EDGS: Eliminating Densification for Efficient Convergence of 3DGS

Dmytro Kotovenko*   Olga Grebenkova*   Björn Ommer

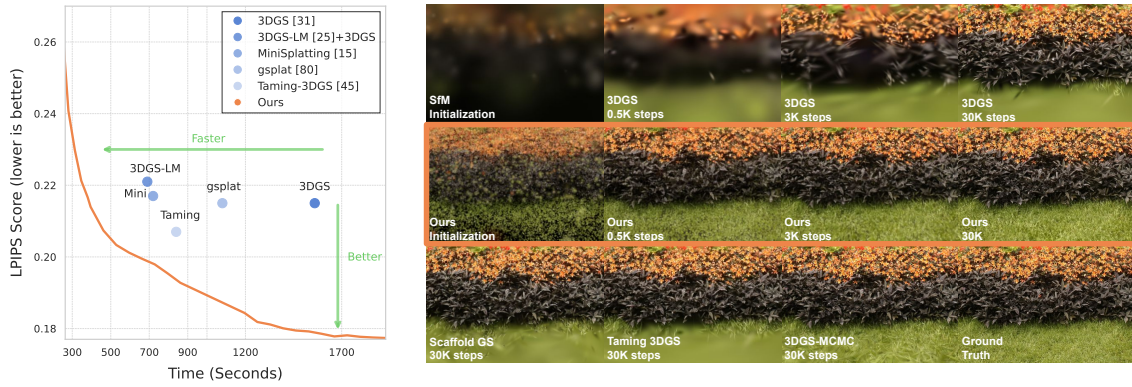CompVis @ LMU Munich,   Munich Center for Machine Learning (MCML)

compvis.github.io/EDGS

Figure 1. 3DGS initializes with a sparse set of Gaussians and progressively adds more in under-reconstructed regions. In contrast, our method begins with a dense initialization derived from triangulated dense 2D correspondences across training image pairs, requiring only minimal refinement. This leads to faster convergence and higher rendering quality. On the left, we compare our approach with state-of-the-art 3DGS acceleration methods on the MipNeRF360 [2] dataset. Our method reaches the original 3DGS [31] LPIPS score in just 25% of the training time and uses only 60% of the final number of splats, outperforming models like 3DGS-LM [25], Taming-3DGS [45], and MiniSplatting [15] trained for the same duration. Note that all reported times include the one-time cost of dense correspondence computation. On the right, we show that our method produces renderings nearly indistinguishable from the ground truth after only 3,000 steps—without any densification. Best viewed zoomed in.

Abstract

3D Gaussian Splatting reconstructs scenes by starting from a sparse Structure-from-Motion initialization and iteratively refining under-reconstructed regions. This process is inherently slow, as it requires multiple densification steps where Gaussians are repeatedly split and adjusted, following a lengthy optimization path. Moreover, this incremental approach often leads to suboptimal renderings, particularly in high-frequency regions where detail is critical.

We propose a fundamentally different approach: we eliminate densification process with a one-step approximation of scene geometry using triangulated pixels from dense image correspondences. This dense initialization allows us to estimate rough geometry of the scene while preserving rich details from input RGB images, providing each Gaussian with well-informed colors, scales, and positions. As

*Equal contribution

1

a result, we dramatically shorten the optimization path and remove the need for densification. Unlike traditional methods that rely on sparse keypoints, our dense initialization ensures uniform detail across the scene, even in high-frequency regions where 3DGS and other methods struggle. Moreover, since all splats are initialized in parallel at the start of optimization, we eliminate the need to wait for densification to adjust new Gaussians.

Our method not only outperforms speed-optimized models in training efficiency but also achieves higher rendering quality than state-of-the-art approaches, all while using only half the splats of standard 3DGS. It is fully compatible with other 3DGS acceleration techniques, making it a versatile and efficient solution that can be integrated with existing approaches.

1. Introduction

Reconstructing accurate 3D scenes from dense collections of 2D images is a fundamental challenge in computer vision [8, 23, 48], with applications in virtual and augmented reality [21, 28, 40, 55, 57, 88], robotics [42, 54, 72], and immersive content creation [1, 4, 20, 36]. The goal is to obtain high-quality 3D representations efficiently, enabling real-time rendering while maintaining reconstruction fidelity. However, achieving this balance between efficiency, speed, and quality requires a representation that is both expressive and computationally efficient. NeRF-based models [2, 18, 48, 49, 83, 84] control the trade-off between quality, computational cost, and representation capacity by designing network architectures and increasing the number of parameters. In contrast, point-based graphics [24, 55, 78] explicitly represent surfaces using discrete primitives, such as meshes or point clouds, offering more direct control over complexity but often struggling with quality and scalability.

Recently, 3D Gaussian Splatting (3DGS) [31] has emerged as a powerful and efficient alternative for 3D scene representation. It represents scenes as a set of optimized 3D Gaussians, mathematical primitives defined by their position, color, and spread. The method starts with a sparse ini-

tialization, typically derived from Structure-from-Motion (SfM) [58], and progressively refines the scene by adding splats to under-reconstructed regions. Through this densification process, 3DGS can reach high rendering quality while efficiently allocating computational resources.

However, this process is suboptimal. The original 3DGS detects under-reconstructed regions using the gradient norm of the photometric loss. But this metric often fails in high-frequency regions and does not align well with human perception. A separate branch of papers has proposed pixel-error-driven formulations [3, 7, 45, 87], gradient calculation improvements [81], and even treating 3DGS as Markov Chain Monte Carlo samples [32]. Despite these efforts, accurately capturing fine details, particularly in high-frequency regions, remains a challenge, as illustrated in Fig. 1. Furthermore, while each densification step is computationally efficient, the overall process is slow. It requires many update steps, as Gaussians must iteratively adjust their parameters before the model determines that additional splats are necessary. This results in a long optimization path, where individual Gaussians undergo multiple refinements before reaching their final states(see Sec. 4.5 and Fig. 5). Densification delays convergence, as it takes many iterations before the model identifies areas requiring higher reconstruction fidelity. These challenges raise an important question: *can we bypass densification entirely?*

The idea of iterative densification of a scene is natural — it mirrors how humans create art. A sculptor starts with rough shapes and progressively refines details; an artist begins with broad strokes before adding finer ones. However, cameras do not operate this way. Instead of refining information over time, a camera captures all available light at once, recording all details simultaneously. It brings us to the idea that waiting for the model to discover where to add details is inefficient. Instead, it is better to allocate resources from the very start and adjust them through all optimization process.

In this paper, we propose a direct initialization strategy that eliminates the need for incremental densification used in the original 3DGS. Rather than waiting for the model to gradually fill in missing

2

details, we precompute a dense set of 3D Gaussians by triangulating dense 2D correspondences across multiple input views. Knowing the viewing rays for each correspondence pixel and the camera poses—but not the depth along those rays—we recover 3D positions by triangulating matched pixels between image pairs. This allows us to assign each Gaussian well-informed initial properties like position, color, and scale from the start. To summarize, we replace the slow, iterative densification of the scene with a densely scattered collection of Gaussians. As a result, each Gaussian is immediately supervised by rich per-pixel photometric signals, allowing for efficient optimization of the entire collection and significantly accelerating convergence.

Although this initialization is noisy(see Fig. 2), we show that it remains robust and leads to faster convergence. Our experiments quantitatively and qualitatively confirm that this approach results in higher reconstruction quality, lower training time, fewer Gaussians, and no need for densification. Our contributions can be summarized as follows:

- We show that initial triangulation based on 2D correspondences can replace the incremental refinement process, fundamentally changing how 3DGS models allocate resources.
- Our method reduces the path each Gaussian must travel in parameter space, demonstrating that careful initialization not only accelerates convergence but also guides optimization toward a convergence point corresponding to lower reconstruction error and thus higher reconstruction quality.
- Our approach outperforms both speed-optimized and quality-focused state-of-the-art models while using only half the splats of standard 3DGS. By improving initialization rather than altering the optimization process, this method is compatible with other 3DGS acceleration techniques, making it a flexible enhancement to existing models.

2. Related Work

Novel View Synthesis It involves generating images from perspectives different from the original input viewpoints. A breakthrough in this area was Neural Radiance Fields (NeRF)[48], which reconstructs complex 3D scenes from 2D images using volumet-

ric rendering techniques[9, 39, 46, 47]. Since then, many follow-up studies have focused on adapting NeRF to sparse input views [27, 33, 52, 63, 84], improving rendering speed [18, 41, 59, 83], and reducing training times [49, 52, 56, 70]. However, sampling points along a ray and passing them through an MLP to obtain density and color introduces significant slowdowns during volume rendering. In contrast, 3D Gaussian Splatting (3DGS) [31] has gained attention due to its explicit representation, high-fidelity results, and real-time rendering speed.

Challenges of 3DGS 3D Gaussian Splatting has shown significant promise in a range of applications, including human avatars [35, 38, 57, 88], text-to-3D generation [6, 61, 82], dynamic scene modeling [10, 30, 44, 67, 74, 76], and more [21, 64, 65, 68, 69, 77, 79]. However, like all methods, 3DGS is not without its limitations. Further advancements have tackled key issues such as anti-aliasing [73, 85], memory usage reduction [19, 37, 43, 50, 51], improving surface reconstruction quality [21, 26], and modeling high-frequency signals by replacing spherical harmonics [75]. Several studies suggest that using an effective strategy for splat densification can significantly enhance performance. RevDev [3] introduced a per-pixel error function as a criterion for densification. AbsGS [81] addressed the issue of gradient collision during the detection of under-reconstructed regions. MiniSplatting [15] proposed a novel densification approach that incorporates both screen-space and world-space information. ScaffoldGS [43] introduced anchor points and implemented a growth algorithm to optimize their distribution. Meanwhile, 3DGS-MCMC [32] reformulated 3DGS densification as a Markov Chain Monte Carlo sampling process, enabling a more efficient Gaussian distribution across the scene. In contrast, we propose an improved initialization method that avoids densification altogether, eliminating the need to detect under-reconstructed regions.

Accelerating 3DGS Several strategies have been developed to improve the speed of 3DGS. One approach leverages pre-trained neural networks as priors to guide reconstruction [5, 14, 71, 89]. For example, MVSplat [5] integrates a multi-view transformer, DepthSplat [71] incorporates depth infor-

Figure 2. Visual comparison of initialization methods on the *stump* scene from the Mip-NeRF360 dataset [2]. The left image represents ground truth. The middle image shows the traditional 3DGS approach initialization with Structure-from-Motion (SfM) [58]. The right image illustrates initialization with our method using matchings. Despite noisy appearance at the initialization, our model can jointly optimize all the gaussians and achieve better reconstruction quality.

mation with the transformer to improve accuracy, and [89] employs a triplane representation. This data-driven strategy enables quick reconstruction with good quality, particularly effective in sparse-view scenarios. In this paper, we focus on dense-view reconstruction. Another area of research targets the optimization of 3DGS efficiency by refining the differentiable rasterizer [11, 16, 45] or improving the framework itself [80]. Separately, 3DGS-LM [25] proposes a Levenberg-Marquardt optimizer that integrates with the 3DGS rasterizer and can be adapted to other rasterization methods. Our approach centers on improving the initialization process, which is compatible with these optimizations and can further increase optimization speed.

Initialization of 3DGS Recent works, such as RAIN-GS [29] and 3DGS-MCMC [32], have shown that random initialization can match the performance of the original 3DGS. In contrast, Rad-Splat [53] initializes from points extracted using pretrained NeRFs to improve quality, though it requires 9 hours of training. Our method departs from both approaches by emphasizing efficiency while outperforming quality-focused methods.

3. Approach

Our key goal is to enhance the initial set of Gaussians (Sec. 3.1) by directly placing them at plausible locations in 3D space, so we can omit the densification process. First, we leverage the availability of multiple images covering the scene and employ a pretrained dense matching network to establish

pixel correspondences across views (Sec. 3.2). To accurately initialize Gaussian positions we solve subsequent 3D triangulation problem (Sec. 3.3).

3.1. Preliminaries

3DGS [31] represents scenes as collections of Gaussians $\mathbb{G} = \bigcup_{i=1}^{N} \boldsymbol{g}_i$, rendered into images using a splatting-based rasterization technique [90]. Each Gaussian component $\boldsymbol{g}_i$ is described by parameters $\{\boldsymbol{g}_i^x, \boldsymbol{\Sigma}_i, \boldsymbol{g}_i^c, \boldsymbol{g}_i^\alpha\}$ for $i \in \{1, \ldots, N\}$. Specifically, $\boldsymbol{g}_i^x \in \mathbb{R}^3$ is the center of the Gaussian $\boldsymbol{g}_i$ in 3D space, $\boldsymbol{\Sigma}_i \in \mathbb{R}^7$ encodes its shape, $\boldsymbol{g}_i^c \in \mathbb{R}^3$ defines its RGB color, and $\boldsymbol{g}_i^\alpha \in \mathbb{R}^1$ indicates its opacity. The color C of a given pixel p is rendered as:

$$C(p) = \sum_{i=1}^{N} \boldsymbol{g}_i^c \boldsymbol{\sigma}_i(p) \prod_{j=1}^{i-1} (1 - \boldsymbol{g}_j^\alpha);$$

$$\boldsymbol{\sigma}_i(p) = \boldsymbol{g}_i^\alpha e^{-\frac{1}{2}(\boldsymbol{p}' - \boldsymbol{g}_i^x)^T \boldsymbol{\Sigma}_i^{-1}(\boldsymbol{p}' - \boldsymbol{g}_i^x)}, \quad (1)$$

where $\boldsymbol{\sigma}_i$ measures the influence of the i-th Gaussian on pixel p, with $(\boldsymbol{p}' - \boldsymbol{g}_i^x)$ representing the shortest distance between the pixel projection line and the Gaussian center $\boldsymbol{g}_i^x$. To project 3D Gaussians to 2D for rendering, following [31], we reparameterize the covariance matrix $\boldsymbol{\Sigma}_i$ as a function of scaling $\boldsymbol{S}_i$ and rotation $\boldsymbol{R}_i$ matrices ensuring the positive semi-definiteness of $\boldsymbol{\Sigma}_i$:

$$\boldsymbol{\Sigma}_i = \boldsymbol{R}_i \boldsymbol{S}_i \boldsymbol{S}_i^T \boldsymbol{R}_i^T. \quad (2)$$

The 3D scene is optimized using a photometric loss function. Specifically, given an image I^i captured from viewpoint C^i, the goal is to refine the

set of Gaussians $\mathbb{G}$ such that the rendering $\mathcal{R}(\mathbb{G}|C^i)$ closely aligns with the image I^i. This alignment is evaluated through a combination of L_1 and the Structural Similarity Index Measure (SSIM) losses.

3.2. Extract information from 2D prior

The main idea behind our approach is to use all the available information from 2D images right from the start, instead of adding it piece by piece through photometric loss. We use 2D correspondences to improve the initialization and project all the known information directly into 3D. We start by selecting a reference image I^i from the training dataset. For each reference image I^i, we identify a set of neighboring images $\mathbb{I} = \{I^1, \ldots, I^j | j \in [0, J]\}$ based on camera parameters and spatial proximity to I^i. These neighboring images maximize overlap with the reference image, enhancing keypoint correspondence reliability. To identify neighboring cameras, we compute the proximity between projection matrices $\boldsymbol{P}$ using the Frobenius norm. Since the camera intrinsics are identical for one scene, we focus solely on the extrinsic parameter differences.

For each neighboring image $I^j \in \mathbb{I}$, dense correspondences relative to I^i are computed using a pretrained dense matching network denoted as $\mathcal{M}$. This network estimates dense pixel-wise correspondences between images I^i and I^j, formalized as:

$$\mathcal{M}(I^i, I^j) \rightarrow \mathcal{W}^{j \rightarrow i}, \mathbf{c}^{ij}, \tag{3}$$

where $\mathcal{W}^{j \rightarrow i} \in \mathbb{R}^{2 \times H \times W}$ is a dense warp field mapping I^j to I^i, and $\mathbf{c}^{ij} \in \mathbb{R}^{H \times W}$ quantifies correspondence confidence. Specifically, for a pixel at coordinates $(u_k^j, v_k^j) \in I^j$, the warp $\mathcal{W}^{j \rightarrow i}$ provides the corresponding pixel location in I^i via the mapping $\mathcal{W}^{j \rightarrow i}(u_k^j, v_k^j)$. For every pair of images, we extract same number of correspondences.

3.3. Splats Initialization

To accurately place 2D correspondences in 3D space, we formulate the task as a triangulation problem. The goal is to find an accurate 3D position for a new Gaussian splat $\boldsymbol{g}_k^x = (x_k, y_k, z_k)$ for each matched keypoint pair (u_k^i, v_k^i) and (u_k^j, v_k^j).

We use the projection equations for each camera, where the projection matrices $\boldsymbol{P}^i$ and $\boldsymbol{P}^j$ are 4x3

matrices that map 3D homogeneous coordinates to 2D homogeneous coordinates. The scalars w_k^i and w_k^j are normalization factors to account for the homogeneous coordinates, ensuring the consistency of the projection across cameras. Specifically:

$$\begin{cases} \begin{bmatrix} \boldsymbol{g}_k^x \\ 1 \end{bmatrix}^T \boldsymbol{P}^i = w_k^i \begin{bmatrix} u_k^i \\ v_k^i \\ 1 \end{bmatrix}^T, \\ \begin{bmatrix} \boldsymbol{g}_k^x \\ 1 \end{bmatrix}^T \boldsymbol{P}^j = w_k^j \begin{bmatrix} u_k^j \\ v_k^j \\ 1 \end{bmatrix}^T. \end{cases} \tag{4}$$

Since we normalize by the third component(third row), this gives the following equations:

$$\begin{cases} \begin{bmatrix} \boldsymbol{g}_k^x \\ 1 \end{bmatrix}^T \boldsymbol{P}_{\text{col},0}^i - u_k^i \begin{bmatrix} \boldsymbol{g}_k^x \\ 1 \end{bmatrix}^T \boldsymbol{P}_{\text{col},2}^i = 0, \\ \begin{bmatrix} \boldsymbol{g}_k^x \\ 1 \end{bmatrix}^T \boldsymbol{P}_{\text{col},1}^i - v_k^i \begin{bmatrix} \boldsymbol{g}_k^x \\ 1 \end{bmatrix}^T \boldsymbol{P}_{\text{col},2}^i = 0, \\ \begin{bmatrix} \boldsymbol{g}_k^x \\ 1 \end{bmatrix}^T \boldsymbol{P}_{\text{col},0}^j - u_k^j \begin{bmatrix} \boldsymbol{g}_k^x \\ 1 \end{bmatrix}^T \boldsymbol{P}_{\text{col},2}^j = 0, \\ \begin{bmatrix} \boldsymbol{g}_k^x \\ 1 \end{bmatrix}^T \boldsymbol{P}_{\text{col},1}^j - v_k^j \begin{bmatrix} \boldsymbol{g}_k^x \\ 1 \end{bmatrix}^T \boldsymbol{P}_{\text{col},2}^j = 0. \end{cases} \tag{5}$$

We rearrange the equations to the form $A\boldsymbol{g}_k^x = -b$, where A is constructed from the projection matrices and b being a vector of constants:

$$A^T = \begin{bmatrix} \boldsymbol{P}_{\text{col},0}^i - u_k^i \boldsymbol{P}_{\text{col},2}^i \\ \boldsymbol{P}_{\text{col},1}^i - v_k^i \boldsymbol{P}_{\text{col},2}^i \\ \boldsymbol{P}_{\text{col},0}^j - u_k^j \boldsymbol{P}_{\text{col},2}^j \\ \boldsymbol{P}_{\text{col},1}^j - v_k^j \boldsymbol{P}_{\text{col},2}^j \end{bmatrix}, \quad b = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}. \tag{6}$$

The system can be solved using the least squares method:

$$\boldsymbol{g}_k^x = \arg \min_{\boldsymbol{g}_k^x} \|A\boldsymbol{g}_k^x + b\|^2. \tag{7}$$

We then augment this solution into homogeneous coordinates for each Gaussian as:

$$\boldsymbol{g}_k^x = [x_k, y_k, z_k, 1]^T. \tag{8}$$

After determining Gaussian positions, we initialize color and scaling parameters. Colors are initialized as the average pixel values at the corresponding matched pixels (u_k^i, v_k^i) and (u_k^j, v_k^j) from the paired images I^i and I^j. The initial scale can be found using the minimum distance from the Gaussian coordinate to the nearest camera C^i or C^j. Rotation is simply an identity matrix. An example of such an initialization is provided in Fig. 2. Finally, these initialized Gaussians undergo standard photometric loss optimization to refine their parameters, correct any inaccuracies, and achieve precise, high-quality 3D reconstructions.

4. Experiments

This section provides both quantitative and qualitative evaluations of our approach. Our implementation builds upon the original 3DGS codebase [31]. All experiments were conducted on an NVIDIA A100 GPU to ensure consistent performance across methods. To maintain fairness, we obtained results for competing methods, including their training times, using the same hardware. For our approach, the initialization time is included in both Tab. 1 and Fig. 1 for comprehensive comparison.

4.1. Datasets and Metrics

We evaluate our method on three established datasets: Mip-NeRF360 [2], Tanks&Temples [34], and Deep Blending [24], which contain 9, 2, and 2 scenes, respectively. These datasets cover a mix of bounded indoor and unbounded outdoor environments with detailed backgrounds.

For evaluation, we use structural similarity (SSIM) [66], peak signal-to-noise ratio (PSNR), and perceptual similarity (LPIPS) [86] metrics on the test dataset. Following prior work [31, 81], every 8th camera view is set aside for testing. For Mip-NeRF360, we follow 3DGS [31] protocol by downsampling outdoor scenes by a factor of four and indoor scenes by a factor of two. For other datasets, we use the original resolution. Additionally, we report optimization runtime and the final number of Gaussians for each method.

4.2. Baselines

We focus on both speed and quality. Since our method can operate in different modes and supports an early stopping mechanism, we compare with representative baselines across categories. For ray-based approaches, we compare against the fast Plenoxels [83] and two advanced NeRF methods: Mip-NeRF360 [2] and Instant-NGP [49]. As our method is based on 3DGS, we also compare with the original 3DGS [31]. To ensure a fair comparison, we retrain it (denoted as 3DGS*), as this resulted in better performance than the originally reported scores. We include AbsGS[81], which focuses on improving the densification strategy, Mip-Splatting[85], a method for mitigating aliasing issues, and two high-quality baselines, 3DGS-MCMC [32] and Scaffold-GS [43]. Since our method emphasizes the initialization stage, we include RAIN-GS [29]. Notably, the mean values for Scaffold-GS and 3DGS-MCMC changed significantly, as they originally reported results for only 7 of the 9 Mip-NeRF360 scenes. Additionally, we report results for Scaffold-GS trained with the same resolution settings as 3DGS, which were not included in the original paper. We compare quality, initialization, and ray tracing categories against our model with full 30000-step convergence, denoted as *Ours 30K*.

To evaluate speed and efficiency, we compare against the fastest competitive methods: EAGLES [19], 3DGS-LM [25], Taming 3DGS [45], a fast reimplementation of 3DGS (gsplat) [80], and Mini Splatting [15], which focuses on optimizing computational budgets. We compare these methods against our model stopped at 5000 steps (*Ours 5K*).

Importantly, our approach is orthogonal to most of the methods listed in Tab. 1, as they primarily enhance computation through different means—such as modifying the optimizer [25], re-implementing the 3DGS framework [80], or improving the rasterization engine [45]. Therefore, we also report results for our method combined with Taming 3DGS.

4.3. Quantitative Evaluations

The quantitative results are presented in Tab. 1. Integrating our method with 3DGS and Taming-3DGS consistently outperforms all other techniques,

		Mip-NeRF 360					Tanks & Temples					Deep Blending				
		SSIM ↑	PSNR ↑	LPIPS ↓	Train time	#G (10^6)	SSIM ↑	PSNR↑	LPIPS↓	Train time	#G (10^6)	SSIM↑	PSNR ↑	LPIPS ↓	Train time	#G (10^6)
Rays	Plenoxels [17]	0.626	23.08	0.463	26 m	-	0.719	21.08	0.379	25 m	-	0.795	23.06	0.510	28 m	-
	INGP-Big [49]	0.699	25.59	0.331	**8 m**	-	0.745	21.92	0.305	**7 m**	-	0.817	24.96	0.390	**8 m**	-
	Mip-NeRF360 [2]	0.792	27.69	0.237	48 h	-	0.759	22.22	0.257	48 h	-	0.901	29.40	0.245	48 h	-
Quality	3D-GS [31]	0.815	27.21	0.214	42 m**	3.5	0.841	23.14	0.183	27 m**	2.0	0.903	29.41	0.243	36 m**	3.2
	3D-GS [31]*	0.816	27.49	0.215	26 m	2.8	0.853	23.76	0.169	19 m	1.6	0.908	29.77	0.242	27 m	2.6
	AbsGS-0004 [81]	0.818	27.41	0.198	20 m	3.1	0.852	23.59	0.162	14 m	3.1	0.901	29.61	0.236	20 m	1.9
	Rain-GS [29]†	0.807	22.23	0.229	32 m**	-	0.823	23.13	0.207	15 m**	-	0.900	29.42	0.255	28 m**	-
	Mip-Splatting [85]	0.838	27.97	0.179	26 m	4.0	0.859	23.81	0.156	16 m	2.4	0.903	29.35	0.239	29 m	3.6
	3DGS-MCMC [32]	**0.842**	**28.15**	0.176	20 m	3.2	0.863	24.22	0.158	13 m	1.9	0.902	29.56	0.244	19 m	2.9
	ScaffoldGS [43]	0.812	27.60	0.222	22 m	**0.6**‡	0.854	24.08	0.165	23 m	**0.6**‡	0.907	<u>30.25</u>	0.245	28 m	**0.4**‡
	Ours + 3DGS 30K steps	<u>0.840</u>	27.80	<u>0.175</u>	29 m	<u>1.9</u>	<u>0.874</u>	<u>24.45</u>	<u>0.124</u>	22 m	<u>1.4</u>	<u>0.909</u>	30.05	<u>0.219</u>	30 m	1.6
	Ours + Taming 3DGS 30K steps	0.839	<u>28.06</u>	**0.174**	16 m	3.2	**0.881**	**24.93**	**0.121**	12 m	1.9	**0.915**	**30.28**	**0.210**	14 m	2.8
Efficiency	Taming 3DGS [45]	<u>0.820</u>	**27.71**	0.207	14 m	3.2	0.856	**24.34**	0.164	9 m	1.9	0.907	29.54	0.237	12 m	2.8
	3DGS+3DGS-LM [25]†	0.813	27.39	0.221	16 m	2.8*	0.845	<u>23.73</u>	0.182	12 m	1.6*	0.903	29.72	0.247	16 m	2.6*
	gsplat [80]	0.818	<u>27.51</u>	0.215	18 m	3.1	0.845	23.57	0.170	13 m	2.8	0.904	29.57	0.237	15 m	2.8
	EAGLES [19]	0.809	27.20	0.232	16 m	<u>1.3</u>	0.837	23.26	0.201	10 m	<u>0.7</u>	**0.910**	<u>29.85</u>	0.246	18 m	<u>1.2</u>
	MiniSplatting [15]	<u>0.820</u>	27.25	0.217	12 m	**0.5**	0.836	23.21	0.203	12 m	**0.3**	0.908	29.98	0.253	<u>8 m</u>	0.4
	Ours + 3DGS 5K steps	<u>0.820</u>	26.70	<u>0.202</u>	8 m	2.9	<u>0.860</u>	22.95	<u>0.164</u>	9 m	2.2	<u>0.909</u>	29.46	<u>0.231</u>	10 m	2.2
	Ours + Taming 3DGS 5K steps	**0.825**	26.89	**0.195**	6 m	2.8	**0.864**	23.08	**0.160**	4 m	1.6	**0.910**	29.46	**0.228**	4 m	2.4

Table 1. Quantitative evaluations across the Mip-NeRF 360 [2], Tanks&Temples [34], and Deep Blending [24] datasets. We assess quality using PSNR, SSIM, and LPIPS, while resource efficiency is measured by training time and, where applicable, the final number of Gaussians (#G). The **best** and <u>second-best</u> results are highlighted for each metric. Note that the reported training time for our method includes initialization, whereas, for other methods except the initialization category, we report only the training time. † indicates that results were taken directly from the paper, as the code is either not publicly available or not functioning. ‡ for ScaffoldGS denotes the number of anchors, not splats. * indicates that for 3DGS-LM, we assume the number of Gaussians is the same as in the original 3DGS, as the method uses the same densification strategy. ** denotes results reported for an NVIDIA A6000, while ** corresponds to results for an NVIDIA RTX 3090. Please refer to supplementary materials for per-scene scores.

demonstrating its effectiveness in enhancing reconstruction quality. Our models, trained for 30,000 steps, surpass quality-focused approaches, while the same models trained for just 5,000 steps achieve faster performance than efficiency-focused methods, matching them in evaluation metrics. Notably, the efficiency of our model stems from its improved initialization rather than optimizing computational steps. This initialization-based approach is compatible with other techniques listed in the second half of Tab. 1. For instance, applying our initialization to Taming-3DGS significantly boosts its performance.

Our method shows particularly strong improvements in SSIM and LPIPS scores compared to PSNR. We attribute this to the fact that our model is less suited for handling reflective surfaces, where the same physical location may radiate different colors depending on the viewpoint. Since our color prediction relies on input viewpoints and detected correspondences, it may struggle with such scenarios. Nonetheless, our approach excels in overall reconstruction quality and efficiency, offering a ro-

bust enhancement to existing 3DGS pipelines.

4.4. Qualitative Evaluations

In Fig. 3, our approach shows clear improvements over other methods on images sampled from Mip-NeRF360 [2], Deep Blending [24], and Tank & Temples [34]. For qualitative evaluation, we compare our approach based on 3DGS without densification to state-of-the-art quality-focused methods, as this provides a more meaningful comparison than benchmarking against the 3DGS baseline. We have cropped regions of interest for the main paper; full-scale results are in the supplementary material. The examples show that our model excels not only in high-frequency regions—such as small stones near railroad tracks, grass, or concrete textures—but also in capturing fine details like flower stems (first row) and distant elements like roads (third row). Other models often fail to reconstruct these details accurately, either blurring them or introducing high-frequency artifacts. EDGS dense initialization ensures a Gaussian splat is placed at every meaningful
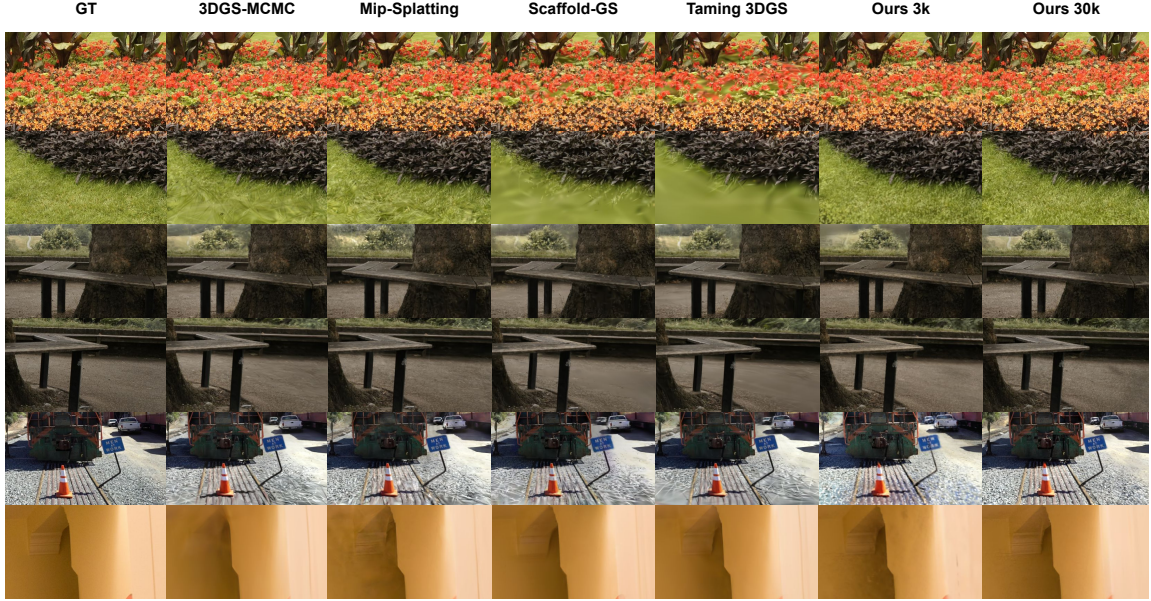
Figure 3. Qualitative comparison on *flowers* and *treehill* from Mip-NeRF360 [2], *train* from Tank & Temples [34] and *Playroom* from Deep Blending [24]. For this visualization, we crop regions of interest. See supplementary materials for full renderings. Our model effectively reduces blur and preserves fine details that other methods often overlook or blur. It also performs comparably to or better than state-of-the-art methods, achieving faster convergence. For comparison, we additionally provide renderings of these cropped regions for our model with 3DGS trained for only 3,000 steps.

Initialization Type	PSNR↑	SSIM↑	LPIPS↓
3DGS (Random Init, w/D)	22.19	0.704	0.313
3DGS ($\mathcal{M}$=COLMAP, w/ D.)	27.49	0.816	0.215
Ours ($\mathcal{M}$=RoMa, w/o D.)	27.80	**0.840**	0.175
Ours ($\mathcal{M}$=RoMa, w/ D.)	**27.84**	0.841	**0.173**
Depth [22] Init (w/ D.)	27.15	0.818	0.198
Depth [22] Init (w/o D.)	26.75	0.807	0.209

Table 2. Impact of densification on different initialization methods. While densification can improve performance of our model, especially in poorly initialized scenes, it significantly increases the number of Gaussians.

Figure 4. Extreme viewpoint rendering. EDGS (right) better preserves details and reduces stretched Gaussians when rendering from viewpoints far outside the training set compared to the 3DGS (left). This results in a more consistent distribution and improved quality, especially in challenging regions like the building and flower pot.

location, enabling precise and detailed reconstruction. We also provide crops for our model with 3000 steps, showing that we achieve comparable perceptual quality much faster than other methods.

Extreme Viewpoint Rendering. Our model effectively handles extreme viewpoint variations, outperforming the baseline when rendering from camera angles far outside the training set. As shown in Fig. 4, our dense initialization prevents the need for stretching small Gaussians to compensate for pixel loss at a distance, resulting in a more stable and accurate reconstruction. As visualized for *garden*
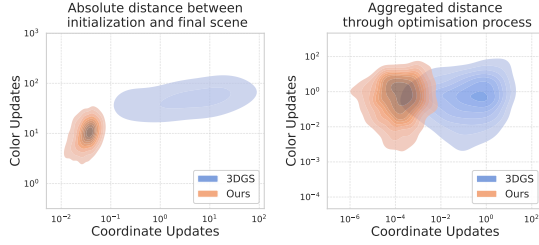
Figure 5. Distributions of 3DGS parameters change in color/coordinate space throughout training. We compare color $\boldsymbol{g}_i^c$ and coordinate $\boldsymbol{g}_i^x$ changes from initialization to 30K steps. Our method not only initializes closer to the solution (right chart) but also requires significantly fewer adjustments (left chart) through the ptimisation process, leading to faster and more stable convergence.

scene from the Mip-NeRF360 dataset, our method avoids large Gaussians and exhibits less noise compared to the competing approach.

4.5. Ablation Studies

Gaussian Motion and Convergence. We study the distance traveled by each Gaussian during optimization. Fig. 5 presents the start-to-finish displacement and full motion path length. Namely, we analyze how Gaussian coordinate and color parameters evolve during the optimization process by measuring two key distributions. Let $\boldsymbol{g}_i(t)$ denote the state of Gaussian $\boldsymbol{g}_i$ at optimization step t for $i \in \{1, \ldots, N\}$. The first distribution captures the *absolute travel distance*, defined as:

$$\begin{pmatrix} \|\boldsymbol{g}_i^c(0) - \boldsymbol{g}_i^c(T)\|_2 \\ \|\boldsymbol{g}_i^x(0) - \boldsymbol{g}_i^x(T)\|_2 \end{pmatrix} \in \mathbb{R}^2. \qquad (9)$$

The second distribution measures the full *trail path length*, computed as:

$$\begin{pmatrix} \sum_{t=0}^{T} \|\boldsymbol{g}_i^c(t) - \boldsymbol{g}_i^c(t+1)\|_2 \\ \sum_{t=0}^{T} \|\boldsymbol{g}_i^x(t) - \boldsymbol{g}_i^x(t+1)\|_2 \end{pmatrix} \in \mathbb{R}^2, \qquad (10)$$

where T denotes the number of optimization steps. Our method significantly reduces the final coordinate displacement, as Gaussians are initialized closer to surfaces, requiring fewer adjustments.

Matching Algorithm	PSNR↑	SSIM↑	LPIPS↓
Ours ($\mathcal{M}$=RoMa)	**27.80**	**0.840**	**0.175**
LoFTR [60] Init	27.71	0.828	0.185
DKM [12] Init	27.69	0.829	0.190
RAFT [62] Init	26.98	0.802	0.218

Table 3. Comparison of different matching algorithms. While RoMa, LoFTR, and DKM perform similarly, RAFT struggles since it was primarily designed for optical flow between consecutive video frames.

However, color changes remain necessary since the initialization provides only an approximate color match. Compared to 3DGS, our model reduces the final coordinate travel distance by 50 times, and the total path length in coordinates is 30 times shorter. The color path length also decreases, though less dramatically, by approximately a factor of two, as small oscillations remain along the trajectory. For a more detailed analysis, we provide videos of Gaussian motion in our supplementary material.

Matching Algorithm Comparison. We evaluate various image matching methods $\mathcal{M}$ for initializing our splats. Throughout this paper, we use RoMa [13] as our primary matching algorithm, but we also experiment with LoFTR [60], DKM [12], and RAFT [62]. While RoMa, LoFTR, and DKM yield comparable performance, RAFT struggles due to its primary design for optical flow in consecutive video frames, where viewpoint differences are minimal. In addition to feature matching, we evaluate an alternative depth-based initialization using DepthFM [22]. However, monocular depth estimates suffer from scale inconsistencies even across neighboring views, leading to worse performance. While DepthFM performs better than the baseline 3DGS (COLMAP initialization + Densification), it remains less effective than our matching-based approach. See Tabs. 2 and 3 for a detailed comparison of performance on the Mip-NeRF360 [2] dataset.

Effect of Densification. We analyze the impact of densification across different initialization strategies, including our method, depth-projected splats, COLMAP, and NeRF-based initialization. While densification can be beneficial in cases where initialization is sparse—such as *treehill* scene in Mip-

NeRF360, where distant regions are underrepresented—it significantly increases the number of Gaussians, making optimization less efficient and harder to control. See Tab. 2 for detailed results.

Robustness to Noise. Our model can tolerate inaccuracies in the initial matches arising from errors in the triangulation process or suboptimal matches from $\mathcal{M}$. To evaluate this robustness, we introduce Gaussian noise $\epsilon \sim \mathcal{N}(0, \sigma)$ to either the coordinates or color of the initialized splats and analyze how final performance changes for varying noise levels σ. Fig. 6 presents the effect of increasing noise on PSNR and LPIPS. Interestingly, our model demonstrates greater robustness to color noise than to coordinate noise, reinforcing our claim that the primary advantage of our approach lies in reducing Gaussian movement during optimization. Despite the added perturbations, performance remains stable for moderate noise levels. Noise ϵ is applied separately to the color parameter $\boldsymbol{g}_i^c$ and the coordinate parameter $\boldsymbol{g}_i^x$. See supplementary material for visualizations of noisy scenes, further illustrating our model's resilience. Notably, our method remains stable even with small amounts of added noise, likely because the initialization itself is already inherently noisy, as shown in Fig. 2. All experiments are conducted on the Mip-NeRF360 dataset.
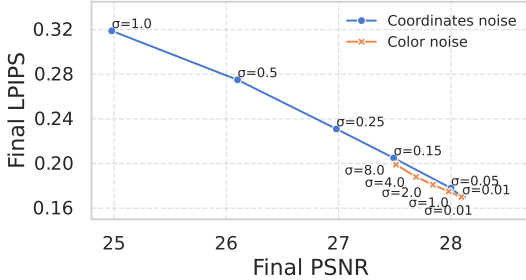

Figure 6. The effect of adding noise $\mathcal{N}(0, \sigma)$ to our model. Please note that noise scale σ is higher for color noise. In the supplementary, we provide images visualizing the effect of noise on the initialization quality.

Hyperparameter Sensitivity. We evaluate the impact of key hyperparameters, including the number of reference frames and matches sampled per view. Specifically, we analyze the effect of varying the
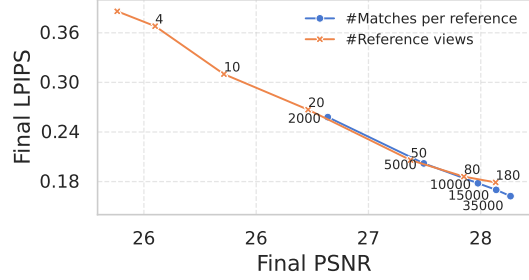

Figure 7. Impact of hyperparameters on final performance. We visualize the importance of sampling a sufficient number of reference frames(orange) and having a sufficient number of points sampled from each viewpoint(blue).

number of reference cameras and sampled keypoints in Fig. 7. Increasing these values beyond a certain point yields diminishing returns, leading us to select 15000 keypoints per reference frame and 180 reference cameras as a balance between performance and computational cost. Regarding the number of nearest neighbors used for match sampling, we observe that while increasing this number significantly affects the initialization time (as more matches need to be computed), its impact on final performance is minimal. The supplementary material provides additional visualization for this hyperparameter.

5. Conclusion

We propose a novel initialization strategy for 3D Gaussian Splatting that directly triangulates dense 2D keypoints into 3D space. Unlike conventional iterative densification, our method begins with a high-density Gaussian distribution, resulting in faster convergence and improved reconstruction quality.

This initialization reduces the distance each Gaussian must travel in parameter space, accelerating convergence and guiding optimization toward lower-error reconstructions. EDGS exceeds the performance of both speed-optimized and quality-focused 3DGS approaches while requiring six times fewer optimization steps and approximately 40% fewer Gaussians. EDGS also integrates with existing acceleration techniques, offering an efficient upgrade for high-quality 3D reconstruction.

Acknowledgement

References

[1] Hendrik Baatz, Jonathan Granskog, Marios Papas, Fabrice Rousselle, and Jan Novák. Nerf-tex: Neural reflectance field textures. *Computer Graphics Forum*, 41, 2022. 2

[2] Jonathan T Barron, Ben Mildenhall, Dor Verbin, Pratul P Srinivasan, and Peter Hedman. Mip-nerf 360: Unbounded anti-aliased neural radiance fields. In *CVPR*, 2022. 1, 2, 4, 6, 7, 8, 9

[3] Samuel Rota Bulò, Lorenzo Porzi, and Peter Kontschieder. Revising densification in gaussian splatting. *arXiv preprint arXiv:2404.06109*, 2024. 2, 3

[4] Jiafu Chen, Boyan Ji, Zhanjie Zhang, Tianyi Chu, Zhiwen Zuo, Lei Zhao, Wei Xing, and Dongming Lu. Testnerf: Text-driven 3d style transfer via cross-modal learning. In *International Joint Conference on Artificial Intelligence*, 2023. 2

[5] Yuedong Chen, Haofei Xu, Chuanxia Zheng, Bohan Zhuang, Marc Pollefeys, Andreas Geiger, Tat-Jen Cham, and Jianfei Cai. Mvsplat: Efficient 3d gaussian splatting from sparse multi-view images. In *European Conference on Computer Vision*, pages 370–386. Springer, 2025. 3

[6] Zilong Chen, Feng Wang, Yikai Wang, and Huaping Liu. Text-to-3d using gaussian splatting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 21401–21412, 2024. 3

[7] Kai Cheng, Xiaoxiao Long, Kaizhi Yang, Yao Yao, Wei Yin, Yuexin Ma, Wenping Wang, and Xuejin

Chen. Gaussianpro: 3d gaussian splatting with progressive propagation. In *Forty-first International Conference on Machine Learning*, 2024. 2

[8] Amaury Dame, Victor A. Prisacariu, Carl Y. Ren, and Ian Reid. Dense reconstruction using 3d object shape priors. In *2013 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1288–1295, 2013. 2

[9] Robert A. Drebin, Loren Carpenter, and Pat Hanrahan. Volume rendering. In *Proceedings of the 15th Annual Conference on Computer Graphics and Interactive Techniques*, page 65–74, New York, NY, USA, 1988. Association for Computing Machinery. 3

[10] Yuanxing Duan, Fangyin Wei, Qiyu Dai, Yuhang He, Wenzheng Chen, and Baoquan Chen. 4d gaussian splatting: Towards efficient novel view synthesis for dynamic scenes. *arXiv preprint arXiv:2402.03307*, 2024. 3

[11] Sankeerth Durvasula, Adrian Zhao, Fan Chen, Ruofan Liang, Pawan Kumar Sanjaya, and Nandita Vijaykumar. Distwar: Fast differentiable rendering on raster-based rendering pipelines. *arXiv preprint arXiv:2401.05345*, 2023. 4

[12] Johan Edstedt, Ioannis Athanasiadis, Mårten Wadenbäck, and Michael Felsberg. DKM: Dense kernelized feature matching for geometry estimation. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2023. 9

[13] Johan Edstedt, Qiyu Sun, Georg Bökman, Mårten Wadenbäck, and Michael Felsberg. RoMa: Robust Dense Feature Matching, 2023. arXiv:2305.15404 [cs]. 9

[14] Zhiwen Fan, Wenyan Cong, Kairun Wen, Kevin Wang, Jian Zhang, Xinghao Ding, Danfei Xu, Boris Ivanovic, Marco Pavone, Georgios Pavlakos, et al. Instantsplat: Unbounded sparse-view posefree gaussian splatting in 40 seconds. *arXiv preprint arXiv:2403.20309*, 2, 2024. 3

[15] Guangchi Fang and Bing Wang. Mini-splatting: Representing scenes with a constrained number of gaussians. In *European Conference on Computer Vision*, 2024. 1, 3, 6, 7

[16] Guofeng Feng, Siyan Chen, Rong Fu, Zimu Liao, Yi Wang, Tao Liu, Zhilin Pei, Hengjie Li, Xingcheng Zhang, and Bo Dai. Flashgs: Efficient 3d gaussian splatting for large-scale and high-resolution rendering. *arXiv preprint arXiv:2408.07967*, 2024. 4

[17] Sara Fridovich-Keil, Alex Yu, Matthew Tancik, Qinhong Chen, Benjamin Recht, and Angjoo Kanazawa.

Plenoxels: Radiance fields without neural networks. In *CVPR*, 2022. 7

[18] 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*, pages 14346–14355, 2021. 2, 3

[19] Sharath Girish, Kamal Gupta, and Abhinav Shrivastava. Eagles: Efficient accelerated 3d gaussians with lightweight encodings. *arXiv preprint arXiv:2312.04564*, 2023. 3, 6, 7

[20] Leonardo Gomes, Luciano Silva, and Olga Bellon. 3d reconstruction methods for digital preservation of cultural heritage: A survey. *Pattern Recognition Letters*, 50, 2014. 2

[21] Antoine Guédon and Vincent Lepetit. Sugar: Surface-aligned gaussian splatting for efficient 3d mesh reconstruction and high-quality mesh rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5354–5363, 2024. 2, 3

[22] Ming Gui, Johannes S. Fischer, Ulrich Prestel, Pingchuan Ma, Dmytro Kotovenko, Olga Grebenkova, Stefan Andreas Baumann, Vincent Tao Hu, and Björn Ommer. DepthFM: Fast Monocular Depth Estimation with Flow Matching, 2024. arXiv:2403.13788 [cs]. 8, 9

[23] Richard Hartley and Andrew Zisserman. *Multiple view geometry in computer vision*. Cambridge university press, 2003. 2

[24] Peter Hedman, Julien Philip, True Price, Jan-Michael Frahm, George Drettakis, and Gabriel Brostow. Deep blending for free-viewpoint image-based rendering. *ACM Transactions on Graphics (ToG)*, 37(6):1–15, 2018. 2, 6, 7, 8

[25] Lukas Höllein, Aljaž Božič, Michael Zollhöfer, and Matthias Nießner. 3dgs-lm: Faster gaussian-splatting optimization with levenberg-marquardt. *arXiv preprint arXiv:2409.12892*, 2024. 1, 4, 6, 7

[26] Binbin Huang, Zehao Yu, Anpei Chen, Andreas Geiger, and Shenghua Gao. 2d gaussian splatting for geometrically accurate radiance fields. In *ACM SIGGRAPH 2024 Conference Papers*, pages 1–11, 2024. 3

[27] Muhammad Zubair Irshad, Sergey Zakharov, Katherine Liu, Vitor Guizilini, Thomas Kollar, Adrien Gaidon, Zsolt Kira, and Rares Ambrus. Neo 360: Neural fields for sparse view synthesis of outdoor scenes. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9187–9198, 2023. 3

[28] Joel Janai, Fatma Güney, Aseem Behl, and Andreas Geiger. Computer vision for autonomous vehicles: Problems, datasets and state-of-the-art. *Found. Trends Comput. Graph. Vis.*, 12:1–308, 2017. 2

[29] Jaewoo Jung, Jisang Han, Honggyu An, Jiwon Kang, Seonghoon Park, and Seungryong Kim. Relaxing accurate initialization constraint for 3d gaussian splatting. *arXiv preprint arXiv:2403.09413*, 2024. 4, 6, 7

[30] Kai Katsumata, Duc Minh Vo, and Hideki Nakayama. An efficient 3d gaussian representation for monocular/multi-view dynamic scenes. *arXiv preprint arXiv:2311.12897*, 2023. 3

[31] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Trans. Graph.*, 42(4):139–1, 2023. 1, 2, 3, 4, 6, 7, 8

[32] Shakiba Kheradmand, Daniel Rebain, Gopal Sharma, Weiwei Sun, Yang-Che Tseng, Hossam Isack, Abhishek Kar, Andrea Tagliasacchi, and Kwang Moo Yi. 3d gaussian splatting as markov chain monte carlo. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024. Spotlight Presentation. 2, 3, 4, 6, 7

[33] Mijeong Kim, Seonguk Seo, and Bohyung Han. Infonerf: Ray entropy minimization for few-shot neural volume rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12912–12921, 2022. 3

[34] Arno Knapitsch, Jaesik Park, Qian-Yi Zhou, and Vladlen Koltun. Tanks and temples: Benchmarking large-scale scene reconstruction. *ACM Transactions on Graphics (ToG)*, 36(4):1–13, 2017. 6, 7, 8

[35] Muhammed Kocabas, Jen-Hao Rick Chang, James Gabriel, Oncel Tuzel, and Anurag Ranjan. Hugs: Human gaussian splats. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 505–515, 2024. 3

[36] Dmytro Kotovenko, Olga Grebenkova, Nikolaos Sarafianos, Avinash Paliwal, Pingchuan Ma, Omid Poursaeed, Sreyas Mohan, Yuchen Fan, Yilei Li, Rakesh Ranjan, and Björn Ommer. Wast-3d: Wasserstein-2 distance for scene-to-scene stylization on 3d gaussians. In *Computer Vision – ECCV 2024*, pages 298–314, Cham, 2025. Springer Nature Switzerland. 2

[37] Joo Chan Lee, Daniel Rho, Xiangyu Sun, Jong Hwan Ko, and Eunbyung Park. Compact 3d

gaussian representation for radiance field. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 21719–21728, 2024. 3

[38] Jiahui Lei, Yufu Wang, Georgios Pavlakos, Lingjie Liu, and Kostas Daniilidis. Gart: Gaussian articulated template models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 19876–19887, 2024. 3

[39] Marc Levoy. Efficient ray tracing of volume data. *TOG*, 1990. 3

[40] Shaopeng Li, Daqiao Zhang, Yong Xian, Bangjie Li, Tao Zhang, and Chengliang Zhong. Overview of deep learning application on visual slam. *Displays*, 74:102298, 2022. 2

[41] Lingjie Liu, Jiatao Gu, Kyaw Zaw Lin, Tat-Seng Chua, and Christian Theobalt. Neural sparse voxel fields. *Advances in Neural Information Processing Systems*, 33:15651–15663, 2020. 3

[42] Guanxing Lu, Shiyi Zhang, Ziwei Wang, Changliu Liu, Jiwen Lu, and Yansong Tang. Manigaussian: Dynamic gaussian splatting for multi-task robotic manipulation. In *European Conference on Computer Vision*, pages 349–366. Springer, 2024. 2

[43] Tao Lu, Mulin Yu, Linning Xu, Yuanbo Xiangli, Limin Wang, Dahua Lin, and Bo Dai. Scaffold-gs: Structured 3d gaussians for view-adaptive rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 20654–20664, 2024. 3, 6, 7

[44] Jonathon Luiten, Georgios Kopanas, Bastian Leibe, and Deva Ramanan. Dynamic 3d gaussians: Tracking by persistent dynamic view synthesis. *arXiv preprint arXiv:2308.09713*, 2023. 3

[45] Saswat Subhajyoti Mallick, Rahul Goel, Bernhard Kerbl, Francisco Vicente Carrasco, Markus Steinberger, and Fernando De La Torre. Taming 3dgs: High-quality radiance fields with limited resources. *arXiv preprint arXiv:2406.15643*, 2024. 1, 2, 4, 6, 7

[46] Nelson Max. Optical models for direct volume rendering. *TVCG*, 1995. 3

[47] Nelson Max and Min Chen. Local and global illumination in the volume rendering integral. Technical report, Lawrence Livermore National Lab.(LLNL), Livermore, CA (United States), 2005. 3

[48] 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*, 65(1):99–106, 2021. 2, 3

[49] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM transactions on graphics (TOG)*, 41(4):1–15, 2022. 2, 3, 6, 7

[50] KL Navaneet, Kossar Pourahmadi Meibodi, Soroush Abbasi Koohpayegani, and Hamed Pirsiavash. Compact3d: Compressing gaussian splat radiance field models with vector quantization. *arXiv preprint arXiv:2311.18159*, 2023. 3

[51] Simon Niedermayr, Josef Stumpfegger, and Rüdiger Westermann. Compressed 3d gaussian splatting for accelerated novel view synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10349–10358, 2024. 3

[52] Michael Niemeyer, Jonathan T Barron, Ben Mildenhall, Mehdi SM Sajjadi, Andreas Geiger, and Noha Radwan. Regnerf: Regularizing neural radiance fields for view synthesis from sparse inputs. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5480–5490, 2022. 3

[53] Michael Niemeyer, Fabian Manhardt, Marie-Julie Rakotosaona, Michael Oechsle, Daniel Duckworth, Rama Gosula, Keisuke Tateno, John Bates, Dominik Kaeser, and Federico Tombari. Radsplat: Radiance field-informed gaussian splatting for robust real-time rendering with 900+ fps. *arXiv.org*, 2024. 4

[54] Onur Ozyesil, Vladislav Voroninski, Ronen Basri, and Amit Singer. A survey of structure from motion, 2017. 2

[55] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 652–660, 2017. 2

[56] Christian Reiser, Songyou Peng, Yiyi Liao, and Andreas Geiger. Kilonerf: Speeding up neural radiance fields with thousands of tiny mlps. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 14335–14345, 2021. 3

[57] Shunsuke Saito, Gabriel Schwartz, Tomas Simon, Junxuan Li, and Giljoo Nam. Relightable gaussian codec avatars. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 130–141, 2024. 2, 3

[58] Johannes L Schonberger and Jan-Michael Frahm. Structure-from-motion revisited. In *Proceedings of*

the IEEE conference on computer vision and pattern recognition, pages 4104–4113, 2016. 2, 4

[59] Cheng Sun, Min Sun, and Hwann-Tzong Chen. Direct voxel grid optimization: Super-fast convergence for radiance fields reconstruction. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 5459–5469, 2022. 3

[60] Jiaming Sun, Zehong Shen, Yuang Wang, Hujun Bao, and Xiaowei Zhou. LoFTR: Detector-free local feature matching with transformers. *CVPR*, 2021. 9

[61] Jiaxiang Tang, Jiawei Ren, Hang Zhou, Ziwei Liu, and Gang Zeng. Dreamgaussian: Generative gaussian splatting for efficient 3d content creation. *arXiv preprint arXiv:2309.16653*, 2023. 3

[62] Zachary Teed and Jia Deng. Raft: Recurrent all-pairs field transforms for optical flow. In *European Conference on Computer Vision*, 2020. 9

[63] Guangcong Wang, Zhaoxi Chen, Chen Change Loy, and Ziwei Liu. Sparsenerf: Distilling depth ranking for few-shot novel view synthesis. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9065–9076, 2023. 3

[64] Peng Wang, Lingjie Liu, Yuan Liu, Christian Theobalt, Taku Komura, and Wenping Wang. Neus: Learning neural implicit surfaces by volume rendering for multi-view reconstruction. *arXiv preprint arXiv:2106.10689*, 2021. 3

[65] Yiming Wang, Qin Han, Marc Habermann, Kostas Daniilidis, Christian Theobalt, and Lingjie Liu. Neus2: Fast learning of neural implicit surfaces for multi-view reconstruction. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3295–3306, 2023. 3

[66] Zhou 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*, 13(4):600–612, 2004. 6

[67] Guanjun Wu, Taoran Yi, Jiemin Fang, Lingxi Xie, Xiaopeng Zhang, Wei Wei, Wenyu Liu, Qi Tian, and Xinggang Wang. 4d gaussian splatting for real-time dynamic scene rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 20310–20320, 2024. 3

[68] Yuxi Xiao, Nan Xue, Tianfu Wu, and Gui-Song Xia. Level-s ^2 fm: Structure from motion on neural level set of implicit surfaces. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 17205–17214, 2023. 3

[69] Tianyi Xie, Zeshun Zong, Yuxing Qiu, Xuan Li, Yutao Feng, Yin Yang, and Chenfanfu Jiang. Physgaussian: Physics-integrated 3d gaussians for generative dynamics. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4389–4398, 2024. 3

[70] Dejia Xu, Yifan Jiang, Peihao Wang, Zhiwen Fan, Humphrey Shi, and Zhangyang Wang. Sinnerf: Training neural radiance fields on complex scenes from a single image. In *European Conference on Computer Vision*, pages 736–753. Springer, 2022. 3

[71] Haofei Xu, Songyou Peng, Fangjinhua Wang, Hermann Blum, Daniel Barath, Andreas Geiger, and Marc Pollefeys. Depthsplat: Connecting gaussian splatting and depth. *arXiv preprint arXiv:2410.13862*, 2024. 3

[72] Chi Yan, Delin Qu, Dan Xu, Bin Zhao, Zhigang Wang, Dong Wang, and Xuelong Li. Gs-slam: Dense visual slam with 3d gaussian splatting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 19595–19604, 2024. 2

[73] Zhiwen Yan, Weng Fei Low, Yu Chen, and Gim Hee Lee. Multi-scale 3d gaussian splatting for anti-aliased rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 20923–20931, 2024. 3

[74] Zeyu Yang, Hongye Yang, Zijie Pan, Xiatian Zhu, and Li Zhang. Real-time photorealistic dynamic scene representation and rendering with 4d gaussian splatting. *arXiv preprint arXiv:2310.10642*, 2023. 3

[75] Ziyi Yang, Xinyu Gao, Yangtian Sun, Yihua Huang, Xiaoyang Lyu, Wen Zhou, Shaohui Jiao, Xiaojuan Qi, and Xiaogang Jin. Spec-gaussian: Anisotropic view-dependent appearance for 3d gaussian splatting. *arXiv preprint arXiv:2402.15870*, 2024. 3

[76] Ziyi Yang, Xinyu Gao, Wen Zhou, Shaohui Jiao, Yuqing Zhang, and Xiaogang Jin. Deformable 3d gaussians for high-fidelity monocular dynamic scene reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 20331–20341, 2024. 3

[77] Lior Yariv, Jiatao Gu, Yoni Kasten, and Yaron Lipman. Volume rendering of neural implicit surfaces. *Advances in Neural Information Processing Systems*, 34:4805–4815, 2021. 3

[78] Lior Yariv, Peter Hedman, Christian Reiser, Dor Verbin, Pratul P Srinivasan, Richard Szeliski, Jonathan T Barron, and Ben Mildenhall. BakedSDF: Meshing Neural SDFs for Real-Time View Synthesis. In *SIGGRAPH*, 2023. 2

14

[79] Mingqiao Ye, Martin Danelljan, Fisher Yu, and Lei Ke. Gaussian grouping: Segment and edit anything in 3d scenes. *arXiv preprint arXiv:2312.00732*, 2023. 3

[80] Vickie Ye, Ruilong Li, Justin Kerr, Matias Turkulainen, Brent Yi, Zhuoyang Pan, Otto Seiskari, Jianbo Ye, Jeffrey Hu, Matthew Tancik, and Angjoo Kanazawa. gsplat: An open-source library for Gaussian splatting. *arXiv preprint arXiv:2409.06765*, 2024. 4, 6, 7

[81] Zongxin Ye, Wenyu Li, Sidun Liu, Peng Qiao, and Yong Dou. Absgs: Recovering fine details in 3d gaussian splatting. In *ACM Multimedia 2024*, 2024. 2, 3, 6, 7

[82] Taoran Yi, Jiemin Fang, Guanjun Wu, Lingxi Xie, Xiaopeng Zhang, Wenyu Liu, Qi Tian, and Xinggang Wang. Gaussiandreamer: Fast generation from text to 3d gaussian splatting with point cloud priors. *arXiv preprint arXiv:2310.08529*, 2023. 3

[83] Alex Yu, Ruilong Li, Matthew Tancik, Hao Li, Ren Ng, and Angjoo Kanazawa. Plenoctrees for real-time rendering of neural radiance fields. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5752–5761, 2021. 2, 3, 6

[84] Alex Yu, Vickie Ye, Matthew Tancik, and Angjoo Kanazawa. pixelnerf: Neural radiance fields from one or few images. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4578–4587, 2021. 2, 3

[85] Zehao Yu, Anpei Chen, Binbin Huang, Torsten Sattler, and Andreas Geiger. Mip-splatting: Alias-free 3d gaussian splatting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 19447–19456, 2024. 3, 6, 7

[86] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 586–595, 2018. 6

[87] Zheng Zhang, Wenbo Hu, Yixing Lao, Tong He, and Hengshuang Zhao. Pixel-gs: Density control with pixel-aware gradient for 3d gaussian splatting. *arXiv preprint arXiv:2403.15530*, 2024. 2

[88] Wojciech Zielonka, Timur Bagautdinov, Shunsuke Saito, Michael Zollhöfer, Justus Thies, and Javier Romero. Drivable 3d gaussian avatars. *arXiv preprint arXiv:2311.08581*, 2023. 2, 3

[89] Zi-Xin Zou, Zhipeng Yu, Yuan-Chen Guo, Yangguang Li, Ding Liang, Yan-Pei Cao, and Song-Hai Zhang. Triplane meets gaussian splatting: Fast and generalizable single-view 3d reconstruction with transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10324–10335, 2024. 3, 4

[90] Matthias Zwicker, Hanspeter Pfister, Jeroen Van Baar, and Markus Gross. Ewa splatting. *IEEE Transactions on Visualization and Computer Graphics*, 8(3):223–238, 2002. 4

15

EDGS: Eliminating Densification for Efficient Convergence of 3DGS –Supplementary Materials–

A. Implementation details

After the initialization phase, we follow the default optimization protocol of 3DGS, with densification disabled and without gradient aggregation for detection of under-reconstructed regions. This ensures a controlled setting to isolate the impact of our initialization. For fair comparison, all models are trained for the same number of iterations as competing methods, except for two models that were stopped early at 5000 steps (*Ours + 3DGS 5K, Ours + Taming 3DGS 5K*). All experiments were conducted on an NVIDIA A100 GPU with 80GB of memory, though our method required only 15GB of GPU memory at peak usage.

Our method is also compatible with front-facing scenes. In a public demo, we show that selecting only 16 random frames from a video is sufficient for rapid convergence, demonstrating the efficiency and robustness of the method even in sparse-view scenarios.

B. Visual results

Full-resolution versions of the renders shown in the main paper are provided in Figs. A1 to A3. For clearer comparison in Fig. A3, we also include renderings from the original 3DGS method.

C. Impact of Nearest Neighbors parameter

Once we have computed dense keypoint correspondences between the reference image I^i and its neighboring images I^j, we need to aggregate them to obtain a robust initialization. In Fig. A4, we visualize keypoint detection confidence across different neighbors. The reference image (referred to as the source) is shown in the top-left, followed by rows 2 through 5, where we display ground truth views (left) of the nearest cameras (in terms of extrinsics) and the corresponding confidence maps $\mathbf{c}^{ij}$ (right), indicating which regions in I^i were matched with each neighbor I^j. It is evident that each neighbor overlaps only partially with the reference view, motivating the need to aggregate confidence scores on a per-pixel level. The resulting aggregated confidence map $\mathbf{c}^i$ is shown in the top-right corner and is used to sample keypoints. This process is illustrated using the *treehill* scene from the Mip-NeRF360 [2] dataset.

While matching with more neighbors increases the total number of reference correspondences, the marginal gain decreases with each additional neighbor due to significant overlap among the sets of matched keypoints. Meanwhile, the initialization time grows linearly with the number of neighbors. Therefore, instead of aggregating over many neighbors per reference image, we achieve better efficiency by sampling more reference views and selecting only the top-1 nearest neighbor for each.

D. Number of gaussians

In Fig. A5, we illustrate the effect of densification as the relative number of Gaussian splats over optimization steps, averaged across multiple scenes and normalized by the final count in the original 3DGS [31]. Our method, even without densification, needs 40% fewer Gaussians, while converging faster and maintaining performance comparable to the version with densification. Although densification offers slight improvements in under-initialized regions, it nearly doubles the number of splats, leading to increased computational overhead and reduced controllability.
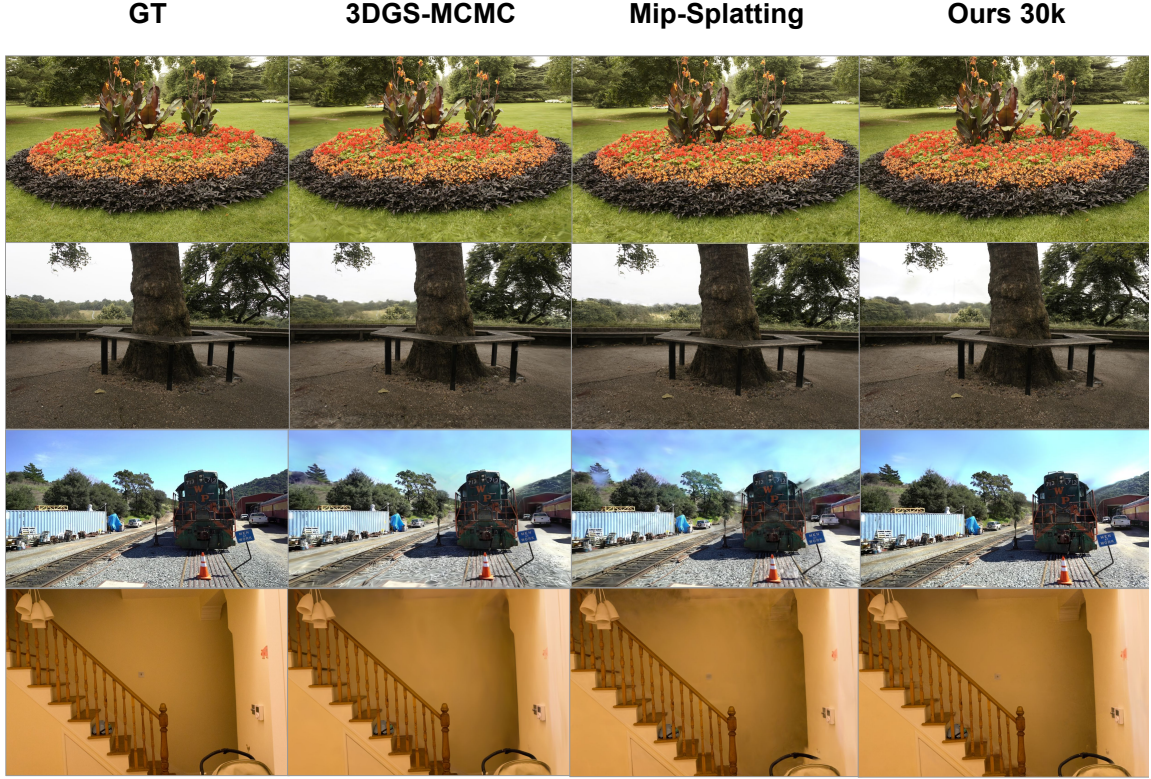
Figure A1. Additional qualitative results are presented for the scenes *treehill*, *flowers*, *train* and *playroom*. For clarity, areas of interest have been zoomed in Fig. 3. These results are best viewed digitally for optimal detail.

E. Undercovered regions

To ensure complete coverage of the scene, it is crucial to sample keypoints from multiple image pairs. Without this, certain regions may remain underrepresented, making it difficult for the network to converge in those areas due to insufficient overlap between the images. For examples, refer to Fig. A6.

F. Confidence of keypoints

We have discovered that uniform sampling of detected keypoints is more crucial than selectively mining keypoints with high confidence. In Fig. A7, we visualize a set of keypoints extracted from a single pair of images. The results highlight that we need to sample keypoints from the image more uniformly, rather than focusing solely on keypoints with high confidence, as confidence in keypoint detection is not uniform. To achieve this, we utilize multiple cameras to find matching keypoints with the source view and then sample all points above a certain confidence threshold.
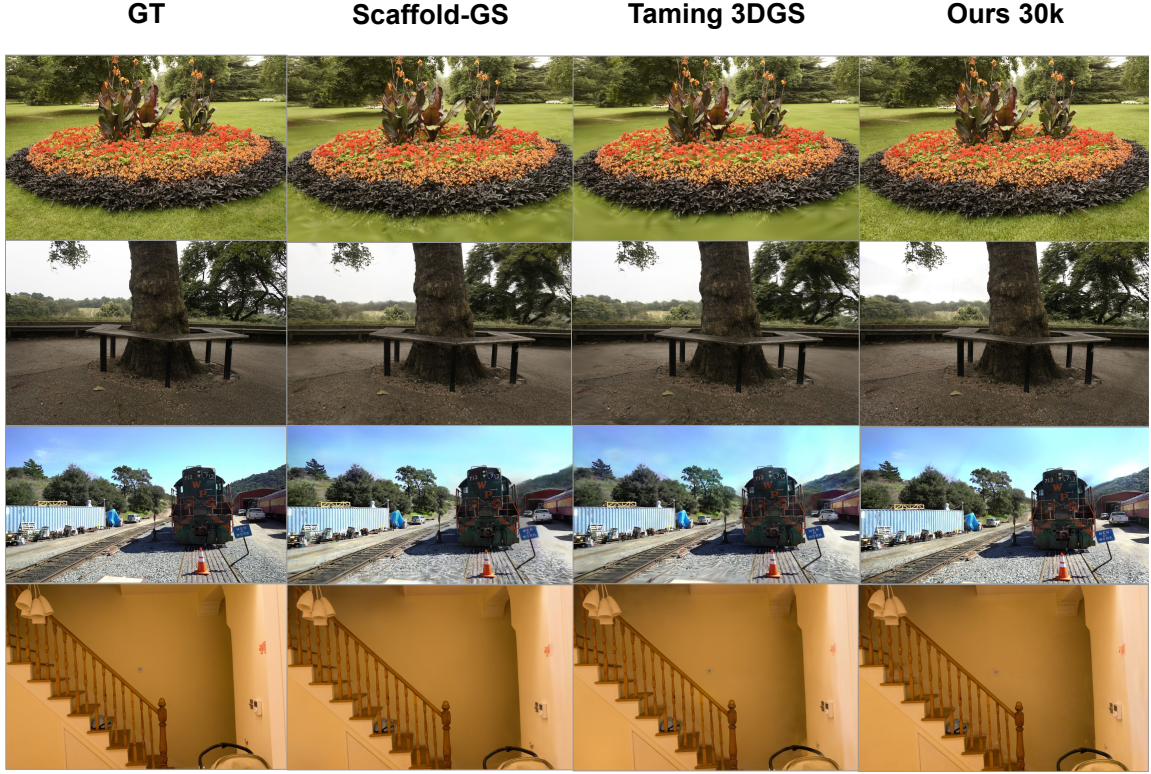
| GT | Scaffold-GS | Taming 3DGS | Ours 30k |

Figure A2. Additional qualitative results are presented for the scenes *treehill*, *flowers*, *train* and *playroom*. For clarity, areas of interest have been zoomed in Fig. 3. These results are best viewed digitally for optimal detail.

G. Impact of noise on initialization

Here, we provide additional visualization for ablation on the robustness of our method to noise. In Fig. A8, we visualized initialization for scene *garden*, which was noised with different scales for both coordinates (first row) and colors (second row).

H. Per-scene results

To provide a more detailed evaluation of our model, we include per-scene scores in Tab. A1, Tab. A2, Tab. A3, Tab. A4, Tab. A5, Tab. A6, Tab. A7, Tab. A8, Tab. A9, Tab. A10.

Table A1. Per-scene quantitative results(SSIM) from the Mip-NeRF360.

	bicycle	flowers	garden	stump	treehill	room	counter	kitchen	bonsai
Ours+ 3DGS 30K	0.794	0.642	0.875	0.782	0.655	0.932	0.998	0.937	0.947
Ours+ Taming 3DGS 30K	0.803	0.67	0.883	0.796	0.666	0.928	0.925	0.939	0.943

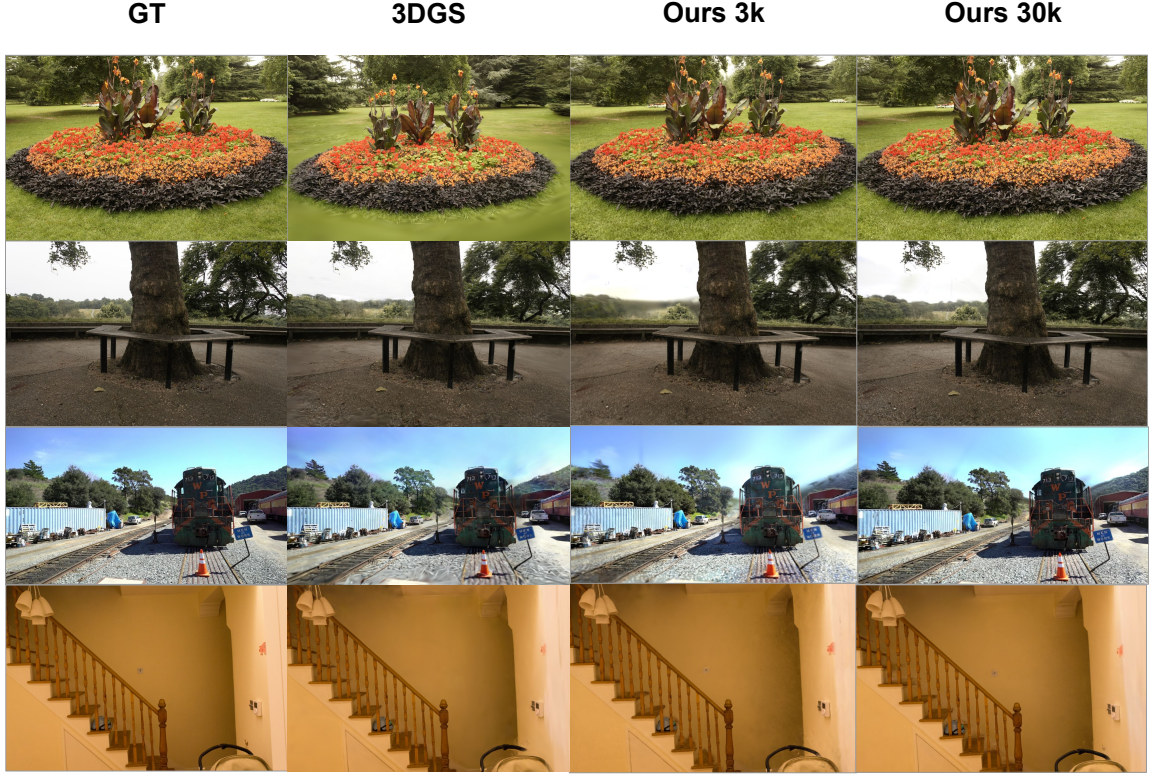|	GT	3DGS	Ours 3k	Ours 30k

Figure A3. Additional qualitative results are presented for the scenes *treehill*, *flowers*, *train* and *playroom*. For clarity, areas of interest have been zoomed in Fig. 3. These results are best viewed digitally for optimal detail.

Table A2. Per-scene quantitative results(PSNR) from the Mip-NeRF360.

	bicycle	flowers	garden	stump	treehill	room	counter	kitchen	bonsai
Ours+ 3DGS 30K	25.45	21.57	27.61	26.69	22.63	32	29.49	32.31	32.45
Ours+ Taming 3DGS 30K	25.83	21.87	28.09	27.05	23.15	32	29.65	32.52	32.34

Table A3. Per-scene quantitative results(LPIPS) from the Mip-NeRF360.

	bicycle	flowers	garden	stump	treehill	room	counter	kitchen	bonsai
Ours+ 3DGS 30K	0.157	0.256	0.09	0.189	0.244	0.182	0.169	0.112	0.174
Ours+ Taming 3DGS 30K	0.153	0.257	0.09	0.179	0.238	0.188	0.168	0.112	0.182

I. Notation

To simplify the understanding of the paper, we include a table of notation Tab. A11 in the supplementary material. This table provides a concise summary of the key symbols and terms used throughout the paper, along with their definitions.

Table A4. Per-scene quantitative results(millions of gaussians #G) from the Mip-NeRF360.

	bicycle	flowers	garden	stump	treehill	room	counter	kitchen	bonsai
Ours+ 3DGS 30K	2.3	2.6	3	2.2	2.5	0.9	1.2	1.3	1.3
Ours+ Taming 3DGS 30K	6	3.6	5.7	4.9	3.8	1.5	1.2	1.8	1.3

Table A5. Per-scene quantitative results(time in minutes) from the Mip-NeRF360.

	bicycle	flowers	garden	stump	treehill	room	counter	kitchen	bonsai
Ours+ 3DGS 30K	24	23	27	19	22	33	40	41	33
Ours+ Taming 3DGS 30K	18	13	21	13	13	13	13	12	12

Table A6. Per-scene quantitative results(SSIM) from the Tanks & Temples and Deep Blending subsets.

	Truck	Train	Dr Johnson	Playroom
Ours+ 3DGS 30K	0.897	0.851	0.908	0.910
Ours+ Taming 3DGS 30K	0.903	0.859	0.915	0.914

Table A7. Per-scene quantitative results(PSNR) from the Tanks & Temples and Deep Blending subsets.

	Truck	Train	Dr Johnson	Playroom
Ours+ 3DGS 30K	26.12	22.78	29.64	30.46
Ours+ Taming 3DGS 30K	26.48	23.38	29.92	30.63

Table A8. Per-scene quantitative results(LPIPS) from the Tanks & Temples and Deep Blending subsets.

	Truck	Train	Dr Johnson	Playroom
Ours + 3DGS 30K	0.09	0.157	0.224	0.213
Ours+ Taming 3DGS 30K	0.086	0.155	0.213	0.206

Table A9. Per-scene quantitative results(millions of gaussians #G) from the Tanks & Temples and Deep Blending subsets.

	Truck	Train	Dr Johnson	Playroom
Ours+ 3DGS 30K	1.6	1.2	1.6	1.5
Ours+ Taming 3DGS 30K	2.6	1.1	3.3	2.3

Table A10. Per-scene quantitative results(time in minutes) from the Tanks & Temples and Deep Blending subsets.

	Truck	Train	Dr Johnson	Playroom
Ours+ 3DGS 30K	21	23	28	31
Ours+ Taming 3DGS 30K	14	9	16	12

Table A11. Table of Notations

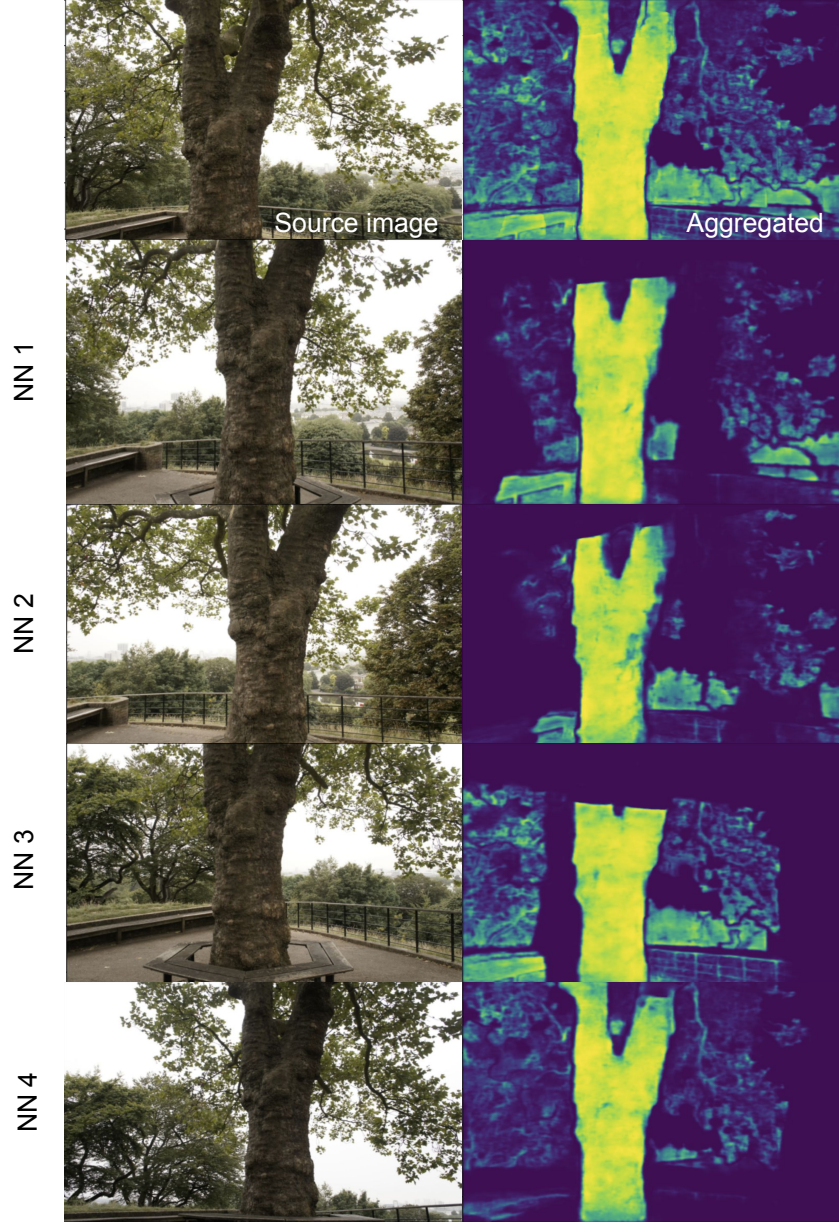| Notation	Description
$\mathbb{G}$	Set of 3D Gaussians representing the scene
$\boldsymbol{g}_i$	Parameters of the i-th Gaussian in the set $\mathbb{G}$
$\boldsymbol{g}_i^x \in \mathbb{R}^3$	Center of the i-th Gaussian in 3D space
$\boldsymbol{\Sigma}_i \in \mathbb{R}^7$	Covariance matrix encoding the shape of the i-th Gaussian
$\boldsymbol{g}_i^c \in \mathbb{R}^3$	RGB color of the i-th Gaussian
$\boldsymbol{g}_i^\alpha \in \mathbb{R}^1$	Opacity of the i-th Gaussian
p	Pixel
$C(p)$	Rendered color of pixel p
$\boldsymbol{\sigma}_i(p)$	Influence of the i-th Gaussian on pixel p
$\boldsymbol{p}' - \boldsymbol{g}_i^x$	Shortest distance from pixel projection line to the Gaussian center
$\boldsymbol{R}_i$	Rotation matrix of the i-th Gaussian
$\boldsymbol{S}_i$	Scaling matrix of the i-th Gaussian
I^i	Reference image selected from the training dataset
$\mathbb{I}$	Set of neighboring images for I^i based on spatial proximity
$\boldsymbol{P}$	Camera projection matrix
M	Pretrained 2D dense matching network
$\mathcal{W}^{j \to i}$	Dense warp field mapping pixels from I^j to I^i
$\mathbf{c}^{ij}$	Matchability score between I^i and I^j
(u_k^j, v_k^j)	Coordinates of a pixel in I^j
$\boldsymbol{g}_k^x \in \mathbb{R}^3$	3D position of the k-th prototype Gaussian
$\mathbf{p}_k^i = (x_k^i, y_k^i, w_k^i)$	homogenous coordinates of the projected pixel p_k^i
$(\hat{u}_k^i, \hat{v}_k^i)$	Projected coordinates of a 3D prototype in I^i
$\mathcal{L}$	Triangulation loss for optimizing 3D Gaussian positions
α	Scaling factor for Gaussian scale estimation
$\mathbf{C}^i$	Camera center of image I^i
$\boldsymbol{g}_k^s$	Scale of the k-th Gaussian based on distance from the camera

Figure A4. Visualization of the keypoint aggregation process from multiple nearest neighbors for the reference image I^i. The top-left panel shows the source image (reference image), while rows 2, 3, 4, and 5 depict ground truth images (on the left) of the nearest cameras, ordered by proximity in terms of camera extrinsics. The right side of each row shows the matching score $\mathbf{c}^{ij}$, representing areas of the source image matched with each neighboring image. The top-right panel displays the aggregated confidence map $\mathbf{c}^i$, combining matching scores from all neighbors. This step achieves fuller and more uniform coverage of the frame, as illustrated using the *treehill* image from the Mip-NeRF360 dataset.
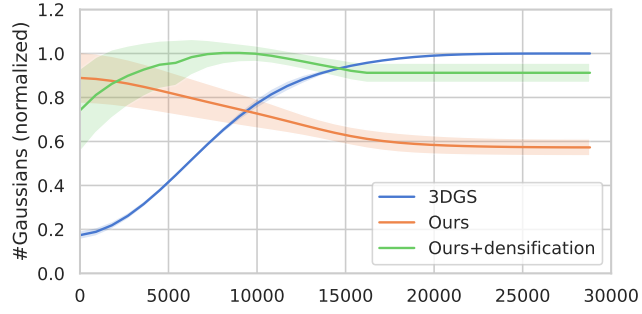
Figure A5. Comparison of the relative number of Gaussian splats over optimization steps. The count is normalized by the final number of splats in 3DGS [31]. Our method without densification requires only 60% of the Gaussians while converging faster and achieving similar performance to our densified version.
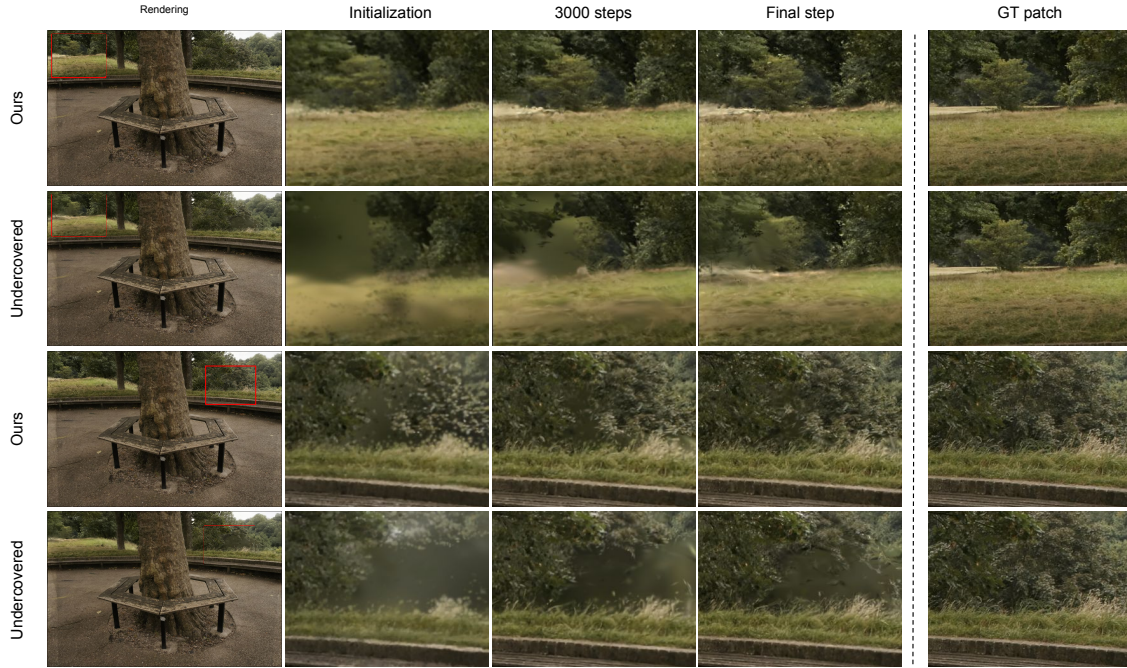

Figure A6. Our model benefits from dense coverage of the scene with paired views. If we sample enough views all the parts of the scene are covered and have enough initial prototypes to converge to a sharp image. Compare to the ground truth patch on the right.
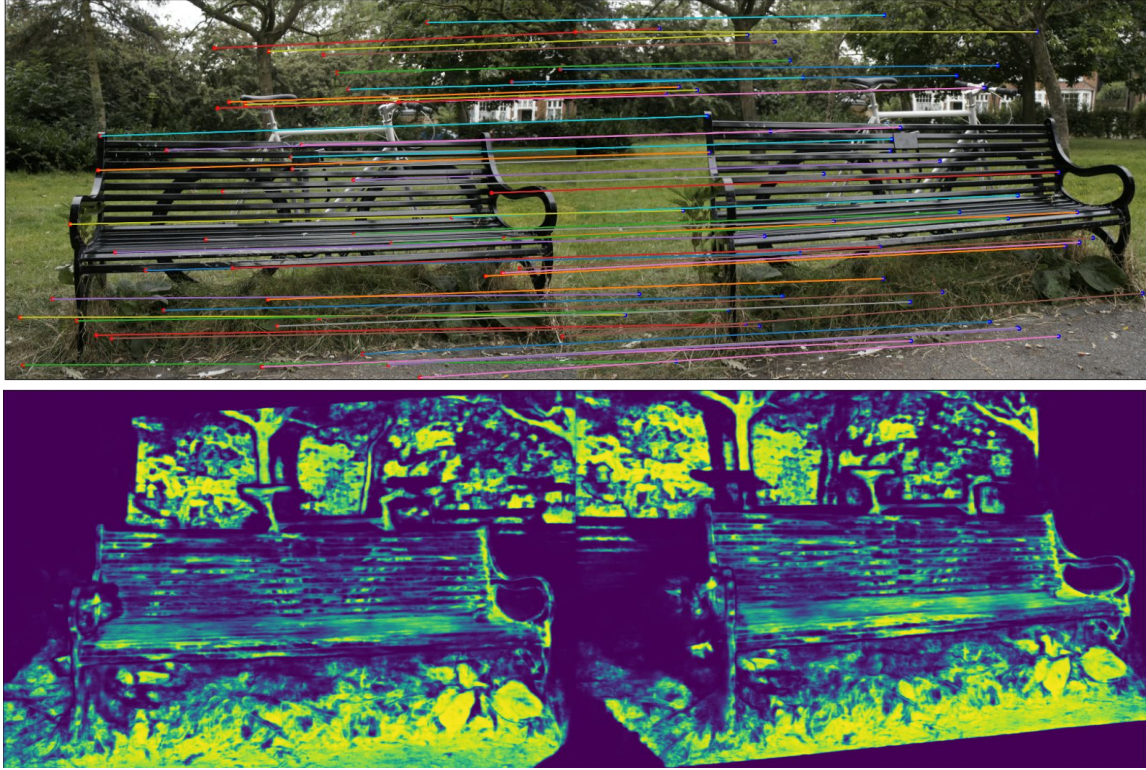
Figure A7. Dense keypoint matches for a *bicycle* scene image pair. The top row shows matched keypoints, and the bottom row visualizes the confidence of finding a match in the neighboring image. The matching model $\mathcal{M}$ is RoMa [13].
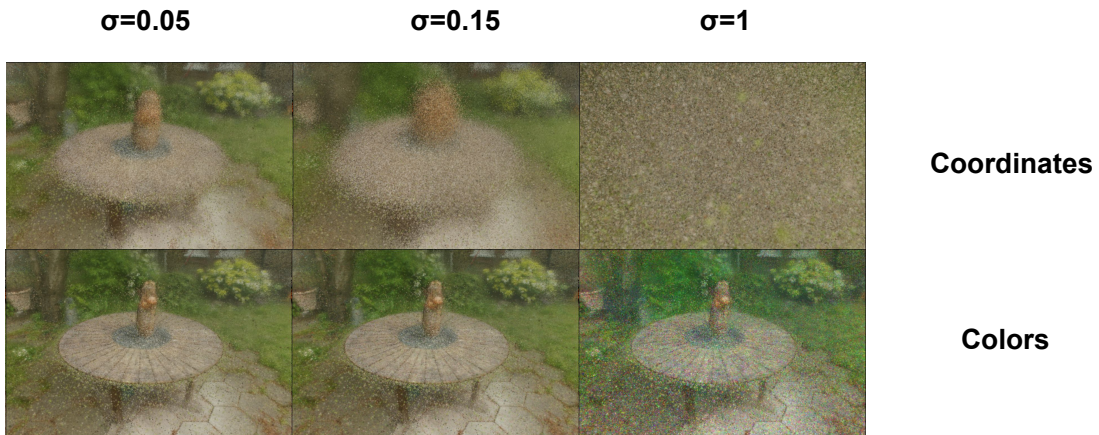

Figure A8. The impact of noise on initialization quality. The first row shows the effect of adding noise to the coordinates, while the bottom row demonstrates the effect of adding noise to the color values.