], and there are separate lines of work distinguished by the category of data being represented - humans [10], objects [2, 9] and environments [6, 19]. Recent advances in neural radiance fields achieved photorealistic rendering quality of arbitrary novel views given a set of posed images [1, 13, 14]. This is a breakthrough in this field because it provides with an efficient 3D representation that can be learned using 2D images only, without any ground truth information about the actual 3D model.

They still suffer from long training and rendering time because of their implicit volumetric rendering approach, which requires a dense sampling along each camera ray to compute the volumetric integrals. To tackle this issue, Kerbl et. al introduced Gaussian Splatting (GS) [8] that replaces the volumetric representation with rasterization using explicitly learned Gaussians, enabling real-time rendering and training with photorealistic qualities.

While Gaussian Splatting allows for realistic 3D reconstruction using input views. Its generalization ability is only tested on a set of existing datasets [1, 6, 9, 13] that are often limited in the scene scale and is captured in a controlled lighting and environmental settings. In this project, we attempt to further test out the generalization ability of GS by capturing more in the wild dataset with different environmental conditions. Specifically, we capture an in-the-wild dataset under high-lighting, normal-lighting, and low-lighting conditions, as well as datasets with complex geometry and topological structures.

Given a set of 2D input images, our task is to construct a 3D representation of the scene using a recent technique called Gaussian Splatting [8]. Specifically, we will use COLMAP [21] to obtain SfM points from a set of 2D RGB images that we collect in the wild. Next, we will initialize GS using these SfM points and run the optimization to produce the 3D representation.

In Figure 2 we describe the entire pipeline starting from SfM points. The optimization of GS starts from initializing Gaussians from SfM points (which were obtained by running COLMAP on the 2D images). Each Gaussian is optimized independently of its color, position, scale, and opacity so that the point-based rasterization can reconstruct the input images. The size of the Gaussian pool is adaptively controlled and the entire pipeline is trained end-to-end with gradient descent.

Our contributions in this project are primarily as follows:

- A new dataset consisting of 2D RGB images collected using a phone camera under varied lighting conditions, consisting of indoor, outdoor, bounded and unbounded scenes. Some of the scenes are (multi) object focused, and some are environment focused.
- Testing out SOTA approach on this new dataset and comparing it with other approaches such as Nerf to evaluate the quality of results. Our goal is to add a new dataset to the set of benchmarks available in this line of research.

2. Related Work

Novel view synthesis. There have several works which focus on learning 3D models of the scene in the form of signed distance functions or occupancy fields [5, 7, 12, 17],

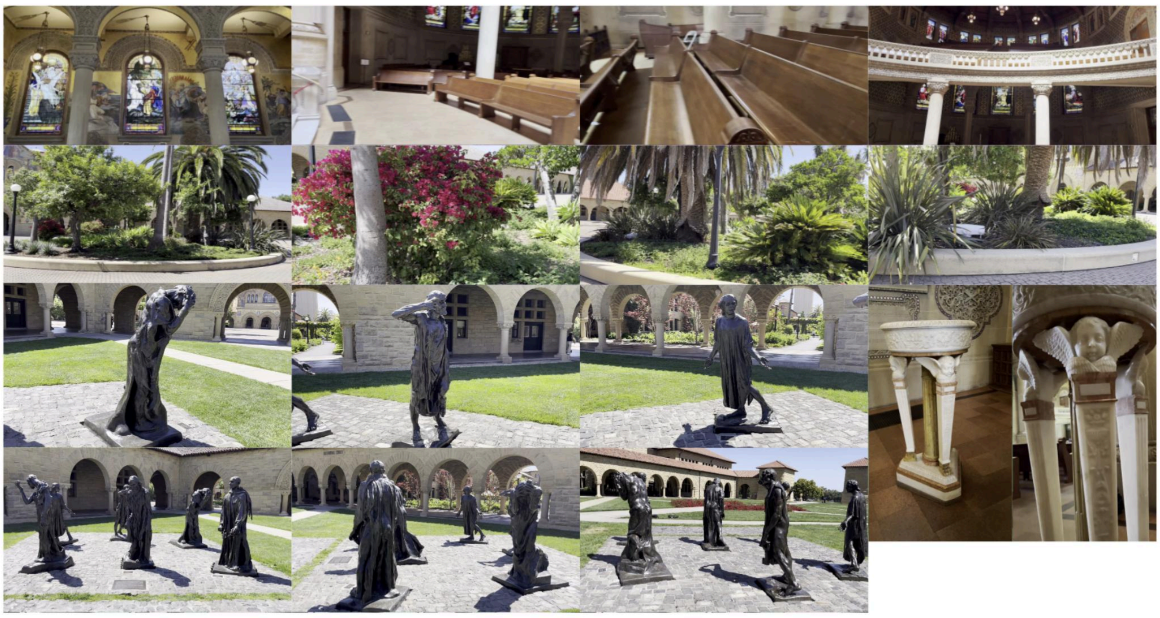

Figure 1. **Stanford Architecture Dataset.** Consists of 6 scenes - four outdoor scenes consisting of hallway, garden, statues and buildings and two indoor scenes consisting of the environment and a marble statue. We capture diversity in terms of lighting conditions, number of objects being captured and the bounds of the scene.

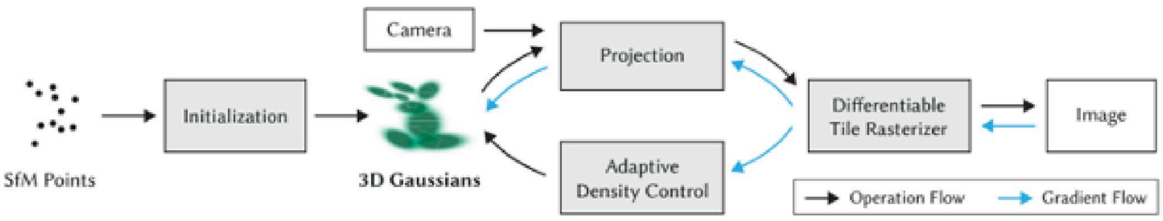

Figure 2. **Gaussian Splatting Pipeline from [8].** The optimization of GS starts from initializing Gaussians from SfM points. Each Gaussians are optimized independently of their color, position, scale, and opacity so that the point-based rasterization can reconstruct the input images. The size of the Gaussian pool is adaptively controlled and the entire pipeline is trained end-to-end with gradient descent.

some others which focus on learning implicit representations using differentiable rendering which can be queried to generate novel views [16, 24] and some other works in CV and graphics communities which focus on generating novel views by learning meshes or voxels [3, 4, 11, 23]. While the first line of work requires supervision in the form of ground truth 3D data, the others can work with only 2D images (multiple views of the scene at hand). In our work, we focus on the task of novel view synthesis to compare different approaches of learning 3D representations from 2D views.

Nerfs. This is one of the breakthroughs in the line of representing 3D information in an efficient manner. Specifically, it uses 5D representation to store the scene information (x,y,z,θ,ϕ) where first three coordinates represent point location and second two denote the viewing angle. With this input, it outputs (R,G,B,O) values where O stands for opacity. To project this 3D information on a 2D surface one needs to integrate across multiple points along multiple rays, making the querying process expensive. There have been works which try to speed up the query time at the cost of rendering quality [1, 14, 25]

Gaussian splatting. This is an alternate solution to Nerfs solving the same problem of learning 3D information [8]. The advantage is that its faster than Nerfs when it comes to fitting the 3D information from 2D views, and also allows for near real time rendering without any drop in quality,

MipNeRF360 [1]	PSNR ↑	SSIM ↑	LPIPS ↓
Plenoxels	23.08	0.626	0.463
Instant NGP	25.30	0.671	0.371
MipNeRF 360	27.69	0.792	0.237
Reported GS-7K	25.60	0.770	0.279
Reported GS-30K	27.21	0.815	0.214
Reproduced GS-7K	25.92	0.766	0.288
Reproduced GS-30K	27.49	0.812	0.221

Table 1. **Reproduced Novel View Synthesis Result on MipNeRF 360 Dataset**. Reported numbers are copied from [8].

MipNeRF360 [1]		bicycle	bonsai	counter	flowers	garden	kitchen	room	stump	treehill
	PSNR ↑	23.604	28.850	26.705	20.515	26.245	28.546	28.139	25.709	22.085
Reported GS-7K	SSIM ↑	0.675	0.910	0.873	0.525	0.836	0.900	0.884	0.728	0.598
	LPIPS ↓	0.318	0.205	0.204	0.336	0.103	0.129	0.220	0.210	0.317
	PSNR ↑	**25.246**	31.980	28.700	**21.520**	**27.410**	30.317	30.632	26.550	22.490
Reported GS-30K	SSIM ↑	**0.771**	0.938	0.905	**0.605**	**0.868**	0.922	0.914	**0.775**	**0.638**
	LPIPS ↓	**0.205**	0.205	0.204	**0.336**	**0.103**	0.129	0.220	**0.210**	**0.317**
	PSNR ↑	23.373	29.815	27.238	20.252	26.069	29.110	29.580	25.703	22.096
Reproduced GS-7K	SSIM ↑	0.639	0.926	0.885	0.510	0.814	0.909	0.903	0.720	0.588
	LPIPS ↓	0.374	0.213	0.229	0.441	0.186	0.150	0.239	0.325	0.435
	PSNR ↑	25.088	**32.294**	**29.080**	21.361	27.309	**31.423**	**31.630**	**26.614**	**22.595**
Reproduced GS-30K	SSIM ↑	0.747	**0.946**	**0.914**	0.588	0.856	**0.931**	**0.926**	0.769	0.636
	LPIPS ↓	0.243	**0.181**	**0.184**	0.359	0.122	**0.117**	**0.198**	0.242	0.346

Table 2. **Per-Scene Breakdown of Novel View Synthesis Result on MipNeRF 360 Dataset**. Reported numbers are copied from [8].

in fact the quality only improves when compared to Nerfs. The idea of this approach is to fit Gaussians at the points outputted from SfM and fit them using the multiple views. We include this approach in the various approaches that we compare on our benchmark.

Existing Datasets. There have been multiple datasets released for this line of work [2, 6, 9]. However, there are several limitations of existing datasets. For example, [6, 9] focuses on object centric 3D scenes primarily. The scenes which are unbounded and more environment focused have been captured in a professional way which is expensive to be done at a larger scale. [2] is a unique dataset whihc contains the 3D ground truth information of the models, however, this information is even more expensive to collect and in fact practically impossible for unbounded scenes. In contrast, we focus on collecting a dataset in a casual setting using not-so-expensive iPhone cameras focusing explicitly on diversity in the scenes for a fairer evaluation.

3. Technical Approach

3.1. Gaussian Splatting

Given a set of images from a static scene, Gaussian Splatting aims to render novel views from unseen camera poses after optimization. To achieve this, it optimizes the parameters of a set of 3D Gaussians to represent the scene. Specifically, a set of 3D Gaussians are initialized either from the structure from motion (SfM) points or randomly within the scene bound, each with a full covariance matrix Σ and a mean μ. Furthermore, each Gaussian also carries two additional parameters of opacity α and color c that is used for rasterization. In 3D, the color contribution from Gaussian G at point $x \in \mathbb{R}^3$ is defined as the density weight color:

$$c_G(x) = c_G \exp\left[(x-\mu_G)^T \Sigma_G^{-1}(x-\mu_G)\right]. \qquad (1)$$

Gaussian splitting trains on the set of 2D posed images by projecting all the 3D Gaussians to 2D. Because the projection operation is nonlinear, making the projected 3D Gaussian complex, it uses a local affine approximation so that the projected 2D Gaussian G^{2D} of the 3D Gaussian G using the view matrix W will be centered at $W\mu_G$ with a 2D Covariance matrix being the first two rows and columns of

$$\Sigma'_G = JW\Sigma_G W^T J^T. \qquad (2)$$

Here, J is the Jacobian of the view projection, which acts as the locally affine approximation. After doing so, the rendered image is computed as an α-blending of all the projected Gaussian in a tile-based rasterization pipeline. In particular, images are first divided into 16×16 tiles, and Gaussians are first culled and sorted based on their depth.

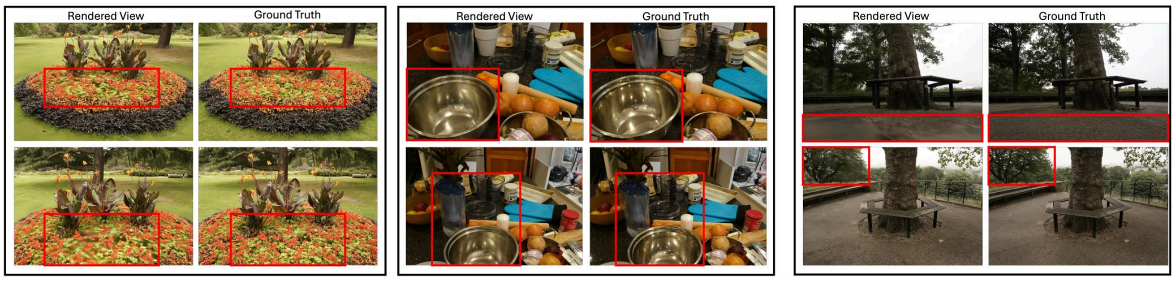

Figure 3. **Qualitative Comparisons on MipNeRF 360.** We visualize test views from our reconstructed Gaussian Splatting on MipNeRF 360. We notice that despite GS's photorealistic NVS result in most of the regions, they still have many artifacts as highlighted in the encircled regions. Notice that they suffer from fine geometry details in under-represented regions as shown in the left comparison. In the middle image, we see that metallic materials cannot be well represented because Gaussians lack ability to represent more complex view-dependent details. Lastly, in the rightmost comparison, the ground plane is blurry and not reconstructed well. In our project, we seek to obtain datasets that further test GS's generalizability to challenging settings.

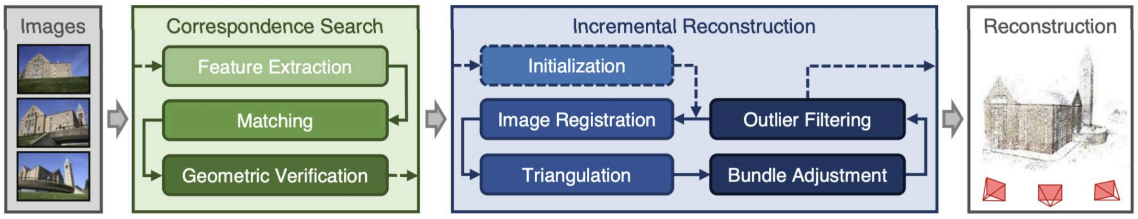

Figure 4. **COLMAP pipeline from [21].**

Finally, each pixel's color is computed via

$$C(x) = \sum_{i \in N} c_i \sigma_i \prod_{j=1}^{i-1} (1 - \sigma_j), \sigma_i = \alpha_i G_i'(x). \tag{3}$$

Here, N is the set of 2D Gaussians sorted based on increasing depth. c_i, α_i are the color and opacity associated with the i-th Gaussian and $G_i'(x)$ is the evaluation of the projected 2D Gaussian kernel at pixel x. To capture view-dependent effects, the colors of each Gaussian is represented using spherical harmonics. Finally, the rendered images are compared against the training images using a combination of photometric L^2 loss and a structural similarity loss, resulting in the total loss of

$$\mathcal{L}_{render} = \mathcal{L}_{L^2} + \lambda \mathcal{L}_{SSIM}. \tag{4}$$

We use $\lambda = 0.2$ for all our experiments.

The above operation defines a way to optimize the parameters of the 3D Gaussian with only 2D supervision. To make the optimization more stable, [8] also parameterizes the covariance matrices each as a scaling matrix S and a rotation matrix R, so that

$$\Sigma = RSS^T R^T. \tag{5}$$

In doing so, GS only optimizes a scaling vector $s \in \mathbb{R}^3$ and a quaternion q that use them to recover the full covariance matrices.

Although Gaussians are initialized using the SfM points, GS also enables adaptive control over the number of optimized Gaussians using a pre-defined scheme. This is done by splitting large Gaussians and cloning small Gaussians to enable more expressivity. Moreover, Gaussians that do not contribute to rasterization are removed at regular intervals to save computation. The entire Gaussian Splatting pipeline is illustrated in Figure 2.

3.2. COLMAP

COLMAP [21] is a general-purpose, end-to-end image-based 3D reconstruction pipeline (i.e., Structure-from-Motion (SfM) and Multi-View Stereo (MVS)) that is popularly used for 3D reconstruction using unstructured 2D images. The software contains a graphical and command-line interface so it is easy to use, and it offers a wide range of features for 3D reconstruction. The pipeline of COLMAP is illustrated in Figure 4, which can be divided into two main parts: correspondence search and 3D reconstruction. Given the input of a set of unstructured images $\{I_1, \ldots, I_n\}$, COLMAP first extracts appearance features

Scene	Enviroment	Setting	Camera Arrangement	# of Images Registred
Hallway	Outdoor	Scene	Front-facing	867
Garden	Outdoor	Scene	Inward-facing	926
Statue 1	Outdoor	Object	Outward-facing	1992
Statue 2	Outdoor	Object	Inward-facing	1676
Church 1	Indoor	Scene	Outward-facing	2205
Church 2	Indoor	Object	Inward-facing	2654

Table 3. **Scene Statistics of Our Collected Stanford Dataset**.

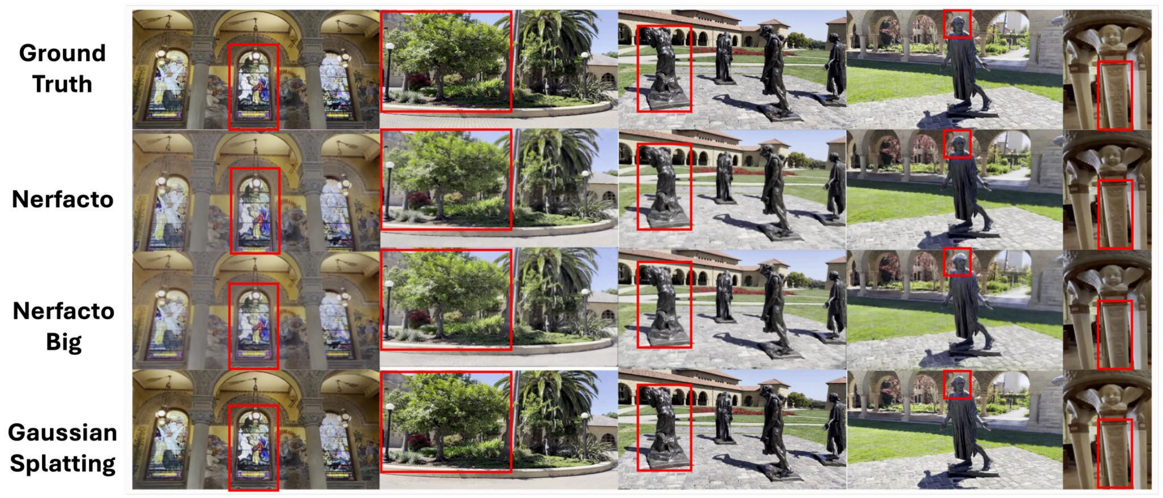

Figure 5. Qualitative Comparison Of NeRF baselines against Gaussian Splatting on Our Collected Scenes.

$\{F_1, \dots, F_n\}$ from each of the images that are used for correspondence matching.

In the next step, the extracted feature maps are matched with each other to obtain correspondences across input views. A naive matching method will matching all pairs of features, resulting in prohibitive computational cost if the number of training images is large. Other matching methods can also be used such as sequential matching when the input images are taken from serial sources such as video frames. This only requires a linear number of matching with respect to the number of input images.

Because the above feature matching only takes account of the images' appearance, the resulting correspondences can be outlier-contaminated and error-prone. To alleviate this issue COLMAP further incorporates geometric verification of the found correspondences, so that a valid transformation (either epipolar or holomorphic transformation) between the matched images.

After correspondence points are found between input views and are verified, COLMAP proceeds to register the input images into a common 3D space. The 3D reconstruction is done incrementally starting with a carefully selected 2-view reconstruction. At each iteration, one new input image is registered into the 3D scene by solving the perspective-n-point problem, with resulting in the registered camera parameters including the extrinsic transformations and intrinsic parameters. After the image is registered, triangulation of the new correspondences is performed to obtain a new set of 3D scene points. Finally, bundle adjustment is performed to further refine the camera parameters and 3D points.

At the end of the pipeline, we end up with a set of posed images and a set of structure-from-motion points in 3D that represent the 3D reconstructed scene. However, the output SfM points are usually sparse and is noisy due to the outlier-contaminated correspondences. Because of this, we only use COLMAP to obtain the camera parameters and use them and the images to train Gaussian Splatting to obtain photorealistic reconstruction of the scene.

4. Experiments and Evaluation

4.1. Data Collection

We describe how to collect data on Stanford Campus and process it so that it can be used for GS training. Existing datasets such as Blender Synthetic [13] or MipNeRF360 [1] are either synthetically created or focusing on objects or

Stanford Collection		Average	Church I	Church II	Statues I	Statues II	Garden	Hallway
Nerfacto	PSNR ↑	21.34	19.46	24.41	22.72	22.8	17.55	21.11
	SSIM ↑	0.66	0.66	0.81	0.7	0.68	0.4	0.73
	LPIPS ↓	0.32	0.42	0.31	0.23	0.2	0.44	0.325
Nerfacto-Big	PSNR ↑	17.36	17.95	20.63	15.78	17.06	15.82	16.94
	SSIM ↑	0.49	0.58	0.68	0.37	0.43	0.28	0.61
	LPIPS ↓	0.35	0.43	0.35	0.25	0.33	0.39	0.36
Gaussian Splatting	PSNR ↑	**23.49**	**20.43**	**29.71**	**25.25**	**26.17**	**19.44**	**26.10**
	SSIM ↑	**0.78**	**0.76**	**0.93**	**0.81**	**0.87**	**0.58**	**0.85**
	LPIPS ↓	**0.25**	**0.37**	**0.26**	**0.19**	**0.16**	**0.36**	**0.24**

Table 4. **Per-Scene Breakdown of Novel View Synthesis Result on Our Stanford Dataset**.

small-scale scenes. While they poses challenges in the level of details of the objects in view, the limited scales and controlled environments of the captured scenes cannot fully test Gaussian Splatting's robustness to different environmental settings.

To this end, we captured 6 scenes around Stanford campus that is *large-scale*, *in-the-wild*, and with *unconstrained* camera poses. Furthermore, to mimic the real life scenario of the data capture process from consumers, we simply use the cameras from and IPhone without using its camera intrinsic information. These constraints make both the image registration and 3D reconstruction process noise-prone and thereby fulfill our goal of testing the robustness of Gaussian Splatting.

Specifically, we captured both indoor and outdoor scenes in various settings and camera arrangements as described in Tab. 3. Visuals are shown in Figure 1.

4.2. Gaussian Splatting Validation

We validated Gaussian Splatting's implementation on MipNerf 360 dataset. Table 1 shows the overall quantitative comparison of the reproduced results against the reported scores. Tab. 2 shows the per-scene breakdown, suggesting (results in Table 1, Table 2, Figure 3). We observe that running the optimization for longer (7K v.s. 30k) always results in better NVS quality, and our reproduction of the scenes reported in the Gaussian Splatting paper shows on-par results against the reported metrics.

4.3. Novel View Synthesis on Stanford Dataset using Gaussian Splatting

We run Gaussian Splatting on our collected data described in Sec. 4.1. We use the default hyperparameters provided by the Gaussian Splatting repository and run each scene for a total of 30000 iterations. Due to memory constraints, we downsample the input views uniformly to within 1000 images. We follow the train-test split given by [1] that holds out every eighth of the input view as the test image. The training takes roughly 40 minutes on a single Nvidia A6000 graphics card.

For baseline comparison, we run two NeRF baselines: Nerfacto and Nerfacto-big using NeRFstudio [25]. Both Nerfacto and Nerfacto-big are hash-grid-based NeRF variants that achieve both efficiency and photorealism. We also use their default parameters as provided by NeRFstudio and run on the same splits as Gaussian Splatting. Nerfacto is run for 30000 iterations while its larger variant Nerfacto-big is run for 100000 iterations. The training takes around 40 minutes and 3 hours respectively for Nerfacto and Nerfacto-big on a single NVidia A6000 graphics card.

Table 4 shows the quantitative comparison of the NVS metrics averaged over all scenes as well as the per-scene breakdown comparison. Fig. 5 shows the qualitative comparison on test views. Notice that Gaussian splitting achieves the best reconstruction quality compared to the NeRF baselines, shown by its better ability to model reflective or translucent surfaces. Quantitatively we see that Gaussian splitting out-performs both NeRF baselines over all metrics, further suggesting its superiority in rendering quality.

5. Conclusion & Future Work

In our project, we collected in-the-wild, large-scale, scenes with unconstrained camera poses to test the generalizability of Gaussian Splatting, when compared with other NeRF-based methods. Based on our comprehensive experiments, Gaussian Spaltting outperforms NeRF baselines in all of the scenes we collected, suggesting its better generalizability under challenge settings.

However, while GS outperforms NeRF in large-scale scenes, they are very memory-intensive because of their explicit nature of 3D representation. While recent works [15, 26] tackles this issue via compressors or structuralization, there is still not a canonical way to efficiently store and optimize the Gaussians on a large-scale scene.

Gaussian Splatting's robustness in other directions still needs to be tested. For example, many works [18, 27] using NeRF-based representations tackle 3D reconstruction with only sparse input views. While Sparse GS [28] tackles this setting by incorporating depth and distillation supervision, it only focuses on object-level datasets and leaving scene-level sparse input an open and challenging problem.

References

[1] Jonathan T. Barron, Ben Mildenhall, Dor Verbin, Pratul P. Srinivasan, and Peter Hedman. Mip-nerf 360: Unbounded anti-aliased neural radiance fields. *CVPR*, 2022. 1, 2, 3, 5, 6

[2] Angel X Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, et al. Shapenet: An information-rich 3d model repository. *arXiv preprint arXiv:1512.03012*, 2015. 1, 3

[3] Wenzheng Chen, Huan Ling, Jun Gao, Edward Smith, Jaakko Lehtinen, Alec Jacobson, and Sanja Fidler. Learning to predict 3d objects with an interpolation-based differentiable renderer. *Advances in neural information processing systems*, 32, 2019. 2

[4] Kyle Genova, Forrester Cole, Aaron Maschinot, Aaron Sarna, Daniel Vlasic, and William T Freeman. Unsupervised training for 3d morphable model regression. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8377–8386, 2018. 2

[5] Kyle Genova, Forrester Cole, Avneesh Sud, Aaron Sarna, and Thomas Funkhouser. Local deep implicit functions for 3d shape. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4857–4866, 2020. 1

[6] Peter Hedman, Julien Philip, True Price, Jan-Michael Frahm, George Drettakis, and Gabriel Brostow. Deep blending for free-viewpoint image-based rendering. *ACM Trans. Graph.*, 37(6), dec 2018. 1, 3

[7] Chiyu Jiang, Avneesh Sud, Ameesh Makadia, Jingwei Huang, Matthias Nießner, Thomas Funkhouser, et al. Local implicit grid representations for 3d scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6001–6010, 2020. 1

[8] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics*, 42(4), July 2023. 1, 2, 3, 4

[9] Arno Knapitsch, Jaesik Park, Qian-Yi Zhou, and Vladlen Koltun. Tanks and temples: benchmarking large-scale scene reconstruction. *ACM Trans. Graph.*, 36(4), jul 2017. 1, 3

[10] Nikos Kolotouros, Georgios Pavlakos, Dinesh Jayaraman, and Kostas Daniilidis. Probabilistic modeling for human mesh recovery. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 11605–11614, 2021. 1

[11] Kiriakos N Kutulakos and Steven M Seitz. A theory of shape by space carving. *International journal of computer vision*, 38:199–218, 2000. 2

[12] Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy networks: Learning 3d reconstruction in function space. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4460–4470, 2019. 1

[13] 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. *Commun. ACM*, 65(1):99–106, dec 2021. 1, 5

[14] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM Trans. Graph.*, 41(4):102:1–102:15, July 2022. 1, 2

[15] Simon Niedermayr, Josef Stumpfegger, and Rüdiger Westermann. Compressed 3d gaussian splatting for accelerated novel view synthesis, 2023. 6

[16] Michael Niemeyer, Lars Mescheder, Michael Oechsle, and Andreas Geiger. Differentiable volumetric rendering: Learning implicit 3d representations without 3d supervision. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 3504–3515, 2020. 2

[17] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 165–174, 2019. 1

[18] Barbara Roessle, Jonathan T. Barron, Ben Mildenhall, Pratul P. Srinivasan, and Matthias Nießner. Dense depth priors for neural radiance fields from sparse input views. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2022. 6

[19] Mehdi SM Sajjadi, Henning Meyer, Etienne Pot, Urs Bergmann, Klaus Greff, Noha Radwan, Suhani Vora, Mario Lučić, Daniel Duckworth, Alexey Dosovitskiy, et al. Scene representation transformer: Geometry-free novel view synthesis through set-latent scene representations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6229–6238, 2022. 1

[20] Ashutosh Saxena, Min Sun, and Andrew Y Ng. Make3d: Learning 3d scene structure from a single still image. *IEEE transactions on pattern analysis and machine intelligence*, 31(5):824–840, 2008. 1

[21] Johannes Lutz Schönberger and Jan-Michael Frahm. Structure-from-motion revisited. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 1, 4

[22] Steven M Seitz, Brian Curless, James Diebel, Daniel Scharstein, and Richard Szeliski. A comparison and evaluation of multi-view stereo reconstruction algorithms. In *2006 IEEE computer society conference on computer vision and pattern recognition (CVPR'06)*, volume 1, pages 519–528. IEEE, 2006. 1

[23] Steven M Seitz and Charles R Dyer. Photorealistic scene reconstruction by voxel coloring. *International journal of computer vision*, 35:151–173, 1999. 2

[24] Vincent Sitzmann, Michael Zollhöfer, and Gordon Wetzstein. Scene representation networks: Continuous 3d-structure-aware neural scene representations. *Advances in Neural Information Processing Systems*, 32, 2019. 2

[25] Matthew Tancik, Ethan Weber, Evonne Ng, Ruilong Li, Brent Yi, Justin Kerr, Terrance Wang, Alexander Kristoffersen, Jake Austin, Kamyar Salahi, Abhik Ahuja, David

McAllister, and Angjoo Kanazawa. Nerfstudio: A modular framework for neural radiance field development. In *ACM SIGGRAPH 2023 Conference Proceedings*, SIGGRAPH '23, 2023. 2, 6

[26] Lu Tao, Mulin Yu, Linning Xu, Yuanbo Xiangli, Limin Wang, Dahua Lin, and Bo Dai. Scaffold-gs: Structured 3d gaussians for view-adaptive rendering. *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024. 6

[27] Mikaela Angelina Uy, Ricardo Martin-Brualla, Leonidas Guibas, and Ke Li. Scade: Nerfs from space carving with ambiguity-aware depth estimates. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023. 6

[28] Haolin Xiong, Sairisheek Muttukuru, Rishi Upadhyay, Pradyumna Chari, and Achuta Kadambi. Sparsegs: Real-time 360 sparse view synthesis using gaussian splatting, 2024. 7