Diff3R: Feed-forward 3D Gaussian Splatting with Uncertainty-aware Differentiable Optimization

Yueh-Cheng Liu[1] **Jozef Hladký**[2] **Matthias Nießner**[1] **Angela Dai**[1]

[1]Technical University of Munich
[2]Computing Systems Lab, Huawei Technologies, Switzerland
https://liu115.github.io/diff3r

Abstract

Recent advances in 3D Gaussian Splatting (3DGS) present two main directions: feed-forward models offer fast inference in sparse-view settings, while per-scene optimization yields high-quality renderings but is computationally expensive. To combine the benefits of both, we introduce Diff3R, a novel framework that explicitly bridges feed-forward prediction and test-time optimization. By incorporating a differentiable 3DGS optimization layer directly into the training loop, our network learns to predict an optimal initialization for test-time optimization rather than a conventional zero-shot result. To overcome the computational cost of backpropagating through the optimization steps, we propose computing analytical gradients via the Implicit Function Theorem and a scalable, matrix-free PCG solver tailored for 3DGS optimization. Additionally, we incorporate a data-driven uncertainty model into the optimization process by adaptively controlling how much the parameters are allowed to change during optimization. This approach effectively mitigates overfitting in under-constrained regions and increases robustness against input outliers. Since our proposed optimization layer is model-agnostic, we show that it can be seamlessly integrated into existing feed-forward 3DGS architectures for both pose-given and pose-free methods, providing improvements for test-time optimization.

1 Introduction

Novel view synthesis (NVS) from a set of unposed or posed images is a fundamental and longstanding challenge in 3D computer vision. Recently, 3D Gaussian Splatting (3DGS) [1] has revolutionized the field by achieving real-time, high-fidelity rendering through an unstructured, explicit point-based representation. However, the standard 3DGS pipeline requires computationally expensive, per-scene optimization, limiting its applicability. To circumvent this, a recent wave of feed-forward 3DGS frameworks has emerged. By leveraging large-scale datasets, these models learn to directly predict 3D Gaussian parameters from multiple input views, operating in both pose-given [2–7] and pose-free [8–12] settings, even with only a few images.

While feed-forward models demonstrate impressive generalization and fast inference, they fundamentally rely on learned priors. Consequently, they often struggle to reconstruct scene-specific, high-frequency details, leading to blurry or over-smoothed renderings. A natural strategy to bridge this domain gap is Test-Time Optimization (TTO), where the predicted Gaussians are fine-tuned using the available context views. However, in the highly under-constrained sparse-view setting, standard TTO frequently falls into a critical failure mode: catastrophic overfitting. Because the optimization lacks sufficient multi-view supervision, the model can easily manipulate the Gaussians to perfectly reconstruct the context views while severely corrupting the underlying geometry (*e.g.*, generating floaters or cheated view-dependent effects), drastically degrading novel view synthesis performance.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

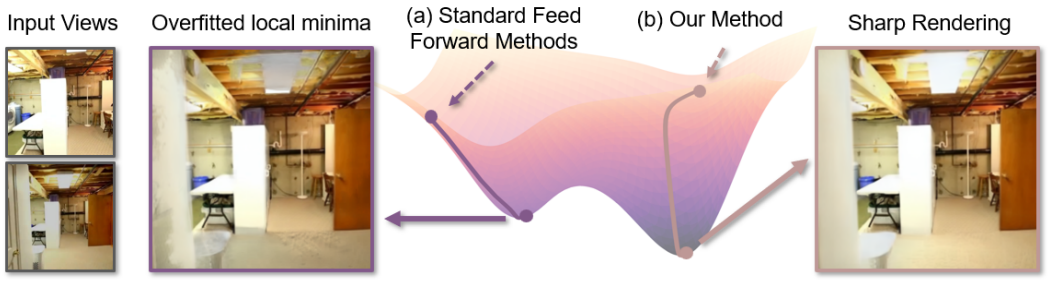

Figure 1: **Optimization-aware 3D Gaussian splatting.** (a) Standard feed-forward 3DGS models directly predict Gaussian parameters, frequently resulting in blurry novel views. Applying test-time optimization (TTO) to these predictions in sparse-view settings is highly unconstrained, which makes it easy to overfit to the input views, trapping the solution in poor local minima with severe visual artifacts. (b) By training through the optimization process via implicit gradients, our method learns an optimization-aware initialization, tailored for 3DGS post-optimization.

To address this issue, we propose Diff3R, a novel method to train feed-forward 3DGS models. Instead of training a network to predict the final set of 3D Gaussians directly, we propose training it explicitly to predict the optimal initialization for test-time optimization (Figure 1). Drawing inspiration from meta-learning and hyperparameter optimization [13, 14], we formulate this as a bilevel optimization problem. During training, our model predicts an initial set of Gaussians, which undergo a brief inner-loop optimization to fit the context views. The network weights are then updated in the outer loop based on the novel-view rendering quality of the optimized Gaussians.

Key to our success is efficiently backpropagating through this optimization during training. To solve this, we propose a tailored, differentiable optimization strategy. By leveraging the Implicit Function Theorem and Gauss-Newton approximations, we analytically compute the implicit gradients [15, 16] of the outer novel-view loss with respect to the initial Gaussian parameters. This formulation completely avoids unrolling the optimization trajectory (*e.g.*, differentiating through every optimization step), enabling memory-efficient, end-to-end training of the entire pipeline.

Furthermore, to explicitly tackle the overfitting problem during TTO, we introduce an uncertainty-aware optimization mechanism. We observe that feed-forward predictions are not uniformly reliable for downstream optimization. Therefore, Diff3R is designed to predict learnable per-Gaussian, per-attribute regularization weights (*i.e.*, uncertainty), alongside the initial 3D Gaussians. These weights dynamically regulate the inner optimization: the model learns to penalize deviations from the feed-forward prior strictly for highly confident predictions, while granting heavily uncertain Gaussians the freedom to adapt. Remarkably, because our implicit gradients flow back through these meta-parameters, the network learns to quantify its own uncertainty purely from outer-loop novel-view supervision, requiring no ground-truth uncertainty labels.

Since our proposed method is model-agnostic, we build our model based on DepthSplat [6] and Depth Anything v3 [10] to support both pose-given and pose-free settings. Our method achieves improvements over feed-forward baselines after test-time optimization, gaining over 0.25 dB PSNR on RealEstate10K and 0.35 dB PSNR on ScanNet++. Ablation studies confirm that both the differentiable optimization framework and the learned uncertainty regularization are essential for these gains.

In summary, our main contributions are as follows:

- We propose a novel bilevel optimization framework for feed-forward 3DGS. By analytically computing implicit gradients via a matrix-free approach, we enable efficient, end-to-end backpropagation through the test-time optimization process.
- We introduce an adaptive uncertainty modeling strategy. By predicting per-parameter regularization weights, our model learns to penalize overfitting to sparse context views during optimization, balancing the feed-forward prior with new information from test-time observations.

2 Related Work

2.1 Feed-forward 3DGS

Feed-forward methods predict 3DGS [1] parameters directly from a few input views, bypassing per-scene optimization. We group them by their reliance on camera information.

Pose-given methods [2–7] leverage known camera intrinsics and extrinsics to construct multi-view geometry priors. By utilizing epipolar attention mechanisms or 3D cost volumes, these networks aggregate features across views to predict the geometry (means, scales, rotations) and appearance (opacity, spherical harmonics) of the Gaussians for each pixel (*i.e.*, per-pixel Gaussians).

Pose-free methods rely on recent advances in two-view [17, 18] or multi-view [10, 19, 20] feed-forward reconstruction models, which directly predict 3D geometry via global attention transformers. Building on these foundations, several approaches [8–11, 21] are now capable of estimating both the camera poses and the Gaussian attributes directly from unposed images.

These methods all minimize zero-shot reconstruction error, leaving their predictions as poor starting points for refinement; we instead train *through* the optimization process to learn an optimization-aware initialization for 3DGS.

2.2 Differentiable Optimization

Differentiable optimization enables end-to-end learning by backpropagating through geometric solvers. Prior works have successfully differentiated through Gauss-Newton steps for non-rigid tracking [22], Levenberg-Marquardt alignment [23], and bundle adjustment [24, 25] using implicit differentiation. More recently, VGGSfM [26] extended these concepts to full Structure-from-Motion (SfM) pipelines using differentiable Cholesky solvers [27].

While these methods address sparse point clouds and camera poses, differentiating through 3DGS optimization remains under-explored due to scale: a typical scene contains hundreds of thousands to millions of Gaussians and equally many per-pixel residuals, making naïve auto-differentiation prohibitively memory-intensive. We address this with a memory-efficient implicit-gradient formulation tailored to this scale.

2.3 Meta-Learning for 3D Reconstruction

Meta-learning, often framed as "learning to learn," focuses on training models that can rapidly adapt to new tasks using minimal data. A foundational approach in this domain is MAML [13], which employs bilevel optimization to discover an optimal network initialization for fast downstream adaptation. To reduce the memory burden of backpropagating through unrolled gradient steps, subsequent works introduced efficient alternatives by first-order approximation [28] or Implicit Function Theorem [15]. Furthermore, methods like Meta-SGD [29] expanded this paradigm by learning both the initialization and per-parameter adaptation rates.

In 3D vision, these strategies have been applied to coordinate-based representations – MetaSDF [30] and meta-learned NeRFs [31] learn initial network weights for fast per-scene convergence. Recently, G3R [32] and QuickSplat [33] train iterative refinement networks for 3DGS but backpropagate through unrolled steps, incurring a large memory footprint. In contrast, our implicit-gradient formulation avoids this and additionally enables end-to-end learning of per-Gaussian uncertainty regularizers.

2.4 Test-time optimization

Test-time optimization (TTO), or test-time training (TTT), specializes a model to each test instance at inference [34, 35]; classical 3D instances include post-optimization of NeRF [36] and 3DGS [1] after feed-forward prediction, and global alignment in DUSt3R [17] and MASt3R-SfM [37]. Recent streaming TTT methods [38–41] target a different long-context regime by updating recurrent state, fast weights, or adapters online, as observations stream in, which is a different problem from our sparse-view, fixed-context setting. Closer in setting, Test3R [42] fine-tunes a DUSt3R-style *pointmap* backbone on the test inputs but does not train the network to be optimization-aware; we instead optimize the explicit Gaussian splats output and train the predictor *through* the inner optimization via implicit differentiation.

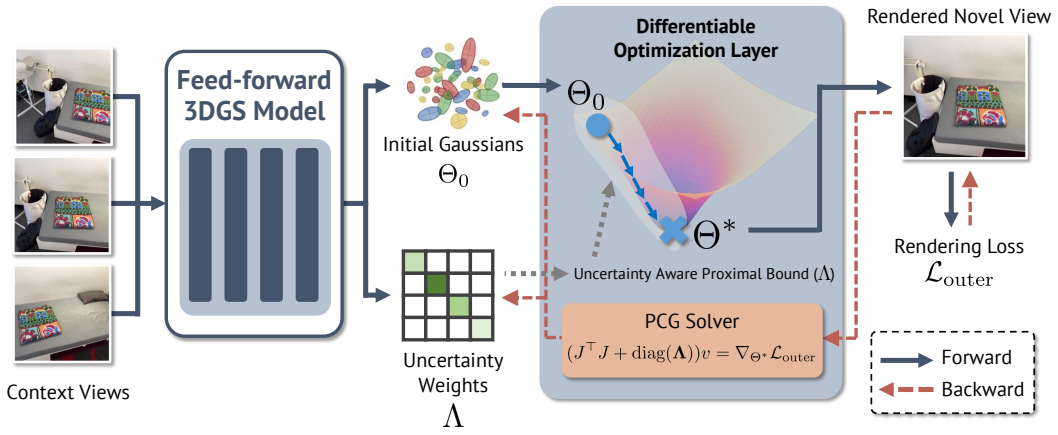

Figure 2: **Overview of our uncertainty-aware differentiable 3DGS framework.** Given a sparse set of context views (with optional camera parameters), our feed-forward network predicts an initial set of 3D Gaussian parameters (Θ_0). Our proposed differentiable optimization layer refines these parameters via gradient descent to yield the optimized Gaussians (Θ^*). To train the network end-to-end, we introduce an efficient analytical solution for the backward pass using implicit gradients and a matrix-free PCG solver. Additionally, to make the optimization more robust in sparse-view settings, we predict learnable uncertainty weights ($\mathbf{\Lambda}$). These weights act as an adaptive proximal bound on the optimization trajectory, preventing the model from overfitting to the context views.

3 Method

3.1 Preliminaries: 3D Gaussian Splatting

We adopt the standard 3D Gaussian Splatting (3DGS) representation [1], where a scene is encoded as an unstructured set of Gaussian splats with parameters $\Theta = \{\mu_i, S_i, R_i, \alpha_i, c_i\}_{i=1}^N$ (mean, scale, rotation, opacity, and SH color). We write the differentiable rendering process under camera pose P as $I = f(\Theta, P)$, or simply $f(\Theta)$ when the pose is implicit. Feed-forward 3DGS replaces per-scene optimization by predicting Θ directly from a set of context views $\{I_{ctx}, P_{ctx}\}$ via a network g_ϕ (parameterized by ϕ); we treat its output as the initialization $\Theta_0 = g_\phi(\{I_{ctx}, P_{ctx}\})$.

3.2 Differentiable 3DGS Optimization

While feed-forward models produce fast initializations Θ_0, they often lack high-frequency detail. Our goal is to train a feed-forward 3DGS model that outputs 3D Gaussians designed for downstream test-time optimization. Therefore, we propose a bilevel optimization framework (Figure 2). The *inner loop* performs a brief test-time optimization (TTO) to fit the context views, while the *outer loop* updates the network weights ϕ by rendering novel views.

The inner optimization objective minimizes the photometric loss against the context views, regularized by a proximal term. Letting $r(\Theta) = f(\Theta) - I_{ctx}$ denote the rendering residual (*i.e.*, the pixel difference between the rendering and the GT for each pixel), the inner loss is formulated as:

$$\mathcal{L}_{\text{inner}}(\Theta) = \mathcal{L}_{\text{photo}}(\Theta) + \mathcal{L}_{\text{proximal}}(\Theta) = \frac{1}{2}\|r(\Theta)\|_2^2 + \frac{\lambda}{2}\|\Theta - \Theta_0\|_2^2, \tag{1}$$

where λ is a global regularization scalar. We denote the converged parameters of this inner optimization as $\Theta^* = \arg\min_\Theta \mathcal{L}_{\text{inner}}(\Theta)$.

Following iMAML [15], we include the proximal term $\mathcal{L}_{\text{proximal}}(\Theta)$ for two critical reasons. First, it bounds the optimization, ensuring the Gaussians do not deviate excessively from the prior learned by the feed-forward network, which is particularly helpful for training differentiable optimization. Second, it ensures the optimality condition such that it has gradients w.r.t. the initial Θ_0 (see next paragraph).

The outer optimization trains the feed-forward network by penalizing novel view rendering errors using the optimized parameters Θ^*:

$$\mathcal{L}_{\text{outer}}(\Theta^*) = \mathcal{L}_{\text{mse}}\left(f(\Theta^*), I_{nv}\right) + \lambda_{\text{lpips}}\mathcal{L}_{\text{LPIPS}}\left(f(\Theta^*), I_{nv}\right), \tag{2}$$

where I_{nv} is the ground-truth image of a novel view not among the context views.

For each training iteration, the network outputs initial Gaussian parameters Θ_0. These are optimized for a fixed number of steps using Equation (1) to yield Θ^*. We then compute the outer loss (Equation (2)) and backpropagate the error to update the feed-forward model parameters ϕ.

Implicit Gradient. To update the network weights ϕ, we apply the chain rule $\frac{\partial \mathcal{L}_{\text{outer}}}{\partial \phi} = \frac{\partial \mathcal{L}_{\text{outer}}}{\partial \Theta^*}\frac{\partial \Theta^*}{\partial \Theta_0}\frac{\partial \Theta_0}{\partial \phi}$. The first and third terms are standard rendering and network backpropagation; the central challenge is the middle term, $\partial\Theta^*/\partial\Theta_0$, which requires differentiating through the inner optimization. Rather than unrolling steps (memory-intensive), we compute the gradient at the optimum via the Implicit Function Theorem (IFT) in the iMAML style [15, 16, 43]. Differentiating the optimality condition $\nabla_\Theta \mathcal{L}_{\text{inner}}(\Theta^*) = \mathbf{0}$ w.r.t. Θ_0 and applying the Gauss-Newton approximation $H_{\text{photo}} \approx J^\top J$ yields the closed-form Jacobian

$$\frac{\partial \Theta^*}{\partial \Theta_0} = \lambda(J^\top J + \lambda I)^{-1}. \tag{3}$$

Here $J = \frac{\partial r(\Theta^*)}{\partial \Theta^*}$ is the Jacobian of the per-pixel rendering residuals with respect to all Gaussian parameters; using $J^\top J$ in place of the exact Hessian avoids second-order rendering derivatives, which would be prohibitive at the scale of 3DGS. The full step-by-step derivation is provided in Section B of the supplementary material.

Explicitly instantiating and inverting the massive matrix $(J^\top J + \lambda I)$ is intractable. Instead, we formulate this as a linear system by introducing an auxiliary vector v:

$$(J^\top J + \lambda I)v = \nabla_{\Theta^*}\mathcal{L}_{\text{outer}}. \tag{4}$$

Because the matrix $(J^\top J + \lambda I)$ is symmetric and positive-definite, we can solve for v efficiently using the matrix-free Preconditioned Conjugate Gradient (PCG) algorithm. PCG only requires matrix-vector products. Once PCG converges to a solution for v, the final gradient passed back to the network is simply $\nabla_{\Theta_0}\mathcal{L}_{\text{outer}} = \lambda v$.

The left-hand side of Equation (4) matches the LM [44] normal equations, so we adopt the efficient CUDA-based LM solver for 3DGS [45] to compute matrix-vector products. We discuss the connection to LM and the behavior of the gradient at extremes of λ in Section B.5 of the supplementary material.

3.3 Uncertainty-aware 3DGS Differentiable Optimization

A critical limitation of the formulation in Equation (1) is the use of a rigid, global regularization λ. It applies the same penalty to every Gaussian regardless of confidence: well-predicted regions are over-constrained while uncertain ones are not protected enough from overfitting to the context views. Moreover, choosing a single scalar λ forces a trade-off between two failure modes during bilevel training – too small and the inner loop overfits while the gradient to the network vanishes, too large and the optimization is over-constrained (see Section B.5 of the supplementary material for a detailed analysis). We therefore would like a regularizer that accounts for this varying "uncertainty" across feed-forward predictions on a per-parameter basis.

We resolve this by reformulating the inner loss with a per-Gaussian, per-attribute parameter regularizer $\mathbf{\Lambda} \in \mathbb{R}^{N_\Theta}$, whose entries are data-dependent meta-parameters predicted alongside Θ_0 by the feed-forward network g_ϕ:

$$\mathcal{L}_{\text{inner}}(\Theta) = \frac{1}{2}\|r(\Theta)\|_2^2 + \frac{1}{2}(\Theta - \Theta_0)^\top \text{diag}(\mathbf{\Lambda})(\Theta - \Theta_0). \tag{5}$$

Here, $\mathbf{\Lambda}$ replaces the global λ. If the network is highly uncertain about a specific Gaussian's initial position or color, it predicts a small penalty value in $\mathbf{\Lambda}$, allowing that parameter to adapt freely during TTO. Conversely, well-predicted Gaussians are anchored strongly to their initialization Θ_0. Revisiting the IFT with the diagonal regularizer (with $\text{diag}(\mathbf{\Lambda})$ replacing λI in our linear system) and reusing the same matrix-free PCG solution v, we obtain the modified gradients:

$$\begin{aligned} \nabla_{\Theta_0}\mathcal{L}_{\text{outer}} &= \mathbf{\Lambda} \odot v, \\ \nabla_{\mathbf{\Lambda}}\mathcal{L}_{\text{outer}} &= -v \odot (\Theta^* - \Theta_0), \end{aligned} \tag{6}$$

Table 1: **Novel view synthesis on RE10K with known poses.** We report PSNR, SSIM, and LPIPS before and after test-time optimization (TTO) with two input views. While DepthSplat and ours achieve similar zero-shot performance, our method exhibits substantially larger gains after optimization (+0.27 dB vs. +0.03 dB) and attains the highest final scores, demonstrating that our optimization-aware initialization effectively balances prior preservation with adaptation flexibility.

	PSNR		SSIM		LPIPS	
	Before	After	Before	After	Before	After
PixelSplat [2]	25.856	26.144 (+0.288)	0.848	0.854	0.150	0.145
MVSplat [3]	26.654	26.649 (−0.005)	0.864	0.866	0.132	0.131
DepthSplat [6]	27.665	27.694 (+0.029)	0.886	0.888	0.117	0.116
Ours	**27.675**	**27.949** (+0.274)	**0.887**	**0.891**	**0.116**	**0.112**

where $\odot$ denotes the element-wise product and v solves $(J^\top J + \mathrm{diag}(\mathbf{\Lambda}))v = \nabla_{\Theta^*}\mathcal{L}_{\text{outer}}$. The full derivation and the relation to Meta-SGD [29] are provided in Section B of the supplementary material.

These analytical gradients offer a straightforward intuition. Consider the gradient for the uncertainty weights, $\nabla_{\mathbf{\Lambda}}$. If the desired update from the outer loss (v) points in the same direction as the actual movement during TTO ($\Theta^* - \Theta_0$), their product is positive. The resulting negative gradient reduces $\mathbf{\Lambda}$, relaxing the penalty to give that Gaussian more freedom to adapt in future iterations. Through this simple alignment check, the network learns an optimal, per-parameter uncertainty representation directly from novel-view supervision, without requiring explicit ground-truth labels.

3.4 Implementation

Model. The proposed methods are agnostic to the design of the feed-forward 3DGS models. In this work, we explore two variations: we build our method on DepthSplat [6] for the feed-forward 3DGS with camera information (*e.g.*, intrinsics and poses), and Depth Anything v3 [10] for the setting with multiple images only (*i.e.*, pose-free).

Both methods rely on the DPT head [46] to predict per-pixel 3D Gaussians and their attributes. Therefore, we add a small MLP to predict the Gaussian parameter regularizer Λ from the intermediate features in DPT. For both methods, we initialize from the official pretrained weights.

Training and solver details. For the inner TTO objective, which runs for 50 (pose-given) or 100 (pose-free) steps during training, we use the standard 3DGS L1+SSIM photometric loss instead of pure L2 (with $r(\Theta)$ and J adjusted accordingly via the chain rule). To stabilize early training, we use a two-stage schedule (pretrain g_ϕ on the outer loss with $\Theta^* = \Theta_0$, then activate the differentiable inner TTO) and add a small proxy rendering loss on Θ_0 with weight $\lambda_{\text{proxy}} = 0.1$. The matrix-free PCG solver builds on the CUDA kernels of 3DGS-LM [45]; we normalize J by the number of residuals and replace the proximal weight with an LM-style diagonal scaling $\lambda \cdot \mathrm{diag}(\mathbf{\Lambda} \odot M)$, where M is a moving average of $\mathrm{diag}(J^\top J)$, to balance the scale of heterogeneous Gaussian attributes. More details are provided in Section C of the supplementary material.

4 Experiments

Dataset. For the pose-given experiments, we train and evaluate the methods on the RealEstate10K (RE10K) dataset [47]. We follow previous methods and use the PixelSplat [2] processed train-test split. For the pose-free methods, we fine-tune Depth Anything v3 using the combination of multiple datasets: RE10K [47], ScanNet [48], ScanNet++ [49], DL3DV [50], Hypersim [51], and ArkitScenes [52]. We evaluate the pose-free methods on the RE10K dataset using the same protocol as pose-given experiments, and ScanNet++ on the official validation split.

Baselines. For the pose-given setting, we compare our method with PixelSplat [2], MVSplat [3], and DepthSplat [6] with two input views. For the pose-free experiments, we compare our method with AnySplat [11] and Depth Anything v3 [10] with four input views. We use 256×256 resolution for the baselines and our method for all the experiments.



Figure 3: **Qualitative results on RE10K dataset with pose-given setting.** We compare ours with feed-forward 3DGS methods such as PixelSplat [2], MVSplat [3], and DepthSplat [6] with two input views. As shown in the zoom-in regions, ours generates sharper results, and is more robust to exposure noise in the input views during optimization thanks to the adaptive regularization.

Metrics. We evaluate the standard novel view synthesis metrics: PSNR, SSIM, and LPIPS after running the test-time optimization on the input views. For the pose-given evaluation, we run 50 steps of test-time optimization as we notice that most of the methods have converged after that. For the pose-free evaluation, we first align the target poses to the predicted camera poses by estimating the global scale using the Umeyama method [53], and then optimize for the target pose while keeping the 3DGS parameters fixed, following previous works [8, 9, 11] to make sure that the target view poses are aligned with the reconstructed 3D scenes. Then, we run 300 steps of test-time optimization.

4.1 Novel view synthesis

Pose given. Quantitative and qualitative results are presented in Table 1 and Figure 3, respectively. Our method consistently outperforms baseline approaches following test-time optimization (TTO). This performance gain is fundamentally driven by our bilevel formulation: rather than strictly maximizing zero-shot rendering fidelity, the network learns to predict optimization-aware initializations that are more favorable for TTO. Furthermore, our adaptive proximal regularizer ensures the optimization process is highly robust against overfitting in under-constrained, sparse-view settings. For instance, as shown in the bottom row of Figure 3, baseline methods degrade severely when exposed to inconsistent color shifts in the context views. Conversely, our approach mitigates these dramatic illumination changes during TTO. By training end-to-end to maximize post-optimization novel view quality, our model learns data-dependent uncertainty bounds that effectively discount such unreliable observations.

Pose free. For pose-free reconstruction, we build upon Depth Anything v3 (DA3) as our base feed-forward model. As shown in Table 2 (RE10K) and Table 3 (ScanNet++), our uncertainty-aware optimization outperforms standard TTO applied to vanilla DA3, achieving larger performance gains after optimization (+0.76 dB vs. +0.60 dB on RE10K) while producing higher-quality initializations. As shown in Figure 4, Diff3R can generate sharper details with fewer artifacts caused by overfitting to the input views.

Table 2: **Pose-free feed-forward 3DGS on RE10K.** Comparison of feed-forward methods using four input views without camera information. We report PSNR, SSIM, and LPIPS before and after test-time optimization (TTO). Notably, AnySplat suffers from catastrophic overfitting (PSNR degrades by 0.2 dB) due to poor initialization, while our method achieves the largest performance gains (+0.76 dB vs. +0.60 dB for DA3) and attains the best final scores across all metrics.

	PSNR		SSIM		LPIPS	
	Before	After	Before	After	Before	After
AnySplat [11]	17.762	17.546 (−0.216)	0.603	0.585	0.325	0.328
DA3 [10]	21.981	22.582 (+0.601)	0.741	0.764	0.223	0.205
Ours	**22.043**	**22.806** (+0.763)	**0.742**	**0.767**	**0.221**	**0.202**

Table 3: **Pose-free evaluation on ScanNet++ (DSLR & iPhone) with four input views.** Results on the official validation split using four unposed input views. We compare against AnySplat and Depth Anything v3 (DA3), reporting metrics before and after TTO. Our method consistently outperforms baselines after optimization across both capture modalities, achieving particularly strong gains on the challenging DSLR split (+0.35 dB PSNR against DA3).

	ScanNet++ (DSLR)						ScanNet++ (iPhone)					
	PSNR		SSIM		LPIPS		PSNR		SSIM		LPIPS	
	Before	After	Before	After	Before	After	Before	After	Before	After	Before	After
AnySplat [11]	18.84	18.66	0.675	0.654	0.307	0.312	18.21	18.08	0.671	0.655	0.304	0.299
DA3 [10]	23.54	24.29	0.820	0.844	0.193	0.174	**23.23**	24.43	**0.840**	0.878	0.171	0.152
Ours	**23.78**	**24.64**	**0.824**	**0.850**	**0.182**	**0.162**	23.15	**24.57**	**0.840**	**0.881**	**0.167**	**0.145**



Figure 4: **Qualitative results on ScanNet++.** We compare our pose-free methods against Depth Anything v3 (DA3) [10] after optimization on ScanNet++ with four input views. Our method is able to reduce the artifacts during optimization and provides sharper renderings.

4.2 Ablations

Is the proximal term effective? We drop the proximal term during test-time optimization after training. The result is shown in Table 4 (b). Compared to vanilla DepthSplat it still improves due to better initialization. However, it is worse than the full method (d) since the optimization is less constrained and still suffers from overfitting to the input views.

Table 4: **Ablation study on Diff3R.** We compare our full method and other variations on RE10K, pose-given, two-view settings. Without proximal term, the method is still benefiting from learning a better initialization but becomes worse due to unconstrained optimization. On the other hand, without the learnable parameter regularization, the global bound makes the optimization too conservative.

	PSNR			SSIM		LPIPS	
	Before	After		Before	After	Before	After
(a) DepthSplat [6]	27.665	27.694	(+0.029)	0.886	0.888	0.117	0.116
(b) Ours w/o $\mathcal{L}_{\text{proximal}}$	**27.675**	27.828	(+0.153)	0.887	0.890	**0.116**	0.115
(c) Ours w/o Λ	27.630	27.746	(+0.116)	**0.888**	0.889	**0.116**	0.115
(d) Ours	**27.675**	**27.949**	(+0.274)	0.887	**0.891**	**0.116**	**0.112**

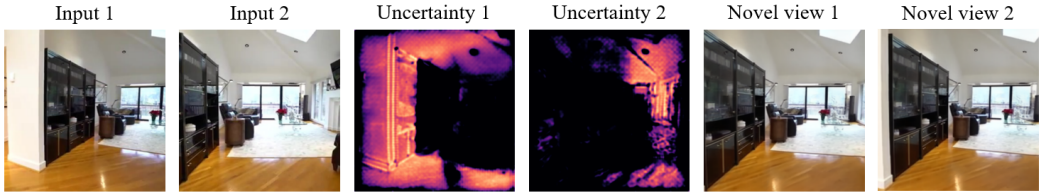

Figure 5: **Visualization of learned uncertainty weights.** We visualize the regularization weights applied to the mean of the per-pixel Gaussians (Λ_{mean}). Notably, the network intrinsically learns to predict higher anchoring weights for regions lacking multi-view constraints (*e.g.*, the left third of Input 1, which is not visible in Input 2) without explicit supervision. This helps to prevent overfitting during the test-time optimization process. More examples are provided in the supplementary material (see Figure A1).

Are the learnable regularization weights effective? The parameters are bounded by a single λ value uniformly without the learnable regularization weights. Therefore, the optimization becomes more conservative. As shown in Table 4 (c), the initializations are forced to behave like a standard feed-forward model, and the improvements from the optimization are limited. We also provide the visualization of the learned regularization weights in Figure 5. The network learns to predict higher anchoring weights for regions lacking multi-view constraints without explicit supervision.

Number of inner steps. Going from $K = 10$ to $K = 50$ improves PSNR after TTO by 0.13 dB, while going from 50 to 100 adds only 0.03 dB at twice the training cost (Table 5). Therefore, we choose $K = 50$ for our pose-given setting as a good trade-off between performance and training cost.

Proxy loss. We test the effect of the proxy loss with different values of λ_{proxy} (Table 6). Without the proxy loss, training degrades and does not recover, since the only gradient path to Θ_0 is through the learnable regularization weights. On the other hand, with $\lambda_{\text{proxy}} = 1$, the proxy dominates the outer signal, so the network optimizes the one-shot output rather than the initialization, which costs 0.14 dB.

Table 5: **Number of inner steps K during training.** RE10K, pose-given, two views.

	PSNR		SSIM	LPIPS
K	Before	After	After	After
10	27.649	27.818	0.890	0.114
50 (ours)	**27.675**	27.949	0.891	**0.112**
100	27.665	**27.981**	**0.892**	**0.112**

Table 6: **Proxy loss weight λ_{proxy}.** RE10K, pose-given, two views.

	PSNR
λ_{proxy}	After
0	degrades
0.1 (ours)	**27.949**
1.0	27.805

4.3 Efficiency analysis

We compare our implicit gradients against unrolled backpropagation through the inner loop in Table 7. Our backward is $O(1)$ in the inner step count K, while unrolled backpropagation is $O(K)$ in both

Table 7: **Efficiency of implicit gradients vs. unrolled backpropagation.** Time and memory of differentiating through K inner optimization steps. Our implicit gradient stays nearly constant in K, while unrolled backpropagation grows linearly in both.

		Backward (ms)		Peak mem. (GB)		Retained (GB)	
K	Forward (ms)	Unrolled	Ours	Unrolled	Ours	Unrolled	Ours
5	60.5	24,897	171.6	17.66	1.97	0.89	0.03
10	117.6	49,104	173.3	18.11	2.22	1.36	0.03
20	230.5	96,263	175.4	19.04	2.21	2.30	0.03
50	587.5	243,885	179.8	21.81	2.21	5.13	0.03

time and memory. At $K = 50$, our backward is about $1{,}350\times$ faster and uses about $10\times$ less peak memory, which makes training in our setting practical.

Setting. We use RE10K with two context views at 256×256, 131,072 Gaussians (one per input pixel), and report the mean over 5 scenes. The unrolled variant keeps all K inner Adam steps in the autograd graph, so its reverse pass costs one Hessian-vector product through the renderer per step. No released 3DGS rasterizer provides the second-order rendering derivatives this needs, so we built a reference implementation that reproduces the unrolled variant's computation and memory pattern. We use it to measure how cost scales with K, not to produce exact gradient values, so its constants are implementation-dependent while the scaling is not.

Time and memory. The unrolled backward replays every step, so its time is $O(K)$. It must also keep every step's parameters and optimizer state alive until the reverse pass reaches them, so its retained memory, held between the forward and the backward pass, is $O(KN)$ and grows by about 94 MB per inner step. Ours solves a single linear system with a fixed number of PCG iterations and stores one Jacobian cache and nothing per step, so its backward moves only from 171.6 ms at $K = 5$ to 179.8 ms at $K = 50$ and its retained memory stays at 0.03 GB.

4.4 Limitations

Similar to differentiable bundle-adjustment in VGGSfM [26], a notable limitation in our differentiable formulation is increased training time. This increases the per-step training time by a factor of $4\times$ to $5\times$ compared to the original feed-forward training. Note that our training implementation remains unoptimized for speed (the forward pass, which runs 50 (pose-given) or 100 (pose-free) gradient descent steps, makes up 65% of the additional time), and there is no resulting increase in runtime at test time.

5 Conclusion

We presented Diff3R, a framework that bridges feed-forward 3D Gaussian Splatting and per-scene test-time optimization (TTO) via an efficient, differentiable 3DGS optimization layer based on the Implicit Function Theorem and a matrix-free PCG solver. End-to-end training through TTO drives the network to predict initializations tailored for downstream refinement, and learnable per-Gaussian regularization weights adapt the anchoring strength to prevent overfitting on sparse context views. Although heavier in training than standard feed-forward, our formulation opens new directions, *e.g.* learning to refine 3DGS on-the-fly (3DGS SLAM) or extending uncertainty-aware optimization to dynamic 4D scenes and transient objects.

Acknowledgments

This work was supported by the ERC Starting Grant SpatialSem (101076253), the ERC Consolidator Grant Gen3D (101171131), and the Computing Systems Lab, part of Huawei Technologies Switzerland AG. The authors gratefully acknowledge the Gauss Centre for Supercomputing e.V. (`www.gauss-centre.eu`) for funding this project by providing computing time through the John von Neumann Institute for Computing (NIC) on the GCS Supercomputer JUWELS [54] at Jülich Supercomputing Centre (JSC). We also acknowledge the EuroHPC Joint Undertaking for providing access to the EuroHPC supercomputer MareNostrum 5, hosted by the Barcelona Supercomputing Center (BSC), Spain. Finally, we would like to thank Lukas Höllein for valuable discussions and his support with the 3DGS-LM code.

References

[1] B. Kerbl, G. Kopanas, T. Leimkühler, G. Drettakis *et al.*, "3d gaussian splatting for real-time radiance field rendering." *ACM Trans. Graph.*, vol. 42, no. 4, pp. 139:1–139:14, 2023.

[2] D. Charatan, S. L. Li, A. Tagliasacchi, and V. Sitzmann, "pixelsplat: 3d gaussian splats from image pairs for scalable generalizable 3d reconstruction," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2024, pp. 19 457–19 467.

[3] Y. Chen, H. Xu, C. Zheng, B. Zhuang, M. Pollefeys, A. Geiger, T.-J. Cham, and J. Cai, "Mvsplat: Efficient 3d gaussian splatting from sparse multi-view images," in *European conference on computer vision*. Springer, 2024, pp. 370–386.

[4] T. Liu, G. Wang, S. Hu, L. Shen, X. Ye, Y. Zang, Z. Cao, W. Li, and Z. Liu, "Mvsgaussian: Fast generalizable gaussian splatting reconstruction from multi-view stereo," in *European Conference on Computer Vision*. Springer, 2024, pp. 37–53.

[5] K. Zhang, S. Bi, H. Tan, Y. Xiangli, N. Zhao, K. Sunkavalli, and Z. Xu, "Gs-lrm: Large reconstruction model for 3d gaussian splatting," in *European Conference on Computer Vision*. Springer, 2024, pp. 1–19.

[6] H. Xu, S. Peng, F. Wang, H. Blum, D. Barath, A. Geiger, and M. Pollefeys, "Depthsplat: Connecting gaussian splatting and depth," in *Proceedings of the Computer Vision and Pattern Recognition Conference*, 2025, pp. 16 453–16 463.

[7] G. Kang, S. Nam, S. Yang, X. Sun, S. Khamis, A. Mohamed, and E. Park, "ilrm: An iterative large 3d reconstruction model," *arXiv preprint arXiv:2507.23277*, 2025.

[8] B. Ye, B. Chen, H. Xu, D. Barath, and M. Pollefeys, "Yonosplat: You only need one model for feedforward 3d gaussian splatting," *arXiv preprint arXiv:2511.07321*, 2025.

[9] B. Ye, S. Liu, H. Xu, X. Li, M. Pollefeys, M.-H. Yang, and S. Peng, "No pose, no problem: Surprisingly simple 3d gaussian splats from sparse unposed images," *arXiv preprint arXiv:2410.24207*, 2024.

[10] H. Lin, S. Chen, J. Liew, D. Y. Chen, Z. Li, G. Shi, J. Feng, and B. Kang, "Depth anything 3: Recovering the visual space from any views," *arXiv preprint arXiv:2511.10647*, 2025.

[11] L. Jiang, Y. Mao, L. Xu, T. Lu, K. Ren, Y. Jin, X. Xu, M. Yu, J. Pang, F. Zhao *et al.*, "Anysplat: Feed-forward 3d gaussian splatting from unconstrained views," *ACM Transactions on Graphics (TOG)*, vol. 44, no. 6, pp. 1–16, 2025.

[12] B. Smart, C. Zheng, I. Laina, and V. A. Prisacariu, "Splatt3r: Zero-shot gaussian splatting from uncalibrated image pairs," *arXiv preprint arXiv:2408.13912*, 2024.

[13] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," in *International conference on machine learning*. PMLR, 2017, pp. 1126–1135.

[14] L. Franceschi, P. Frasconi, S. Salzo, R. Grazzi, and M. Pontil, "Bilevel programming for hyperparameter optimization and meta-learning," in *International conference on machine learning*. PMLR, 2018, pp. 1568–1577.

[15] A. Rajeswaran, C. Finn, S. M. Kakade, and S. Levine, "Meta-learning with implicit gradients," *Advances in neural information processing systems*, vol. 32, 2019.

[16] J. Lorraine, P. Vicol, and D. Duvenaud, "Optimizing millions of hyperparameters by implicit differentiation," in *International conference on artificial intelligence and statistics*. PMLR, 2020, pp. 1540–1552.

[17] S. Wang, V. Leroy, Y. Cabon, B. Chidlovskii, and J. Revaud, "Dust3r: Geometric 3d vision made easy," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2024, pp. 20 697–20 709.

[18] V. Leroy, Y. Cabon, and J. Revaud, "Grounding image matching in 3d with mast3r," in *European conference on computer vision*. Springer, 2024, pp. 71–91.

[19] J. Wang, M. Chen, N. Karaev, A. Vedaldi, C. Rupprecht, and D. Novotny, "Vggt: Visual geometry grounded transformer," in *Proceedings of the Computer Vision and Pattern Recognition Conference*, 2025, pp. 5294–5306.

[20] N. Keetha, N. Müller, J. Schönberger, L. Porzi, Y. Zhang, T. Fischer, A. Knapitsch, D. Zauss, E. Weber, N. Antunes *et al.*, "Mapanything: Universal feed-forward metric 3d reconstruction," *arXiv preprint arXiv:2509.13414*, 2025.

[21] R. Huang and K. Mikolajczyk, "No pose at all: Self-supervised pose-free 3d gaussian splatting from sparse views," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2025, pp. 27 947–27 957.

[22] Y. Li, A. Božič, T. Zhang, Y. Ji, T. Harada, and M. Nießner, "Learning to Optimize Non-Rigid Tracking," in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2020, pp. 4909–4917.

[23] L. von Stumberg, P. Wenzel, N. Yang, and D. Cremers, "LM-Reloc: Levenberg-Marquardt Based Direct Visual Relocalization," in *2020 International Conference on 3D Vision (3DV)*, Nov. 2020, pp. 968–977.

[24] C. Tang and P. Tan, "BA-Net: Dense Bundle Adjustment Network," arXiv preprint arXiv:1806.04807, Aug. 2019.

[25] B. Roessle and M. Nießner, "End2end multi-view feature matching with differentiable pose optimization," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2023.

[26] J. Wang, N. Karaev, C. Rupprecht, and D. Novotny, "Vggsfm: Visual geometry grounded deep structure from motion," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2024, pp. 21 686–21 697.

[27] L. Pineda, T. Fan, M. Monge, S. Venkataraman, P. Sodhi, R. T. Chen, J. Ortiz, D. DeTone, A. Wang, S. Anderson, J. Dong, B. Amos, and M. Mukadam, "Theseus: A Library for Differentiable Nonlinear Optimization," *Advances in Neural Information Processing Systems*, 2022.

[28] A. Nichol, J. Achiam, and J. Schulman, "On first-order meta-learning algorithms," *arXiv preprint arXiv:1803.02999*, 2018.

[29] Z. Li, F. Zhou, F. Chen, and H. Li, "Meta-sgd: Learning to learn quickly for few-shot learning," *arXiv preprint arXiv:1707.09835*, 2017.

[30] V. Sitzmann, E. R. Chan, R. Tucker, N. Snavely, and G. Wetzstein, "Metasdf: Meta-learning signed distance functions," in *Proc. NeurIPS*, 2020.

[31] M. Tancik, B. Mildenhall, T. Wang, D. Schmidt, P. P. Srinivasan, J. T. Barron, and R. Ng, "Learned initializations for optimizing coordinate-based neural representations," in *CVPR*, 2021.

[32] Y. Chen, J. Wang, Z. Yang, S. Manivasagam, and R. Urtasun, "G3r: Gradient guided generalizable reconstruction," in *European Conference on Computer Vision*, 2024.

[33] Y.-C. Liu, L. Höllein, M. Nießner, and A. Dai, "Quicksplat: Fast 3d surface reconstruction via learned gaussian initialization," *arXiv preprint arXiv:2505.05591*, 2025.

[34] Y. Sun, X. Wang, Z. Liu, J. Miller, A. A. Efros, and M. Hardt, "Test-time training with self-supervision for generalization under distribution shifts," in *International Conference on Machine Learning (ICML)*, 2020.

[35] D. Wang, E. Shelhamer, S. Liu, B. Olshausen, and T. Darrell, "Tent: Fully test-time adaptation by entropy minimization," in *International Conference on Learning Representations (ICLR)*, 2021.

[36] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng, "NeRF: Representing scenes as neural radiance fields for view synthesis," in *European Conference on Computer Vision (ECCV)*, 2020.

[37] B. Duisterhof, L. Zust, P. Weinzaepfel, V. Leroy, Y. Cabon, and J. Revaud, "MASt3R-SfM: a fully-integrated solution for unconstrained structure-from-motion," *arXiv preprint arXiv:2409.19152*, 2024.

[38] X. Chen, Y. Chen, Y. Xiu, A. Geiger, and A. Chen, "Ttt3r: 3d reconstruction as test-time training," *arXiv preprint arXiv:2509.26645*, 2025.

[39] T. Zhang, S. Bi, Y. Hong, K. Zhang, F. Luan, S. Yang, K. Sunkavalli, W. T. Freeman, and H. Tan, "Test-time training done right," *arXiv preprint arXiv:2505.23884*, 2025.

[40] C. Wang, H. Tan, W. Yifan, Z. Chen, Y. Liu, K. Sunkavalli, S. Bi, L. Liu, and Y. Hu, "tttlrm: Test-time training for long context and autoregressive 3d reconstruction," *arXiv preprint arXiv:2602.20160*, 2026.

[41] T. Xie, P. Yang, Y. Jin, Y. Cai, W. Yin, W. Ren, Q. Zhang, W. Hua, S. Peng, X. Guo *et al.*, "Scal3r: Scalable test-time training for large-scale 3d reconstruction," *arXiv preprint arXiv:2604.08542*, 2026.

[42] Y. Yuan, Q. Shen, S. Wang, X. Yang, and X. Wang, "Test3r: Learning to reconstruct 3d at test time," *arXiv preprint arXiv:2506.13750*, 2025.

[43] S. Bai, J. Z. Kolter, and V. Koltun, "Deep equilibrium models," *Advances in neural information processing systems*, vol. 32, 2019.

[44] J. J. Moré, "The levenberg-marquardt algorithm: implementation and theory," in *Numerical analysis: proceedings of the biennial Conference held at Dundee, June 28–July 1, 1977*. Springer, 2006, pp. 105–116.

[45] L. Höllein, A. Božič, M. Zollhöfer, and M. Nießner, "3dgs-lm: Faster gaussian-splatting optimization with levenberg-marquardt," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2025, pp. 26 740–26 750.

[46] R. Ranftl, A. Bochkovskiy, and V. Koltun, "Vision transformers for dense prediction," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2021, pp. 12 179–12 188.

[47] T. Zhou, R. Tucker, J. Flynn, G. Fyffe, and N. Snavely, "Stereo magnification: Learning view synthesis using multiplane images," *arXiv preprint arXiv:1805.09817*, 2018.

[48] A. Dai, A. X. Chang, M. Savva, M. Halber, T. Funkhouser, and M. Nießner, "Scannet: Richly-annotated 3d reconstructions of indoor scenes," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 5828–5839.

[49] C. Yeshwanth, Y.-C. Liu, M. Nießner, and A. Dai, "Scannet++: A high-fidelity dataset of 3d indoor scenes," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023, pp. 12–22.

[50] L. Ling, Y. Sheng, Z. Tu, W. Zhao, C. Xin, K. Wan, L. Yu, Q. Guo, Z. Yu, Y. Lu *et al.*, "Dl3dv-10k: A large-scale scene dataset for deep learning-based 3d vision," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024, pp. 22 160–22 169.

[51] M. Roberts, J. Ramapuram, A. Ranjan, A. Kumar, M. A. Bautista, N. Paczan, R. Webb, and J. M. Susskind, "Hypersim: A photorealistic synthetic dataset for holistic indoor scene understanding," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2021, pp. 10 912–10 922.

[52] G. Baruch, Z. Chen, A. Dehghan, T. Dimry, Y. Feigin, P. Fu, T. Gebauer, B. Joffe, D. Kurz, A. Schwartz *et al.*, "Arkitscenes: A diverse real-world dataset for 3d indoor scene understanding using mobile rgb-d data," *arXiv preprint arXiv:2111.08897*, 2021.

[53] S. Umeyama, "Least-squares estimation of transformation parameters between two point patterns," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 13, no. 4, pp. 376–380, 2002.

[54] Jülich Supercomputing Centre, "JUWELS Cluster and Booster: Exascale Pathfinder with Modular Supercomputing Architecture at Juelich Supercomputing Centre," *Journal of large-scale research facilities*, vol. 7, no. A183, 2021. [Online]. Available: http://dx.doi.org/10.17815/jlsrf-7-183

[55] A. Liu, R. Tucker, V. Jampani, A. Makadia, N. Snavely, and A. Kanazawa, "Infinite nature: Perpetual view generation of natural scenes from a single image," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2021.

Supplementary Material

A Additional Qualitative Results

We provide additional qualitative results for the two-view, pose-given setting on the RE10K dataset in Figure A1. Visualizations of the predicted uncertainty weights demonstrate that Diff3R intrinsically

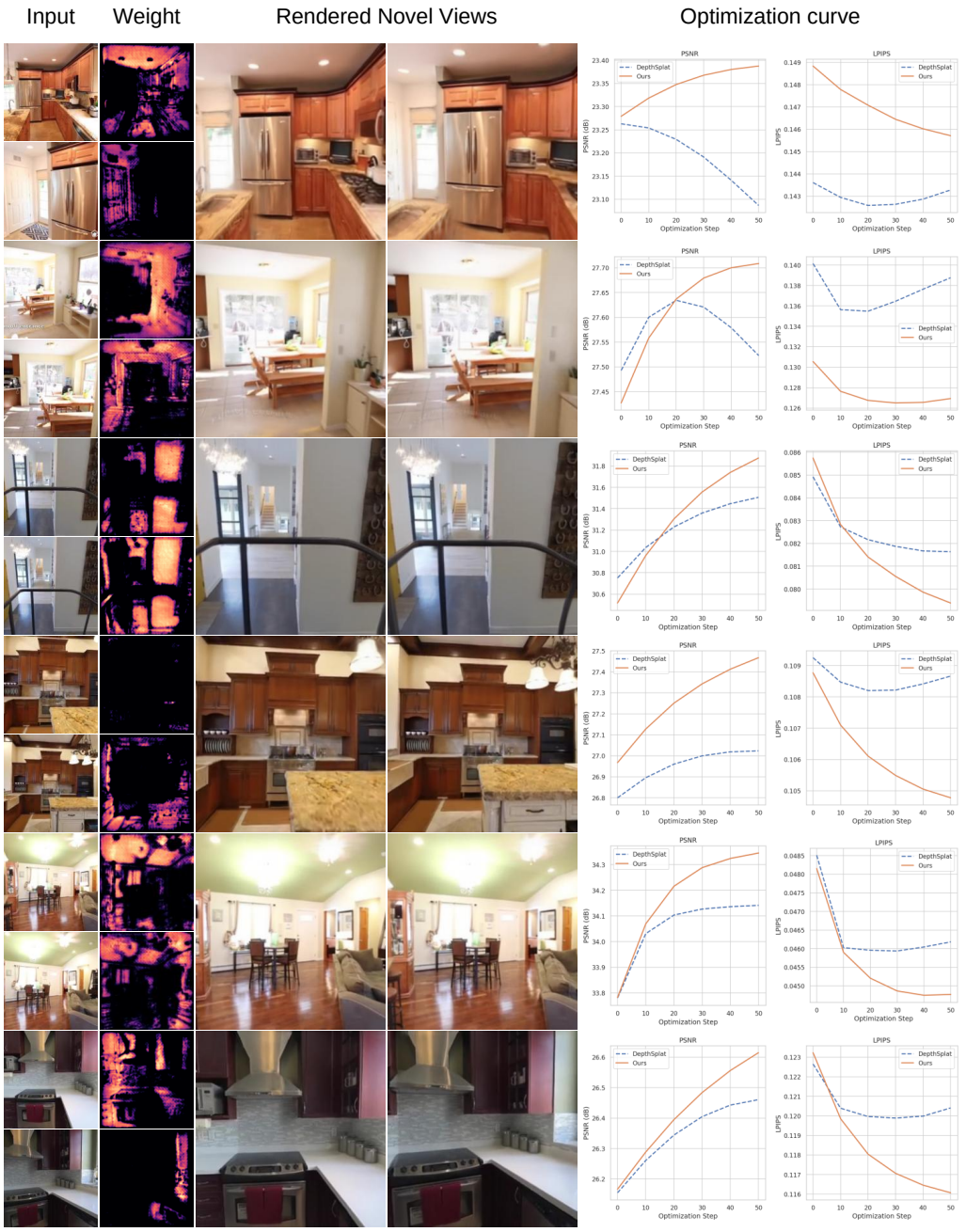

Figure A1: **More qualitative results on RE10K.** Our predicted uncertainty weights implicitly learn to penalize changes (*i.e.*, higher values) in non-overlapping regions during post-optimization. Crucially, we show that Diff3R learns a more effective initialization: while the initial predictions may yield worse zero-shot renderings, they result in higher-quality outputs following optimization.

learns to penalize parameter updates in regions lacking multi-view overlap. This adaptive regularization effectively anchors uncertain Gaussians, making the test-time optimization significantly more robust against overfitting to the input views.

Furthermore, the accompanying optimization curves illustrate our core idea. Rather than strictly maximizing zero-shot rendering quality, Diff3R predicts an optimization-aware initialization. By starting in a more favorable region of the loss landscape, our method consistently converges to a preferable local minimum, ultimately achieving higher-fidelity post-optimization renderings compared to the baseline.

B Detailed Derivation of the Implicit Gradient

In this section, we provide the derivation details of the implicit gradient used to backpropagate the outer loss through the inner test-time optimization (TTO) process.

B.1 Optimality Condition and Implicit Function Theorem

Recall that the inner optimization objective is defined as:

$$\mathcal{L}_{\text{inner}}(\Theta) = \mathcal{L}_{\text{photo}}(\Theta) + \frac{\lambda}{2}\|\Theta - \Theta_0\|_2^2, \tag{A1}$$

where $\mathcal{L}_{\text{photo}}(\Theta) = \frac{1}{2}\|r(\Theta)\|_2^2$ is the photometric loss based on the rendering residual $r(\Theta)$, Θ_0 is the feed-forward initialization, and λ is the proximal regularization weight.

We assume the inner optimization converges to a local minimum Θ^*. At this optimum, the derivative of the inner loss with respect to the Gaussian parameters Θ must be exactly zero. This gives us the optimality condition:

$$\nabla_\Theta \mathcal{L}_{\text{inner}}(\Theta^*) = \nabla_\Theta \mathcal{L}_{\text{photo}}(\Theta^*) + \lambda(\Theta^* - \Theta_0) = \mathbf{0}. \tag{A2}$$

To compute how a change in the initialization Θ_0 affects the optimal parameters Θ^*, we differentiate both sides of Equation (A2) with respect to Θ_0. Applying the multivariate chain rule yields:

$$\nabla_\Theta^2 \mathcal{L}_{\text{photo}}(\Theta^*)\frac{\partial \Theta^*}{\partial \Theta_0} + \lambda \frac{\partial \Theta^*}{\partial \Theta_0} - \lambda I = \mathbf{0}, \tag{A3}$$

where I is the identity matrix. Let $H_{\text{photo}} = \nabla_\Theta^2 \mathcal{L}_{\text{photo}}(\Theta^*)$ denote the exact Hessian of the photometric loss. We can factor out the Jacobian $\frac{\partial \Theta^*}{\partial \Theta_0}$:

$$(H_{\text{photo}} + \lambda I)\frac{\partial \Theta^*}{\partial \Theta_0} = \lambda I. \tag{A4}$$

Assuming the regularized Hessian $(H_{\text{photo}} + \lambda I)$ is invertible (which is guaranteed for a sufficiently large proximal weight λ as it strictly convexifies the local landscape), we obtain the closed-form Jacobian of the optimal parameters with respect to the initialization:

$$\frac{\partial \Theta^*}{\partial \Theta_0} = \lambda (H_{\text{photo}} + \lambda I)^{-1}. \tag{A5}$$

B.2 Gauss-Newton Approximation

Computing the exact Hessian H_{photo} requires second-order derivatives of the rendering process, which is computationally intractable for 3D Gaussian Splatting. Because the photometric loss is a sum-of-squares residual objective ($\frac{1}{2}\|r(\Theta)\|_2^2$), we can employ the Gauss-Newton approximation.

By definition, the exact Hessian is $H_{\text{photo}} = J^\top J + \sum_i r_i \nabla^2 r_i$, where $J = \frac{\partial r(\Theta^*)}{\partial \Theta^*}$ is the Jacobian of the residual. Assuming the residuals r_i are small near the optimum, we can drop the second-order term, yielding:

$$H_{\text{photo}} \approx J^\top J. \tag{A6}$$

Substituting this back into Equation (A5), we get our workable Jacobian formulation:

$$\frac{\partial \Theta^*}{\partial \Theta_0} \approx \lambda (J^\top J + \lambda I)^{-1}. \tag{A7}$$

B.3 Vector-Jacobian Product and PCG Objective

To update the feed-forward network weights, we need the gradient of the outer loss $\mathcal{L}_{\text{outer}}$ with respect to Θ_0. Using the chain rule, this is the Vector-Jacobian Product (VJP) between the outer loss gradient and the implicit Jacobian:

$$\nabla_{\Theta_0}\mathcal{L}_{\text{outer}} = \frac{\partial\Theta^*}{\partial\Theta_0}^\top \nabla_{\Theta^*}\mathcal{L}_{\text{outer}} = \lambda(J^\top J + \lambda I)^{-1}\nabla_{\Theta^*}\mathcal{L}_{\text{outer}}. \tag{A8}$$

Note that $(J^\top J + \lambda I)$ is symmetric, so its transpose is itself.

Explicitly instantiating and inverting the matrix $(J^\top J + \lambda I)$ is impossible since it scales quadratically with the number of Gaussian parameters ($N \times N$, where N is often in the millions). Instead, we treat the term $(J^\top J + \lambda I)^{-1}\nabla_{\Theta^*}\mathcal{L}_{\text{outer}}$ as an unknown vector v.

We define v such that:

$$v = (J^\top J + \lambda I)^{-1}\nabla_{\Theta^*}\mathcal{L}_{\text{outer}}. \tag{A9}$$

Multiplying both sides by the regularized Hessian transforms the matrix inversion problem into a large-scale system of linear equations:

$$(J^\top J + \lambda I)v = \nabla_{\Theta^*}\mathcal{L}_{\text{outer}}. \tag{A10}$$

Once PCG converges to a solution for v, substituting it back into Equation (A8) yields the final, exact gradient to be passed back to the network:

$$\nabla_{\Theta_0}\mathcal{L}_{\text{outer}} = \lambda v. \tag{A11}$$

B.4 Gradients for the Uncertainty Weights

When transitioning from a global scalar λ to the per-parameter uncertainty regularizer $\mathbf{\Lambda} \in \mathbb{R}^{N_\Theta}$, the inner loss becomes:

$$\mathcal{L}_{\text{inner}}(\Theta) = \mathcal{L}_{\text{photo}}(\Theta) + \frac{1}{2}(\Theta - \Theta_0)^\top \text{diag}(\mathbf{\Lambda})(\Theta - \Theta_0). \tag{A12}$$

At convergence, the new optimality condition is:

$$\nabla_\Theta\mathcal{L}_{\text{inner}}(\Theta^*) = \nabla_\Theta\mathcal{L}_{\text{photo}}(\Theta^*) + \text{diag}(\mathbf{\Lambda})(\Theta^* - \Theta_0) = \mathbf{0}. \tag{A13}$$

To find the implicit gradient of the optimal parameters with respect to the uncertainty weights, we differentiate this optimality condition with respect to $\mathbf{\Lambda}$. Applying the product rule yields:

$$\nabla^2_\Theta\mathcal{L}_{\text{photo}}(\Theta^*)\frac{\partial\Theta^*}{\partial\mathbf{\Lambda}} + \text{diag}(\mathbf{\Lambda})\frac{\partial\Theta^*}{\partial\mathbf{\Lambda}} + \text{diag}(\Theta^* - \Theta_0) = \mathbf{0}. \tag{A14}$$

Substituting the Gauss-Newton approximation $H_{\text{photo}} \approx J^\top J$ and rearranging the terms allows us to solve for the Jacobian $\frac{\partial\Theta^*}{\partial\mathbf{\Lambda}}$:

$$(J^\top J + \text{diag}(\mathbf{\Lambda}))\frac{\partial\Theta^*}{\partial\mathbf{\Lambda}} = -\text{diag}(\Theta^* - \Theta_0) \tag{A15}$$

$$\frac{\partial\Theta^*}{\partial\mathbf{\Lambda}} = -(J^\top J + \text{diag}(\mathbf{\Lambda}))^{-1}\text{diag}(\Theta^* - \Theta_0). \tag{A16}$$

Next, we apply the chain rule to compute the gradient of the outer loss with respect to $\mathbf{\Lambda}$. This requires multiplying the transpose of our Jacobian by the outer loss gradient:

$$\nabla_{\mathbf{\Lambda}}\mathcal{L}_{\text{outer}} = \left(\frac{\partial\Theta^*}{\partial\mathbf{\Lambda}}\right)^\top \nabla_{\Theta^*}\mathcal{L}_{\text{outer}}. \tag{A17}$$

Substituting Equation (A16) into this equation gives:

$$\nabla_{\mathbf{\Lambda}}\mathcal{L}_{\text{outer}} = -\left((J^\top J + \text{diag}(\mathbf{\Lambda}))^{-1}\text{diag}(\Theta^* - \Theta_0)\right)^\top \nabla_{\Theta^*}\mathcal{L}_{\text{outer}}. \tag{A18}$$

Because the approximated Hessian $(J^\top J + \text{diag}(\mathbf{\Lambda}))$ is symmetric, its inverse is also symmetric. Likewise, the diagonal matrix $\text{diag}(\Theta^* - \Theta_0)$ is symmetric. Thus, the transpose operation simply reverses the order of the matrices without altering their contents:

$$\nabla_{\mathbf{\Lambda}} \mathcal{L}_{\text{outer}} = -\text{diag}(\Theta^* - \Theta_0) \underbrace{(J^\top J + \text{diag}(\mathbf{\Lambda}))^{-1} \nabla_{\Theta^*} \mathcal{L}_{\text{outer}}}_{v}. \tag{A19}$$

Notice that the rightmost term is exactly the auxiliary vector v we already solved for via the matrix-free PCG solver (adapted from Equation (A10) to use the diagonal $\mathbf{\Lambda}$ matrix):

$$v = (J^\top J + \text{diag}(\mathbf{\Lambda}))^{-1} \nabla_{\Theta^*} \mathcal{L}_{\text{outer}}. \tag{A20}$$

Substituting v back into the equation simplifies the expression significantly:

$$\nabla_{\mathbf{\Lambda}} \mathcal{L}_{\text{outer}} = -\text{diag}(\Theta^* - \Theta_0) v. \tag{A21}$$

Finally, multiplying a diagonal matrix by a column vector is mathematically equivalent to computing the element-wise product of the two vectors. This yields our final, highly efficient analytical gradient:

$$\nabla_{\mathbf{\Lambda}} \mathcal{L}_{\text{outer}} = -v \odot (\Theta^* - \Theta_0). \tag{A22}$$

By similar logic, the gradient for the initialization Θ_0 under the per-parameter formulation cleanly updates to $\nabla_{\Theta_0} \mathcal{L}_{\text{outer}} = \mathbf{\Lambda} \odot v$. Because both gradients reuse the exact same PCG solution v, computing the uncertainty gradients incurs virtually zero additional computational overhead.

B.5 Discussion: relation to Levenberg-Marquardt and behavior at extremes of λ

It is worth noting the mathematical similarity between our implicit gradient formulation and standard second-order optimization techniques. The left-hand side of our linear system in Equation (4) is structurally identical to the normal equations solved during a Levenberg-Marquardt (LM) optimization step [44]. Therefore, we leverage the efficient CUDA-based LM solver for 3DGS [45] to speed up our implementation.

In LM, $(J^\top J + \lambda I)$ is computed for the update step, where λ acts as a damping factor blending Gauss-Newton and gradient-descent directions. In our bilevel framework, computing the backward pass of the outer loss through the proximal inner optimization mimics this exact damped step. The proximal weight λ from our inner loss translates into the LM damping factor during the backward pass, ensuring that the gradient flow back to the feed-forward network remains well-conditioned and robust, even when the outer novel-view supervision is noisy or sparse.

Furthermore, analyzing the extreme values of λ provides a clear physical intuition for how the gradient behaves during backpropagation:

- **Strong prior (large λ):** The optimization is heavily restricted, forcing the final Gaussians Θ^* to stay near their initialization Θ_0. The gradient passes almost directly through to the network ($\nabla_{\Theta_0} \mathcal{L}_{\text{outer}} \approx \nabla_{\Theta^*} \mathcal{L}_{\text{outer}}$), effectively bypassing the inner optimization loop.
- **Weak prior ($\lambda \to 0$):** The Gaussians freely fit the context views and "forget" their initialization. Because the final outcome becomes independent of the starting point, the gradient vanishes ($\nabla_{\Theta_0} \mathcal{L}_{\text{outer}} \to \mathbf{0}$), cutting off the learning signal entirely.

This trade-off motivates our learnable per-parameter uncertainty regularizer $\mathbf{\Lambda}$, which lets the model dynamically balance test-time flexibility with a robust gradient flow.

B.6 Discussion: relation to Meta-SGD

$\mathbf{\Lambda}$ shares the spirit of Meta-SGD's [29] learned per-parameter adaptation. Meta-SGD learns a single vector α, of the same size as the parameter vector, jointly with the initialization over the task distribution, and applies it as one unrolled step $\Theta_0 - \alpha \odot \nabla \mathcal{L}$, with the meta-gradient taken by backpropagating through that step. Both control the per-parameter behavior of the inner optimization via learning from data, but $\mathbf{\Lambda}$ differs on three counts. First, it is predicted from the input rather than learned once for the whole task distribution. Secondly, it enters the inner objective (Equation (5)),

so Θ^* is a minimizer, whereas α appears only in the update rule and never in an objective. Lastly, the effect of $\mathbf{\Lambda}$ is coupled across Gaussians, while that of α is not. In Meta-SGD, the update of each parameter, $-\alpha_i \nabla_i \mathcal{L}$, depends only on its own α_i. In our case, $\Theta^* - \Theta_0 \approx -(J^\top J + \text{diag}(\mathbf{\Lambda}))^{-1} \nabla \mathcal{L}_{\text{photo}}(\Theta_0)$, where $J^\top J$ links Gaussians that render the same pixels. For example, if two Gaussians cover the same pixel and $\mathbf{\Lambda}$ anchors one of them, the other Gaussian would have to move further to compensate.

C Implementation Details

Residuals. In practice, rather than a pure L2 loss, we employ a combination of L1 and SSIM losses for the inner TTO objective, following the standard 3DGS optimization, to improve visual fidelity and preserve high-frequency structures. The definition of the residual $r(\Theta)$ and its corresponding Jacobian J are adjusted accordingly via chain rules.

Two-stage training. To ensure stable convergence and avoid degenerate solutions early in training, we adopt a two-stage training strategy. We first pretrain the feed-forward network g_ϕ purely on the outer loss without the differentiable optimization (equivalent to setting $\Theta^* = \Theta_0$). Once the model generates a reasonable initialization, we activate the inner TTO loop and fine-tune the Gaussian DPT head and the uncertainty modules.

To further stabilize training, we apply an auxiliary rendering loss directly to the feed-forward predictions Θ_0, prior to the differentiable optimization layer. Although the optimal initialization for test-time optimization may not perfectly align with the best zero-shot rendering, this proxy loss anchors the network's predictions. Scaled by a down-weighting factor of $\lambda_{\text{proxy}} = 0.1$, this supervision prevents the model from generating degenerate or unrecoverable Gaussian configurations early in training.

Backward path details. We implement our matrix-free PCG solver utilizing the efficient CUDA kernels from 3DGS-LM [45]. To ensure stable gradients within the linear system, we normalize the Jacobian matrix J by the total number of rendering residuals.

In practice, the attributes of a 3D Gaussian (e.g., position, scale, and spherical harmonics) exhibit vastly different scales and gradient magnitudes. To properly balance the regularization across these heterogeneous parameter types, we modulate the proximal weight using a Levenberg-Marquardt heuristic. Specifically, the effective proximal term used in our system is formulated as $\lambda \cdot \text{diag}(\mathbf{\Lambda} \odot M)$, where $M \in \mathbb{R}^{N_\Theta}$ tracks the moving average of the diagonal of $J^\top J$ during training. This ensures a parameter-aware scaling: attributes that dictate strong changes in the photometric loss, such as the Gaussian means, produce larger values in M, thereby receiving a proportionally stronger regularization bound. This technique harmonizes the scale differences across the Gaussian parameters, stabilizing both the inner test-time optimization and the outer PCG gradient computation.

Training hyperparameters. We optimize the outer-loop parameters ϕ with AdamW at a learning rate of 1×10^{-5} with a cosine-annealing schedule, training for 100K iterations. The batch size is 8 per GPU in the two-view setting and 4 per GPU in the four-view setting. We train our models on $4\times$ server-grade GPUs with 80 GB HBM each for approximately 5 days per setting. The outer LPIPS weight is $\lambda_{\text{lpips}} = 0.05$, and we apply simple input augmentation (random scaling and cropping).

Solver hyperparameters. The backward linear solve uses our matrix-free PCG solver with 15 CG iterations, and the Levenberg-Marquardt damping in $\lambda \cdot \text{diag}(\mathbf{\Lambda} \odot M)$ is set to $\lambda = 1 \times 10^{-4}$.

D Generalization

We evaluate zero-shot transfer to ACID [55] with the RE10K model and a higher view count on DL3DV [50] with four context views at 256×448. Our method improves over DepthSplat in both settings (Table A1).

Table A1: **Generalization.** PSNR after TTO on ACID [55] (two views, zero-shot with the RE10K model) and DL3DV [50] (four views, 256×448).

	ACID, zero-shot	DL3DV, 4 views
DepthSplat [6]	28.459	26.003
Ours	**28.591**	**26.179**

E Broader Impacts

Our method improves the quality of feed-forward 3D Gaussian Splatting and could benefit AR/VR content creation, robotics simulation, novel-view synthesis for cultural-heritage capture, and digital-twin applications. As with any photorealistic 3D reconstruction technique, downstream misuse for synthetic media or surveillance is conceivable, but our contribution operates at the optimization-layer level and does not lower the barrier to such misuse beyond the existing 3DGS literature. We use only publicly released, peer-reviewed scene datasets and do not introduce a generative model that synthesizes new identities or scenes from scratch.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and Section 1 state our main contributions (the bilevel optimization framework, the implicit-gradient differentiable 3DGS layer, and uncertainty-aware learnable regularization), as well as the empirical claims, all of which are supported by Section 4 and the ablations therein.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Section 4 contains a dedicated "Limitations" subsection discussing the 4–5$\times$ training-time overhead introduced by the differentiable inner loop, and notes that there is no resulting increase at test time.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [Yes]

Justification: Section 3 states the assumptions used for the implicit gradient (inner-loop optimality, Gauss–Newton approximation, regularized Hessian invertibility); the full derivations and the relationship to Levenberg–Marquardt are provided in Sections B and B.5 of the supplementary material.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: Section 4 specifies the datasets (RE10K, ScanNet, ScanNet++, DL3DV-10K, Hypersim, ArkitScenes), the 256×256 input resolution, baselines, two-stage training schedule, TTO step counts (50 pose-given / 300 pose-free), the L1+SSIM photometric loss with proximal regularization, and $\lambda_{\text{proxy}} = 0.1$; remaining hyperparameters and architecture details are listed in Section C.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).

(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [No]

   Justification: The code is not yet publicly available; we plan to release it on the project page. All datasets used are publicly available, and the algorithm and implementation details are fully described in Sections B to C.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: Section 4 specifies datasets, splits, resolution, the L1+SSIM photometric loss, $\lambda_{\text{proxy}} = 0.1$, TTO step counts (50/300), the two-stage training schedule, and the architecture (DepthSplat / Depth Anything v3 + DPT head + uncertainty MLP). Optimizer, learning rate, batch size, training iterations, λ_{lpips}, PCG iteration count, LM damping, and data augmentation are listed in Section C.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [No]

Justification: Following common practice in feed-forward 3D reconstruction (e.g., PixelSplat, MVSplat, DepthSplat, AnySplat), we report single-run results on standard validation splits without error bars due to the cost of training multiple full models in the paper. We tried retraining the RE10K pose-given setting with three seeds, which gives 27.947 ± 0.022 PSNR after TTO.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Section 4 reports the relative training-time overhead (4–5$\times$ per step over the feed-forward baseline, with the forward pass accounting for $\sim$65% of the added cost) and confirms there is no test-time overhead. Hardware (4$\times$ server GPU with 80 GB HBM each) and total wall-clock training time ($\sim$5 days for 100K iterations) are listed in Section C.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The work uses only publicly available scene-reconstruction datasets, involves no human subjects, and complies with the NeurIPS Code of Ethics in all respects.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.

- If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: Section E of the supplementary discusses both positive applications (AR/VR content creation, robotics simulation, novel-view synthesis for cultural-heritage capture, digital twins) and the limited risk of misuse, noting that our contribution operates at the optimization-layer level and does not lower the barrier for malicious synthetic-media generation beyond existing 3DGS work.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: The paper releases neither a generative language model, an image generator, nor a scraped dataset; we use only public, peer-reviewed scene-reconstruction benchmarks.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: All datasets (RE10K, ScanNet, ScanNet++, DL3DV-10K, Hypersim, ArkitScenes) and pretrained models (DepthSplat, Depth Anything v3) are cited at first use in Sections 3 and 4 and used under their respective public research licenses; we follow each asset's terms of use.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [N/A]

    Justification: The paper does not release new assets. The code will be released on the project page.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing or research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.

- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: The paper does not involve research with human subjects, so IRB approval is not applicable.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: LLMs were used only for coding assistance and writing/editing support; they are not an important, original, or non-standard component of the core methods, so per the NeurIPS LLM policy declaration is not required.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.