DN-Splatter: Depth and Normal Priors for Gaussian Splatting and Meshing

Matias Turkulainen*[1], Xuqian Ren*[2], Iaroslav Melekhov[3], Otto Seiskari[4], Esa Rahtu[2,4], and Juho Kannala[3,4]

[1] ETH Zurich, [2] Tampere University, [3] Aalto University, [4] Spectacular AI
Corresponding author: matiasturkulainen@gmail.com
* Denotes equal contribution

Abstract. 3D Gaussian splatting, a novel differentiable rendering technique, has achieved state-of-the-art novel view synthesis results with high rendering speeds and relatively low training times. However, its performance on scenes commonly seen in indoor datasets is poor due to the lack of geometric constraints during optimization. We extend 3D Gaussian splatting with depth and normal cues to tackle challenging indoor datasets and showcase techniques for efficient mesh extraction, an important downstream application. Specifically, we regularize the optimization procedure with depth information, enforce local smoothness of nearby Gaussians, and use the geometry of the 3D Gaussians supervised by normal cues to achieve better alignment with the true scene geometry. We improve depth estimation and novel view synthesis results over baselines and show how this simple yet effective regularization technique can be used to directly extract meshes from the Gaussian representation yielding more physically accurate reconstructions on indoor scenes. Our code will be released in https://github.com/maturk/dn-splatter.

Keywords: Gaussian splatting · indoor reconstruction · prior regularization

1 Introduction

Photorealistic and accurate 3D reconstruction of common indoor scenes from casually captured sensor data has been a long lasting problem in 3D computer vision and computer graphics. Textureless and less-observed regions cause ambiguities in reconstructions and do not provide enough constraints for valid geometric solutions. Recently, great success has been achieved with neural implicit representations that represent scenes as a continuous volume with fully differentiable geometric and appearance properties like color and density [29]. However, reconstruction of common everyday indoor scenes still remains a daunting problem, even for state-of-the-art methods, with methods rarely achieving good results in both photorealism and geometry reconstruction and far too often suffering from long training and rendering times, making them inaccessible for general use.

3D Gaussian splatting [16] introduces a novel method for inverse rendering by representing a scene by many differentiable 3D Gaussian primitives with

optimizable geometric and appearance properties. This explicit representation enables real time rendering capabilities of large complex scenes; a capability that most neural implicit models lack, and results in a more interoperable scene representation since the scene appearance and geometry are directly expressed by the location, shape, and color attributes of Gaussians. However, due to the lack of 3D cues and surface constraints during optimization, artifacts and ambiguities are likely to occur. Scenes often contain millions of Gaussians and their properties are directly modified by gradient descent based on photometric losses only. Little focus has been given to exploring better regularization techniques that result in visually and geometrically smoother and more plausible 3D reconstructions which could be converted into mesh representations, an important downstream application.

Motivated by progress in general monocular depth [1, 50] and normal estimation networks [5] and due to the abundance of mobile devices equipped with time-of-flight and depth sensors, we explore regularization of 3D Gaussian splatting with these geometric priors, with the aim of enhancing photorealism and surface reconstruction. We regularize the optimization of 3D Gaussian splatting in indoor scenes with depth and smoothness constraints enhancing novel view synthesis results whilst respecting the captured scene geometry better. We estimate normals from Gaussians and orient them based on scene geometry estimated from monocular normal priors to help align Gaussians with the real surface boundaries. We show how this simple regularization strategy, illustrated in Fig. 1, can be directly used to extract meshes from the Gaussian scene representation, yielding reconstructions that are smoother and more geometrically valid.

We summarize our contributions with the following statements:

- We incorporate depth and smoothness priors to Gaussian splatting optimization to improve novel view synthesis and reconstruction on indoor scenes.
- We use monocular normal priors to align Gaussians with the scene geometry and show how this results in more accurate geometry reconstructions.
- We showcase how this regularization strategy enables efficient mesh extraction directly from the optimized Gaussian scene.

2 Related Work

Here we give a brief introduction to image-based rendering (IBR) methods for scene reconstruction and an overview of previous regularization strategies, mainly for NeRF and neural implicit models, to enhance geometry reconstruction. We then give an overview of the current state-of-the-art for Gaussian surface extraction.

2.1 Traditional IBR

Traditional techniques enabled by Structure-from-Motion (SfM) [37, 40] and multi-view stereo (MVS) [8] techniques focused on reconstructing geometry as

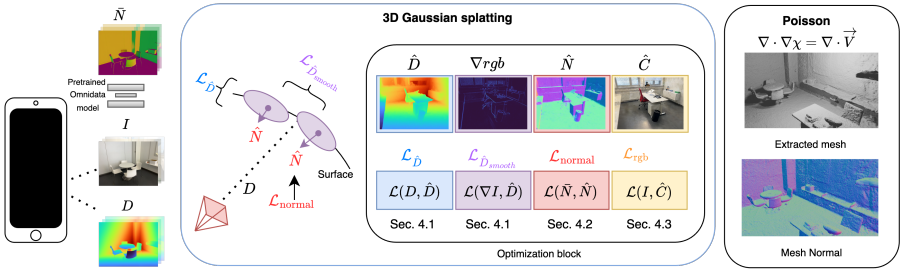


Fig. 1: Overview: We use depth and normal priors obtained from common handheld devices and general-purpose networks to enhance Gaussian splatting reconstruction quality. We regularize Gaussian positions, local smoothness, and orientations and demonstrate how this approach leads to more accurate mesh reconstructions directly from the scene representation.

3D points from images [19, 36]. Work focused on dense point reconstruction and normal estimation [38]. Several techniques construct triangle meshes from point sets [15, 27] to render novel views.

2.2 Neural implicit IBR

Most success has been achieved with neural based inverse rendering methods, most notably that of NeRF [29], which apply volume rendering [14] to represent scenes as continuous volumes with density and color attributes encoded within a neural network. However, the 3D geometry extracted from these scenes are often ill-defined and suffer from artifacts. Subsequent work has focused on improving the rendering quality and scene reconstruction with regularization techniques and adapting other scene representations such as SDFs or occupancy grids.

Prior regularization: Prior regularization of neural implicit models has been an active area of research. Previous NeRF-based approaches add depth regularization to explicitly supervise ray termination [4, 49] or impose smoothness constraints [31] on rendered depth maps. Other works also explore regularizing with multi-view consistency [4, 9, 24] in sparse view settings. For signed distance function (SDF) based models, Manhattan-SDF [10] uses planar constraints on walls and flat surfaces to improve reconstruction in indoor datasets and MonoSDF [55] uses both depth and normal monocular estimates for regularizing scene geometry. These works have shown compelling results by adding priors to inverse rendering. In this work, we explore regularizing 3D Gaussian splatting optimization with depth and normal priors to enhance photometric and geometric reconstruction.

Meshable implicit representations: Surface extraction as triangle meshes is an important problem since most computer graphics pipelines still rely on triangle rasterization. Watertight meshes also provide a good approximation of scene geometry and surface quality, thus various metrics for mesh quality and geometry evaluation have been devised. Prior work has focused on extracting meshes

from NeRF representations [34, 43] with some success; however, often relies on expensive post training and refinement stages. Some work, like in [42], attempts to extract meshes directly from NeRFs by predicting normals and surface intersections points that can be meshed with Poisson surface reconstruction [15]. However, most state-of-the-art techniques use SDF or occupancy representations [25, 47, 51, 55] combined with marching cubes [27] to achieve finer details where volume density based methods fail. These methods rely on querying and evaluating dense 3D volumes, often at multiple levels of detail and resolutions, and thus are generally slow to train and extract meshes from. In this work, we investigate extracting meshable surfaces directly from the explicit Gaussian scene representation.

Meshable 3D Gaussians: Extracting meshable surfaces from Gaussian primitives is a relatively new topic. Keselman *et al.* [17] propose to extract a mesh as a post-processing step from a trained Gaussian scene by simplifying the problem to generate an oriented point set (points with normals) that can be meshed with Poisson surface reconstruction [15]. To create an oriented point set, the authors propose to back-project rendered depth maps and to estimate normals analytically by differentiating rendered depth maps or by estimating normals from the derivative of the Gaussian densities. Through our analysis, Tab. 1a and Tab. 7, we found that with no regularization on rendered depths and normal estimates, this approach results in noisy point clouds that are difficult to mesh, especially in larger indoor datasets. NeuSG [2] attempts to solve this by jointly training a dense SDF neural implicit model with the Gaussian scene, and aligning the Gaussians with the positions and normals estimated by the SDF model. Although this method achieves good reconstructions; its reliance on the neural implicit model makes training times exceed that of 16 hours on high-end GPUs [2], diminishing the appeal of 3DGS.

Instead, SuGaR [11] proposes to treat the positions of Gaussians and their densities as estimates of intersections of a level set representing the surface of the scene and they optimise a signed-distance loss to encourage Gaussians to converge to the real surface and to be well spread during optimization. The authors also propose to estimate normals from the derivative of the signed distance, similar to Keselman *et al.* However, due to the lack of geometric priors, the reconstructions extracted using this method are still imperfect and noisy, since the densities and thus normals estimated at surface points are not smooth. SuGaR proposes to further refine the coarse mesh obtained from the scene by differentiable optimization of the mesh vertices and faces, making the entire training and mesh extraction process computationally costly. Instead, we investigate how to improve the coarse mesh extracted directly from the Gaussian scene and show how using depth and normal priors during optimization can greatly improve the reconstruction results.

3 Preliminaries

Our work builds on 3D Gaussian splatting (3DGS, [16]) and in this section we describe the rasterization algorithm and mathematical notation. 3DGS represents a scene with many differentiable 3D Gaussian primitives parameterized by their mean $\boldsymbol{\mu} \in \mathbb{R}^3$, covariance matrix $\boldsymbol{\Sigma} \in \mathbb{R}^{3\times3}$ decomposed into a scaling vector $\boldsymbol{s} \in \mathbb{R}^3$ and a rotation quaternion $\boldsymbol{q} \in \mathbb{R}^4$, opacity $o \in \mathbb{R}$, and color $\boldsymbol{c} \in \mathbb{R}^3$ represented via spherical harmonics. Rendering a new view from this representation consists of projecting 3D Gaussians into camera space as 2D Gaussians. The 2D Gaussians are then sorted by z-depth in a single global sorting operation and alpha-composited using the discrete volume rendering equation to produce pixel colors $\hat{\mathbf{C}}$:

$$\hat{\mathbf{C}} = \sum_{i \in N} \mathbf{c}_i \alpha_i T_i, \text{ where } T_i = \prod_{j=1}^{i-1} (1 - \alpha_j) \tag{1}$$

where T_i is the accumulated transmittance at pixel location p and α_i is the blending coefficient for a Gaussian with center μ_i in screen space:

$$\alpha_i = o_i \cdot \exp\left(\frac{1}{2}(\boldsymbol{p} - \boldsymbol{\mu}_i)^\intercal \boldsymbol{\Sigma}_i^{-1} (\boldsymbol{p} - \boldsymbol{\mu}_i)\right) \tag{2}$$

The α_i term is referred to as density in some work [11]. Gaussian projection and blending operations are parallelized allowing for real-time rendering performance.

The scene is initialized with Gaussian means and colors obtained from SfM [37, 38] and initial covariance matrices are initialised with axes equal to the average distance of a Gaussian to its closest three neighbours. During optimization, the parameters of individual Gaussian's are updated via gradient descent by many rendering iterations to best fit the training dataset. The algorithm progressively culls, splits, and duplicates Gaussians in the scene at fixed iterations based on Gaussian opacity, screen-space size, and the magnitude of the gradient of Gaussian means respectively.

4 Method

Our approach optimizes 3D Gaussian splatting with depth and normal priors to better represent the underlying geometry of indoor scenes. To this end, we introduce sensor and monocular depth priors to regularize Gaussian positions and enforce smoothness constraints on rendered depth maps in Sec. 4.1. Next, we show how to use the geometry of the Gaussians to extract normal directions and utilize monocular normal cues for regularization in Sec. 4.2. We further discuss loss functions and the overall optimization process in Sec. 4.3. Lastly in Sec. 4.4 we use the optimized Gaussian scene to directly extract meshes using Poisson surface reconstruction without any additional optimization or refinement stages.

4.1 Leveraging Depth Cues

Per-pixel z–depth estimates $\hat{D}$ are rendered using the discrete volume rendering approximation similar to color values:

$$\hat{D} = \sum_{i \in N} d_i \alpha_i \prod_{j=1}^{i-1} (1 - \alpha_j) \tag{3}$$

where d_i is the i^{th} Gaussian z-depth coordinate in view space. Since 3DGS does not sort Gaussians individually per-pixel along a viewing ray, and instead relies on a single global sort for efficiency; this is only an approximation of per-pixel depth. The z-depth ordering for close by Gaussians and specific orientations is not guaranteed to be correct after 2D projection and a global sort, as explained in StopThePop [33]; however, sorting by depth for each pixel would be too computationally costly since scenes can contain millions of Gaussians. Nevertheless, this approximation still remains effective, particularly for more regular geometries typically encountered in indoor datasets. We normalize depth estimates with the final accumulated transmittance T_i per-pixel, ensuring that we correctly estimate a depth value for pixels where the accumulated transmittance does not equal 1. We rasterize color and depths simultaneously per-pixel in a single CUDA kernel forward pass, improving inference and training speed compared to separate rendering steps.

Regularization with sensor depth: We directly apply depth regularization on predicted depth maps for datasets containing lidar or sensor depth measurements [35, 41, 53]. Common commercial depth sensors, especially low-resolution variants found in consumer devices like iPhones, contain non-smooth edges at object boundaries and often provide inaccurate readings on smooth surfaces. Based on this observation and inspired by [3, 20], we propose a gradient-aware depth loss for adaptive depth regularization based on the current RGB image. The depth loss is lowered in regions with large image gradients, signifying edges, ensuring that regularization is more enforced on smoother texture-less regions that typically pose challenges to only photometric loss. In addition, through experiments (Tab. 4) we found that a logarithmic penalty resulted in smoother reconstructions compared to linear or quadratic ones. We formulate our gradient-aware depth loss as follows:

$$\mathcal{L}_{\hat{D}} = g_{\text{rgb}} \frac{1}{|\hat{D}|} \sum \log(1 + \|\hat{D} - D\|_1) \tag{4}$$

Where $g_{\text{rgb}} = \exp(-\nabla I)$ and ∇I is the gradient of the current aligned RGB image. $|\hat{D}|$ indicates the total number of pixels in $\hat{D}$.

To further enforce smoothness on the rendered depths, we employ a total variation (TV) loss as a penalty:

$$\mathcal{L}_{\hat{D}_{\text{smooth}}} = \frac{1}{|\hat{D}|} \sum_{i,j} (|\hat{D}_{i,j} - \hat{D}_{i+1,j}| + |\hat{D}_{i,j} - \hat{D}_{i,j+1}|) \tag{5}$$

where $\hat{D}_{i,j}$ is the depth value at pixel (i, j).

Regularization with monocular depth: For datasets containing no depth data, we rely on scale aligned monocular depth estimation networks for regularization. We use off-the-shelf monocular depth networks, ZoeDepth [1] and DepthAnything [50] for dense per-pixel depth priors and rely on solving for the scale ambiguity between estimated depths and the scene by comparison with sparse SfM points, similar to prior work [3, 55]. Specifically, for each monocular depth estimate D_{mono} we align the scale to match that of the sparse depth map D_{sparse} obtained by projecting SfM points to the camera view and solving for a per image scale a and shift b parameter using the closed-form linear regression solution to:

$$\hat{a}, \hat{b} = \underset{a,b}{\operatorname{argmin}} \sum_{ij} \|(a * D_{\text{sparse},ij} + b) - D_{\text{mono},ij}\|_2^2 \tag{6}$$

where we denote $D_{\text{sparse},ij}$ and $D_{\text{mono},ij}$ as per-pixel correspondences between the two depth maps. We then apply the same loss as in Eq. (4) for regularization.

4.2 Leveraging Normal Cues

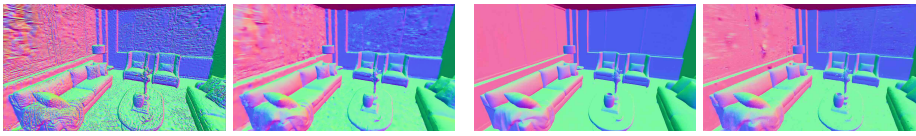
During optimization, we expect Gaussians to become flat, disc-like, with one scaling axis much smaller than the other two. This scaling axis serves as an approximation of a normal direction. Specifically, we define a geometric normal for a Gaussian with a rotation matrix $\mathbf{R} \in SO(3)$, obtained from its quaternion $\mathbf{q}$, and scaling coefficients $\boldsymbol{s} \in \mathbb{R}^3$:

$$\hat{\boldsymbol{n}}_i = \mathbf{R} \cdot \text{OneHot}(\text{argmin}(s_1, s_2, s_3)) \tag{7}$$

where $\text{OneHot}(.) \in \mathbb{R}^3$ returns a unit vector with zeros everywhere except where the scaling is minimum. To ensure correct orientations, we further flip the direction of the normals in the beginning of training in cases where the dot product between the current camera viewing direction and the Gaussian normal is negative. Normals are transformed into camera space with the current camera transform and alpha-composited according to the rendering equation to give a single per-pixel normal estimate:

$$\hat{\boldsymbol{N}} = \sum_{i \in N} \hat{\boldsymbol{n}}_i \alpha_i T_i \tag{8}$$

In contrast to prior research [6, 26], which append an additional learnable parameter per Gaussian for normal prediction, this approach involves deriving normals directly from the geometry. Consequently, during back-propagation, adjustments to Gaussian scale and rotation parameters, *i.e.* covariance matrices, also directly lead to updates in the normal estimates. Therefore, no additional learnable parameters are needed. Intuitively, this also results in Gaussians better conforming to the scene's geometry, as their orientations and scales are compelled to align with the surface normal.



(a) Depth gradient normal **(b)** Monocular normal (Omnidata)

Fig. 2: Depth gradient normal vs. monocular normal supervision. Comparing different design choices for normal supervision using the same regularization at 10k training iterations, we observe that using pseudo normal maps (Fig. 2a, left) derived from the gradient of rendered depths [6] for supervision leads to noisy predicted normals (Fig. 2a, right), especially in challenging scenes. In contrast, we apply a pretrained Omnidata model [5] to acquire normal maps (Fig. 2b, left) that help better capture the geometry of the scene and produce smoother normal estimates (Fig. 2b, right).

Regularization with monocular normals: Gao *et al.* [6] propose using pseudo-ground truth normal maps estimated from the gradient of rendered depths for supervision. However, due to noise in depth maps, especially in complex scenes, this method results in artifacts. Instead, we supervise predicted normals using monocular cues, obtained from Omnidata [5], which provide much smoother estimates for normals. Fig. 2 highlights this difference. We regularise with a L1 loss:

$$\mathcal{L}_{\hat{N}} = \frac{1}{|\hat{N}|} \sum \|\hat{\boldsymbol{N}} - \boldsymbol{N}\|_1 \tag{9}$$

We further apply a prior on the total variation of predicted normals, encouraging smooth normal predictions at neighbouring pixels with:

$$\mathcal{L}_{\hat{N}_{\text{smooth}}} = \frac{1}{|\hat{N}|} \sum_{i,j} \left(|\hat{\boldsymbol{N}}_{i+1,j} - \hat{\boldsymbol{N}}_{i,j}| + |\hat{\boldsymbol{N}}_{i,j+1} - \hat{\boldsymbol{N}}_{i,j}| \right) \tag{10}$$

where $\hat{N}_{ij}$ represents estimated normal values at pixel position (i, j). Thus, our normal regularization loss is defined as follows: $\mathcal{L}_{\text{normal}} = \mathcal{L}_{\hat{N}} + \mathcal{L}_{\hat{N}_{\text{smooth}}}$.

Normal initialization: We initialize Gaussian orientations by estimating normal directions from the initial SfM point cloud using [56], aligning the Gaussian orientations q based on these estimated normals, and setting one of the scaling axis to be smaller than the rest. This initialization helps to speed up convergence.

4.3 Optimization

The final loss we use for optimization is defined as follows:

$$\mathcal{L} = \mathcal{L}_{\text{rgb}} + \lambda_{\text{depth}} \underbrace{(\mathcal{L}_{\hat{D}} + \lambda_{\text{smooth}} \mathcal{L}_{\hat{D}_{\text{smooth}}})}_{\mathcal{L}_{\text{depth}}} + \lambda_{\text{normal}} \underbrace{(\mathcal{L}_{\hat{N}} + \mathcal{L}_{\hat{N}_{\text{smooth}}})}_{\mathcal{L}_{\text{normal}}} \tag{11}$$

Where $\mathcal{L}_{\text{rgb}}$ is the original photometric loss proposed in [16]. We set $\lambda_{\text{depth}} = 0.2$, $\lambda_{\text{smooth}} = 0.5$ and $\lambda_{\text{normal}} = 0.1$ in our experiments.

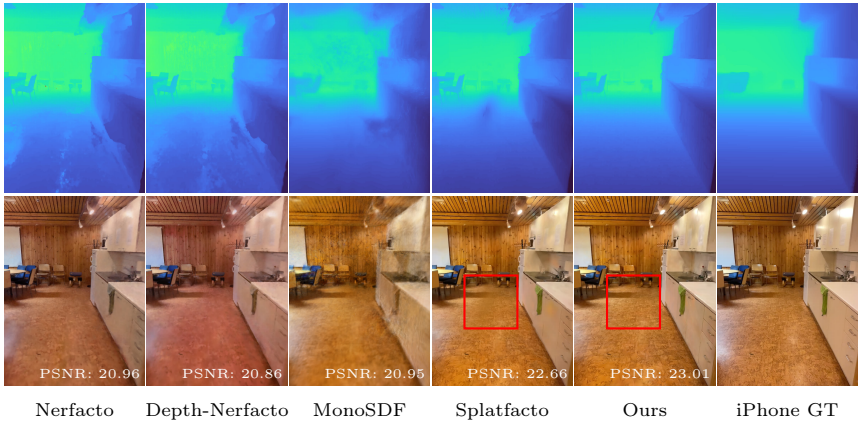


Fig. 3: Qualitative comparison of different baseline methods. Comparison of rendered depths and RGB images on the MuSHRoom dataset.

4.4 Meshing

After optimization with our depth and normal regularization with Eq. (11), we apply Poisson surface reconstruction [15] to extract a mesh. SuGaR [11] proposes to extract a coarse mesh from the scene by projecting rays from camera views and linearly interpolating intersection points based on the value of the local density (Eq. (2)) derived from nearby Gaussians along a ray. However, since scenes contain thousands of Gaussians, the local density along nearby points is non-smooth, resulting in noisy surfaces (see Tab. 6). In addition, in the context of depth regularization, we have already enforced that the positions of the Gaussians are well distributed and aligned along the surface of the scene, thus we directly back-project rendered depth and normal maps from training views to create an oriented point set for meshing. We show qualitative and quantitative differences between different Poisson mesh extraction methods in Tab. 1 and Tab. 6.

5 Experiments

We showcase our regularization strategy on mesh extraction using indoor datasets. We compare against state-of-the-art neural implicit approaches based on NeRFs and SDFs as well as the recent SuGaR method. We also give an extensive ablation study on depth and normal supervision as well as mesh extraction techniques.

Datasets: We focus on indoor datasets and consider the following: a) MuSHRoom [35]: a real-world indoor dataset containing separate training and evaluation camera trajectories; b) ScanNet++ [53]: a real-world indoor dataset with high fidelity 3D geometry and RGB data; c) Replica [41]: smaller real-world indoor scans; d) Tanks & Temples [18]: a real-world high-resolution dataset representing indoor and outdoor scenes.

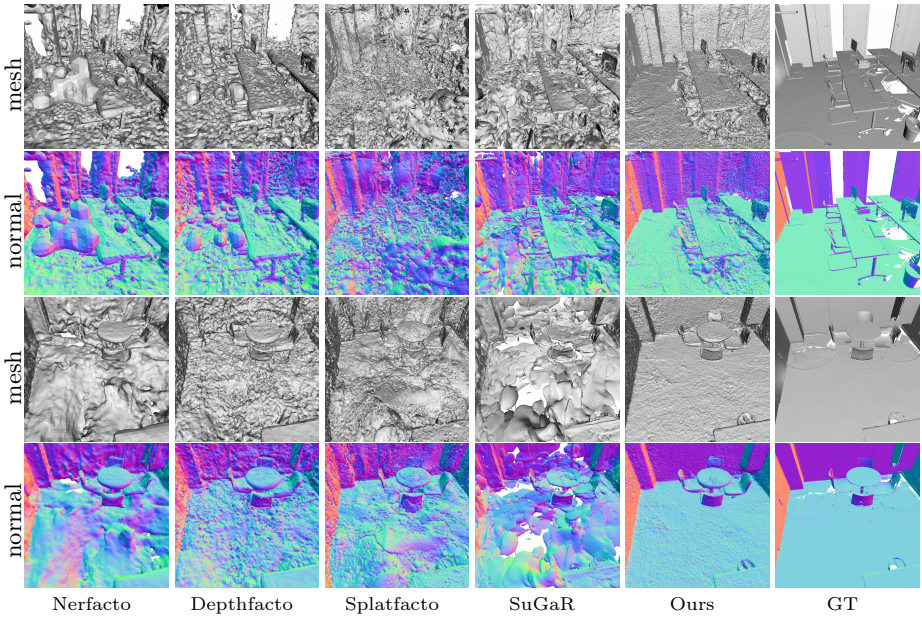


Fig. 4: Reconstruction results on ScanNet++. NeRF variants, even with depth supervision, suffer from artefacts and floaters in reconstruction. The SuGaR method, optimized mainly by photometric losses, struggles to capture the scene geometry in low texture scenes. However, adding depth and normal supervision to 3DGS greatly improves reconstruction performance.

Baselines: We consider a range of baseline methods for comparison a) state-of-the-art NeRF-based method Nerfacto [42] b) its depth regularized version Depth-Nerfacto with a direct loss on ray termination distribution for depth supervision similar to DS-NeRF [4]; c) Neusfacto [54] and MonoSDF [55] for SDF-based implicit surface reconstruction; d) baseline 3DGS Splatfacto method [42], and e) SuGaR [11], a 3DGS variant for mesh reconstruction.

Evaluation metrics: We follow standard practice and report PSNR, SSIM and LPIPS metrics for color images and common depth metrics, similar to [21, 28, 30, 39, 44, 49, 57], to analyze depth quality for datasets containing sensor or ground truth depth data. For mesh evaluation, we follow the evaluation protocol from [35, 46] and report Accuracy, Completion, Chamfer-L_1 distance, Normal Consistency, and F-scores with a threshold of 5cm.

Implementation details: We implement our method in PyTorch [32] and use `gsplat` [52], an open-source re-implementation of the original 3DGS [16] work, which serves as our baseline 3DGS (Splatfacto) method. We train all models for a total of 30k iterations. To obtain monocular normal cues, we propagate RGB images through the pre-trained Omnidata model [5]. For Poisson reconstruction, we extract a total of 2 million points and use a depth level of 9 for all methods, unless otherwise stated. More settings can be seen in the supplementary materials.

Table 1: Mesh evaluation results. We report reconstruction performance on Replica, ScanNet++, and MuSHRoom datasets using Poisson and marching cubes (MC) based baselines. Incorporating depth and normal priors to 3DGS optimization improves mesh reconstruction making it competitive even against SDF based methods. The best results for each category of methods (SDF/NeRF/3DGS) are marked in **bold**.

(a) Reconstruction on Replica/ScanNet++ datasets. We report the average results over 5 and 2 scenes respectively.

	Algorithm	Accuracy ↓	Completion ↓	Chamfer-L_1 ↓	Normal Consistency ↑	F-score ↑
SDF Neusfacto [54]	MC	7.36/32.65	19.45/43.99	13.40/38.32	71.59/51.24	46.05/11.03
SDF MonoSDF [55]	MC	**1.42**/**3.03**	**1.69**/**5.73**	**1.56**/**4.38**	**93.60**/**88.81**	**93.32**/**85.77**
NeRF Nerfacto [42]	Poisson	6.71/13.05	**1.73**/**14.84**	4.22/13.94	81.79/**71.53**	**87.33**/46.98
NeRF Depth-Nerfacto [42]	Poisson	**1.97**/**7.31**	6.30/16.47	**4.13**/**11.89**	**84.79**/68.48	82.18/**50.18**
3DGS Splatfacto [42]	Poisson	3.93/19.34	5.51/15.03	4.72/17.19	83.03/67.41	71.84/17.90
3DGS SuGaR-coarse [11]	Poisson	2.43/9.34	5.37/10.07	3.90/9.71	83.34/71.67	80.18/44.15
3DGS SuGaR-refined [11]	IBR	2.49/9.40	5.40/10.11	3.94/9.75	85.62/72.41	79.39/43.67
3DGS Ours	Poisson	**0.74**/**4.87**	**3.12**/**8.73**	**1.94**/**6.80**	**94.28**/**84.04**	**93.10**/**69.41**

(b) Reconstruction on MuSHRoom datasets. The presented results are an average over 10 scenes.

	Accuracy ↓	Completion ↓	Chamfer-L_1 ↓	Normal Consistency ↑	F-score ↑
Nerfacto [42]	6.52	6.03	6.28	74.91	63.90
Depth-Nerfacto [42]	6.53	6.14	6.34	73.54	61.26
MonoSDF [55]	7.92	**2.37**	5.41	82.00	75.96
Splatfacto [42]	10.74	7.08	8.81	76.02	44.05
Ours	**5.24**	2.86	**4.05**	**83.86**	**80.81**

5.1 Mesh Evaluation

We demonstrate the effectiveness of our depth and normal regularization strategy on the geometry of the scene by extracting meshes directly after optimization, with no additional refinement steps. In Tab. 1a we show how this simple regularization strategy enables competitive mesh extraction on densely captured Replica and ScanNet++ scenes. Improving depth and normal estimation over the baseline 3DGS Splatfacto method makes our method competitive even against the more computationally expensive SDF [54,55] methods. NeRF variants fail to consistently achieve similar results, with depth supervision sometimes hindering mesh performance. Fig. 4 qualitatively compares our mesh extraction with other NeRF and 3DGS baselines on ScanNet++. Tab. 1b uncovers that in larger scale indoor scenes with sparser training views, state-of-the-art SDF-based models relying on marching cubes, like MonoSDF, struggle with reconstruction. This is because MonoSDF faces challenges in optimizing its SDF volume representation from frames with substantial baseline differences. Meticulous rescaling of the input poses as a pre-processing step can help to mitigate this; however, our 3DGS approach succeeds in handling these difficulties even in diverse-sized indoor rooms.

5.2 Depth Estimation and Novel View Synthesis

We show an extensive study on depth and normal supervised 3DGS and its effect on novel view synthesis and depth metrics on challenging real-world scenes from

Table 2: Depth estimation and novel view synthesis on the MuSHRoom dataset. We compare rendered depth and novel view synthesis metrics using two evaluation protocols: Tab. 2a and Tab. 2b. We report results average over 10 scenes.

(a) Test split obtained by sampling uniformly every 10 frames within the training sequence.

Methods	Depth estimation							Novel view synthesis		
	Abs Rel ↓	Sq Rel ↓	RMSE ↓	RMSE log ↓	$\delta < 1.25$ ↑	$\delta < 1.25^2$ ↑	$\delta < 1.25^3$ ↑	PSNR ↑	SSIM ↑	LPIPS ↓
Nerfacto [42]	0.0926	0.0296	0.0802	0.0845	0.9222	0.9702	0.9849	20.83	0.7653	0.2506
Depth-Nerfacto [42]	0.0783	**0.0151**	**0.0615**	0.0734	0.9353	0.9800	0.9917	21.23	0.7623	0.2612
MonoSDF [55]	0.1041	0.0830	0.4170	0.1032	0.8507	0.9495	0.9832	19.79	0.6972	0.4122
Splatfacto (no cues) [16]	0.0826	0.0494	0.3139	0.0956	0.9076	0.9703	0.9874	24.21	0.8375	0.1421
Splatfacto+Only Depth (ours)	**0.0294**	0.0172	0.1893	**0.0300**	0.9774	0.9942	0.9978	**24.40**	**0.8424**	**0.1338**
Splatfacto+Depth+Normal (ours)	0.0295	0.0171	0.1897	**0.0300**	**0.9779**	**0.9944**	**0.9980**	24.35	0.8412	0.1356

(b) Test split obtained from a different camera trajectory with no overlap with the training sequence.

Methods	Depth estimation							Novel view synthesis		
	Abs Rel ↓	Sq Rel ↓	RMSE ↓	RMSE log ↓	$\delta < 1.25$ ↑	$\delta < 1.25^2$ ↑	$\delta < 1.25^3$ ↑	PSNR ↑	SSIM ↑	LPIPS ↓
Nerfacto [42]	0.0910	0.0283	0.0908	0.0818	0.9236	0.9728	0.9855	20.36	0.7448	0.2781
Depth-Nerfacto [42]	0.0750	**0.0141**	**0.0671**	0.0704	0.9410	0.9833	0.9929	20.67	0.7423	0.2873
MonoSDF [55]	0.1053	0.0927	0.4450	0.1046	0.8447	0.9501	0.9835	17.92	0.6683	0.4384
Splatfacto (no cues) [16]	0.0832	0.0563	0.3412	0.1007	0.9134	0.9668	0.9814	21.39	0.7738	0.1986
Splatfacto+Only Depth (ours)	**0.0350**	0.0221	0.2241	**0.0354**	0.9714	0.9930	0.9982	**21.60**	0.7799	0.1889
Splatfacto+Depth+Normal (ours)	0.0354	0.0220	0.2249	0.0356	**0.9717**	**0.9934**	**0.9984**	21.59	**0.7805**	**0.1854**

Table 3: Depth estimation and novel view synthesis on ScanNet++ dataset.

Methods	Depth estimation							Novel view synthesis		
	Abs Rel ↓	Sq Rel ↓	RMSE ↓	RMSE log ↓	$\delta < 1.25$ ↑	$\delta < 1.25^2$ ↑	$\delta < 1.25^3$ ↑	PSNR ↑	SSIM ↑	LPIPS ↓
Nerfacto [42]	0.4780	0.5331	0.3787	0.2628	0.7554	0.7913	0.8307	19.41	0.8776	0.1529
Depth-Nerfacto [42]	0.0698	**0.0133**	0.0815	0.0651	0.9573	0.9834	0.9924	19.52	0.8766	0.1561
MonoSDF [55]	0.1523	0.0137	**0.0450**	0.2331	0.5351	0.9332	0.9967	17.41	0.8253	0.2561
Splatfacto (no cues) [16]	0.1481	0.1345	0.5122	0.2192	0.7602	0.9103	0.9572	23.41	0.9111	0.1316
Splatfacto+Only Depth (ours)	**0.0287**	0.0164	0.1794	**0.0276**	0.9781	0.9933	0.9976	**23.62**	**0.9121**	**0.1276**
Splatfacto+Depth+Normal (ours)	0.0289	0.0162	0.1793	0.0278	**0.9785**	**0.9935**	**0.9977**	23.18	0.9049	0.1311

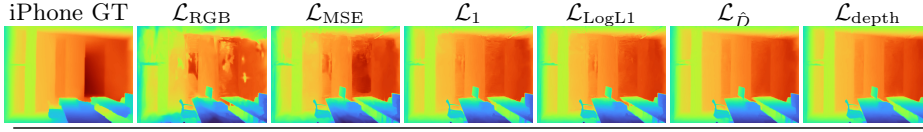
MuSHRoom (Tab. 2) and ScanNet++ (Tab. 3) datasets with iPhone depth data. Tab. 2 and Tab. 3 demonstrate that incorporating sensor depth supervision into 3DGS enhances both depth and RGB metrics in comparison to scenarios without depth supervision. Fig. 3 qualitatively highlights the improvement in floaters and artifacts in depth and RGB renders using our approach over other baselines.

5.3 Ablation Studies

We analyze the impact of different loss functions on depth estimation and novel-view synthesis. Furthermore, we ablate various design choices for normal estimation and mesh extraction.

Depth losses: We analyze the choice of our proposed depth loss on the ScanNet++ dataset and provide qualitative and quantitative analysis in Tab. 4. We compare common depth losses: $\mathcal{L}_{\text{MSE}}$, $\mathcal{L}_1$, $\mathcal{L}_{\text{LogL1}}$ [12] and their effect on depth and novel view synthesis metrics. For more comparisons and definitions, including ablations on $\mathcal{L}_{\text{DSSIML1}}$ [7, 48], $\mathcal{L}_{\text{HuberL1}}$ [22, 23], see the supplementary material. We observe that $\mathcal{L}_{\text{LogL1}}$ and $\mathcal{L}_1$ losses generally perform the best on depth and color metrics, with the logarithmic variant giving smoother qualitative reconstructions. In addition, we observe that adding smoothing regularization with Eq. (5) on rendered depth maps significantly improves qualitative results.

Table 4: Ablation of depth losses on ScanNet++. We get the best results with a logarithmic depth penalty and a smoothing prior. Please zoom in to see the details.



Methods	Depth estimation							Novel view synthesis		
	Abs Rel ↓	Sq Rel ↓	RMSE ↓	RMSE log ↓	$\delta < 1.25$ ↑	$\delta < 1.25^2$ ↑	$\delta < 1.25^3$ ↑	PSNR ↑	SSIM ↑	LPIPS ↓
Splatfacto (no cues)	0.1481	0.1345	0.5122	0.2192	0.7602	0.9103	0.9572	23.41	0.9111	0.1316
Splatfacto+$\mathcal{L}_{\text{MSE}}$	0.0551	0.0180	0.2104	0.0569	0.9647	**0.9940**	**0.9985**	**23.84**	**0.9140**	0.1246
Splatfacto+$\mathcal{L}_{1}$	0.0351	0.0170	0.1889	0.0346	0.9758	0.9932	0.9977	23.74	0.9138	0.1235
Splatfacto+$\mathcal{L}_{\text{LogL1}}$	0.0364	0.0180	0.1955	0.0361	0.9744	0.9925	0.9975	23.77	0.9139	**0.1234**
Splatfacto+$\mathcal{L}_{\hat{D}}$	**0.0285**	0.0162	0.1790	**0.0275**	**0.9782**	0.9934	0.9977	23.61	0.9122	0.1269
Splatfacto+$\mathcal{L}_{\text{depth}}$	0.0287	0.0164	0.1794	0.0276	0.9781	0.9933	0.9976	23.62	0.9121	0.1276

Table 5: Monocular depth supervision with dense and sparse views. We report novel view synthesis results on the advanced "Courtroom" scene from Tanks & Temples [18] and show that monocular depth supervision improves rendering quality.

Methods	Novel view synthesis														
	Dense views			Sparse views											
				1/3 training views			1/5 training views			1/8 training views			1/12 training views		
	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓
Splatfacto (no cues)	22.86	0.7941	0.1817	20.68	0.7445	0.1921	18.50	0.6991	0.2110	16.86	0.6459	0.2474	14.76	0.5580	0.3332
+ZoeDepth [1]	**23.02**	**0.7990**	**0.1649**	20.88	0.7518	0.1833	19.58	0.7118	0.2007	**17.60**	**0.6568**	**0.2433**	15.90	0.5835	0.2971
+DepthAnything [50]	22.93	0.7969	0.1684	**20.91**	**0.7528**	**0.1830**	**19.60**	**0.7153**	**0.1997**	17.44	**0.6568**	0.2456	**16.24**	**0.5902**	**0.2924**

Table 6: Poisson mesh extraction techniques on Replica. We provide qualitative and quantitative comparison of Poisson-based mesh extraction techniques. The proposed approach achieves better reconstruction results by leveraging geometric cues.

	Gaussians	Density 0.1 [11]	Density 0.3 [11]	Density 0.5 [11]	Ours
Accuracy ↓	2.06	1.30	0.91	0.83	**0.74**
Completion ↓	4.12	3.57	3.13	**3.04**	3.12
Chamfer-L_1 ↓	3.09	2.43	2.02	1.94	**1.94**
Normal Consistency↑	90.91	93.01	93.35	93.09	**94.28**
F-score↑	91.17	92.75	**93.31**	93.25	93.10

Monocular depth supervision: To further study the advantage of depth regularized 3DGS on RGB-only datasets, we report RGB metrics on the advanced "Courtroom" scene from Tanks & Temples [18] using monocular depth supervision. We choose recent state-of-the-art monocular depth estimation networks ZoeDepth [1] and DepthAnything [50] for ablation. Monocular depth estimates are aligned with sparse SfM point clouds obtained from COLMAP [37] as explained in Sec. 4.1. Tab. 5 shows that monocular depth supervision improves RGB reconstruction results with dense and sparse input views.

Normal supervision and losses: We investigate the quality of our normal supervision strategy using the synthetic Replica [41] dataset which contains ground truth normal maps for comparison. In Tab. 7 we compare rendered nor-

mal quality supervised by either the gradient of depth maps (∇D-from [13]) or monocular normals estimated from a pre-trained Omnidata model [5]. We report the mean-angular-error (MAE) and root-mean-square-error (RMSE) errors compared with ground truth normal maps. We compare against a baseline model with no supervision, depth gradient supervision [13] with and without our depth loss Eq. (4), and our monocular normal supervision ($\mathcal{L}_{\text{normal}}$) strategy. Normal supervision with monocular normals significantly outperforms the more noise prone depth gradient method and the baseline. We also observe a significant improvement when using the smoothing prior proposed in $\mathcal{L}_{\text{normal}}$ which enforces nearby Gaussians to have similar orientations. Furthermore, we investigate the impact of normal supervision on the resulting mesh extraction in Tab. 7. For all experiments, we enable our depth loss for a fair comparison and use our mesh extraction strategy. We show that improving Gaussian normal estimation has a direct impact on the generated mesh.

Table 7: Ablation on normal supervision strategies on Replica.

	Normal estimation		Reconstruction	
	MAE (°) ↓	RMSE ↓	Chamfer-L_1 ↓	F-score ↑
3DGS (no normal cues) [16]	29.17	0.342	2.10	92.52
∇D-based $\mathcal{L}_{\hat{N}}$+Depth Cues	13.74	0.203	2.06	92.82
Monocular normal: $\mathcal{L}_{\hat{N}}$	<u>5.89</u>	<u>0.099</u>	**1.97**	**93.29**
Monocular normal: $\mathcal{L}_{\text{normal}}$	**5.06**	**0.085**	<u>1.99</u>	**93.29**

Mesh extraction techniques: Lastly, we investigate various oriented point sets to Poisson extraction techniques. In Tab. 6 we show that extracting oriented point sets from optimized depth and normal maps results in smoother and more realistic reconstructions. We report mesh evaluation metrics for the different methods in Tab. 6. We compare with directly using trained Gaussian means and normals (total of 512k Gaussians), extraction of surface density level 0.1 and level 0.5 as proposed in SuGaR [11], and back-projection of optimized depth and normal maps (Ours). All models were trained with our depth and normal regularization and we set the total number of extracted points to 500k for the surface density and backprojection methods for a fair comparison.

6 Conclusion

We presented DN-Splatter, a method for depth and normal regularization of 3D Gaussian splatting. We showed how this simple yet effective strategy can enhance photorealism by improving common novel-view RGB metrics as well as significantly improving the depth estimation and surface quality extracted from the Gaussian scene representation. We showed how prior regularization is necessary to achieve more geometrically valid reconstructions on challenging indoor scenes. Although we improved over the baseline methods, we limited our focus on densely captured scenes with relatively still cameras. Future work could focus on more difficult and sparser data captures suffering from motion blur and other capture artefacts. In addition, better meshing techniques for concurrently optimizing the Gaussian scene parameters and mesh quality are needed. Poisson surface reconstruction is more sensitive to estimated positions of the points than their normal estimates, thus further smoothing and regularizing the Gaussian scene representation would result in smoother reconstructions. This is left as future work.

References

1. Bhat, S.F., Birkl, R., Wofk, D., Wonka, P., Müller, M.: Zoedepth: Zero-shot transfer by combining relative and metric depth (2023). https://doi.org/10.48550/ARXIV.2302.12288, https://arxiv.org/abs/2302.12288
2. Chen, H., Li, C., Lee, G.H.: Neusg: Neural implicit surface reconstruction with 3d gaussian splatting guidance (2023)
3. Chung, J., Oh, J., Lee, K.M.: Depth-regularized optimization for 3d gaussian splatting in few-shot images. arXiv preprint arXiv:2311.13398 (2023)
4. Deng, K., Liu, A., Zhu, J.Y., Ramanan, D.: Depth-supervised NeRF: Fewer views and faster training for free. In: CVPR (June 2022)
5. Eftekhar, A., Sax, A., Malik, J., Zamir, A.: Omnidata: A scalable pipeline for making multi-task mid-level vision datasets from 3d scans. In: ICCV. pp. 10786–10796 (2021)
6. Gao, J., Gu, C., Lin, Y., Zhu, H., Cao, X., Zhang, L., Yao, Y.: Relightable 3d gaussian: Real-time point cloud relighting with brdf decomposition and ray tracing. arXiv:2311.16043 (2023)
7. Godard, C., Mac Aodha, O., Brostow, G.J.: Unsupervised monocular depth estimation with left-right consistency. In: CVPR. pp. 270–279 (2017)
8. Goesele, M., Snavely, N., Curless, B., Hoppe, H., Seitz, S.M.: Multi-view stereo for community photo collections. In: ICCV. pp. 1–8 (2007). https://doi.org/10.1109/ICCV.2007.4408933
9. Guangcong, Chen, Z., Loy, C.C., Liu, Z.: Sparsenerf: Distilling depth ranking for few-shot novel view synthesis. ICCV (2023)
10. Guo, H., Peng, S., Lin, H., Wang, Q., Zhang, G., Bao, H., Zhou, X.: Neural 3d scene reconstruction with the manhattan-world assumption. In: CVPR (2022)
11. Guédon, A., Lepetit, V.: Sugar: Surface-aligned gaussian splatting for efficient 3d mesh reconstruction and high-quality mesh rendering (2023)
12. Hu, J., Ozay, M., Zhang, Y., Okatani, T.: Revisiting single image depth estimation: Toward higher resolution maps with accurate object boundaries. In: 2019 IEEE Winter Conference on Applications of Computer Vision (WACV) (Dec 2018). https://doi.org/10.1109/wacv.2019.00116, http://dx.doi.org/10.1109/wacv.2019.00116
13. Jiang, Y., Tu, J., Liu, Y., Gao, X., Long, X., Wang, W., Ma, Y.: Gaussianshader: 3d gaussian splatting with shading functions for reflective surfaces (2023)
14. Kajiya, J.T.: The rendering equation. In: Proceedings of the 13th Annual Conference on Computer Graphics and Interactive Techniques. p. 143–150. SIGGRAPH '86, Association for Computing Machinery, New York, NY, USA (1986). https://doi.org/10.1145/15922.15902, https://doi.org/10.1145/15922.15902
15. Kazhdan, M., Bolitho, M., Hoppe, H.: Poisson Surface Reconstruction. In: Sheffer, A., Polthier, K. (eds.) Symposium on Geometry Processing. The Eurographics Association (2006). https://doi.org/10.2312/SGP/SGP06/061-070
16. Kerbl, B., Kopanas, G., Leimkühler, T., Drettakis, G.: 3d gaussian splatting for real-time radiance field rendering. ACM TOG **42**(4) (2023)
17. Keselman, L., Hebert, M.: Flexible techniques for differentiable rendering with 3d gaussians. In: ICCV (2023)
18. Knapitsch, A., Park, J., Zhou, Q.Y., Koltun, V.: Tanks and temples: Benchmarking large-scale scene reconstruction. ACM TOG **36**(4) (2017)
19. Kopanas, G., Philip, J., Leimkühler, T., Drettakis, G.: Point-based neural rendering with per-view optimization. Computer Graphics Forum (Proceedings of

the Eurographics Symposium on Rendering) **40**(4) (June 2021), http://www-sop.inria.fr/reves/Basilic/2021/KPLD21
20. Kosheleva, E., Jaiswal, S., Shamsafar, F., Cheema, N., Illgner-Fehns, K., Slusallek, P.: Edge-aware consistent stereo video depth estimation. arXiv preprint arXiv:2305.02645 (2023)
21. Kusupati, U., Cheng, S., Chen, R., Su, H.: Normal assisted stereo depth estimation. In: CVPR. pp. 2189–2199 (2020)
22. Kuznietsov, Y., Stuckler, J., Leibe, B.: Semi-supervised deep learning for monocular depth map prediction. In: CVPR (Jun 2017). https://doi.org/10.1109/cvpr.2017.238, http://dx.doi.org/10.1109/cvpr.2017.238
23. Laina, I., Rupprecht, C., Belagiannis, V., Tombari, F., Navab, N.: Deeper depth prediction with fully convolutional residual networks. In: 2016 Fourth International Conference on 3D Vision (3DV) (Sep 2016). https://doi.org/10.1109/3dv.2016.32, http://dx.doi.org/10.1109/3dv.2016.32
24. Lao, Y., Xu, X., Cai, Z., Liu, X., Zhao, H.: CorresNeRF: Image correspondence priors for neural radiance fields. In: NeurIPS (2023)
25. Li, Z., Müller, T., Evans, A., Taylor, R.H., Unberath, M., Liu, M.Y., Lin, C.H.: Neuralangelo: High-fidelity neural surface reconstruction. In: CVPR (2023)
26. Liang, Z., Zhang, Q., Feng, Y., Shan, Y., Jia, K.: Gs-ir: 3d gaussian splatting for inverse rendering (2023)
27. Lorensen, W.E., Cline, H.E.: Marching cubes: A high resolution 3d surface construction algorithm. In: Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques. p. 163–169. SIGGRAPH '87, Association for Computing Machinery, New York, NY, USA (1987). https://doi.org/10.1145/37401.37422, https://doi.org/10.1145/37401.37422
28. Luo, X., Huang, J.B., Szeliski, R., Matzen, K., Kopf, J.: Consistent video depth estimation. ACM TOG **39**(4), 71–1 (2020)
29. Mildenhall, B., Srinivasan, P.P., Tancik, M., Barron, J.T., Ramamoorthi, R., Ng, R.: Nerf: Representing scenes as neural radiance fields for view synthesis. In: ECCV (2020)
30. Murez, Z., Van As, T., Bartolozzi, J., Sinha, A., Badrinarayanan, V., Rabinovich, A.: Atlas: End-to-end 3d scene reconstruction from posed images. In: ECCV. pp. 414–431. Springer (2020)
31. Niemeyer, M., Barron, J.T., Mildenhall, B., Sajjadi, M.S.M., Geiger, A., Radwan, N.: Regnerf: Regularizing neural radiance fields for view synthesis from sparse inputs. In: CVPR (2021)
32. Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., Chintala, S.: Pytorch: An imperative style, high-performance deep learning library. In: NeurIPS, pp. 8024–8035. Curran Associates, Inc. (2019), http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf
33. Radl, L., Steiner, M., Parger, M., Weinrauch, A., Kerbl, B., Steinberger, M.: Stopthepop: Sorted gaussian splatting for view-consistent real-time rendering (2024)
34. Rakotosaona, M.J., Manhardt, F., Arroyo, D.M., Niemeyer, M., Kundu, A., Tombari, F.: Nerfmeshing: Distilling neural radiance fields into geometrically-accurate 3d meshes. In: Proc. of the International Conf. on 3D Vision (3DV) (2024)

35. Ren, X., Wang, W., Cai, D., Tuominen, T., Kannala, J., Rahtu, E.: Mushroom: Multi-sensor hybrid room dataset for joint 3d reconstruction and novel view synthesis. In: Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision. pp. 4508–4517 (2024)
36. Rückert, D., Franke, L., Stamminger, M.: Adop: Approximate differentiable one-pixel point rendering. ACM TOG **41**(4), 1–14 (2022)
37. Schönberger, J.L., Frahm, J.M.: Structure-from-motion revisited. In: CVPR (2016)
38. Schönberger, J.L., Zheng, E., Pollefeys, M., Frahm, J.M.: Pixelwise view selection for unstructured multi-view stereo. In: ECCV (2016)
39. Sinha, A., Murez, Z., Bartolozzi, J., Badrinarayanan, V., Rabinovich, A.: Deltas: Depth estimation by learning triangulation and densification of sparse points. In: ECCV. pp. 104–121. Springer (2020)
40. Snavely, N., Seitz, S.M., Szeliski, R.: Photo tourism: exploring photo collections in 3d. In: ACM siggraph 2006 papers, pp. 835–846. Association for Computing Machinery (ACM) (2006)
41. Straub, J., Whelan, T., Ma, L., Chen, Y., Wijmans, E., Green, S., Engel, J.J., Mur-Artal, R., Ren, C., Verma, S., Clarkson, A., Yan, M., Budge, B., Yan, Y., Pan, X., Yon, J., Zou, Y., Leon, K., Carter, N., Briales, J., Gillingham, T., Mueggler, E., Pesqueira, L., Savva, M., Batra, D., Strasdat, H.M., Nardi, R.D., Goesele, M., Lovegrove, S., Newcombe, R.: The Replica dataset: A digital replica of indoor spaces. arXiv preprint arXiv:1906.05797 (2019)
42. Tancik, M., Weber, E., Ng, E., Li, R., Yi, B., Kerr, J., Wang, T., Kristoffersen, A., Austin, J., Salahi, K., Ahuja, A., McAllister, D., Kanazawa, A.: Nerfstudio: A modular framework for neural radiance field development. In: ACM SIGGRAPH 2023 Conference Proceedings. SIGGRAPH '23 (2023)
43. Tang, J., Zhou, H., Chen, X., Hu, T., Ding, E., Wang, J., Zeng, G.: Delicate textured mesh recovery from nerf via adaptive surface refinement. arXiv preprint arXiv:2303.02091 (2022)
44. Teed, Z., Deng, J.: Deepv2d: Video to depth with differentiable structure from motion. arXiv preprint arXiv:1812.04605 (2018)
45. Verbin, D., Hedman, P., Mildenhall, B., Zickler, T., Barron, J.T., Srinivasan, P.P.: Ref-NeRF: Structured view-dependent appearance for neural radiance fields. CVPR (2022)
46. Wang, J., Bleja, T., Agapito, L.: Go-surf: Neural feature grid optimization for fast, high-fidelity rgb-d surface reconstruction. In: 2022 International Conference on 3D Vision (3DV). pp. 433–442. IEEE (2022)
47. Wang, P., Liu, L., Liu, Y., Theobalt, C., Komura, T., Wang, W.: Neus: Learning neural implicit surfaces by volume rendering for multi-view reconstruction. arXiv preprint arXiv:2106.10689 (2021)
48. Watson, J., Firman, M., Brostow, G.J., Turmukhambetov, D.: Self-supervised monocular depth hints. In: ICCV. pp. 2162–2171 (2019)
49. Wei, Y., Liu, S., Rao, Y., Zhao, W., Lu, J., Zhou, J.: Nerfingmvs: Guided optimization of neural radiance fields for indoor multi-view stereo. In: ICCV. pp. 5610–5619 (2021)
50. Yang, L., Kang, B., Huang, Z., Xu, X., Feng, J., Zhao, H.: Depth anything: Unleashing the power of large-scale unlabeled data. arXiv:2401.10891 (2024)
51. Yariv, L., Gu, J., Kasten, Y., Lipman, Y.: Volume rendering of neural implicit surfaces. In: NeurIPS (2021)
52. Ye, V., Turkulainen, M., the Nerfstudio team: gsplat, `https://github.com/nerfstudio-project/gsplat`

53. Yeshwanth, C., Liu, Y.C., Nießner, M., Dai, A.: Scannet++: A high-fidelity dataset of 3d indoor scenes. In: ICCV (2023)
54. Yu, Z., Chen, A., Antic, B., Peng, S., Bhattacharyya, A., Niemeyer, M., Tang, S., Sattler, T., Geiger, A.: Sdfstudio: A unified framework for surface reconstruction (2022), `https://github.com/autonomousvision/sdfstudio`
55. Yu, Z., Peng, S., Niemeyer, M., Sattler, T., Geiger, A.: Monosdf: Exploring monocular geometric cues for neural implicit surface reconstruction. NeurIPS (2022)
56. Zhou, Q.Y., Park, J., Koltun, V.: Open3D: A modern library for 3D data processing. arXiv:1801.09847 (2018)
57. Zhou, T., Brown, M., Snavely, N., Lowe, D.G.: Unsupervised learning of depth and ego-motion from video. In: CVPR. pp. 1851–1858 (2017)

Supplementary Material for DN-Splatter: Depth and Normal Priors for Gaussian Splatting and Meshing

Matias Turkulainen*[1], Xuqian Ren*[2], Iaroslav Melekhov[3], Otto Seiskari[4], Esa Rahtu[2,4], and Juho Kannala[3,4]

[1] ETH Zurich, [2] Tampere University, [3] Aalto University, [4] Spectacular AI
Corresponding author: matiasturkulainen@gmail.com

In this **supplementary material**, we provide further implementation details in Section 1 and report additional quantitative and qualitative results in Section 2. Finally, we provide additional ablation studies of sensor depth losses and monocular depth supervision in Section 3.

1 Implementation Details

In this section, we first present an overview of the baseline methods and the datasets evaluated in this work in Section 1.1 and Section 1.2, respectively. Next, we describe additional details regarding depth evaluation metrics in Section 1.3 and discuss depth objective functions in Section 1.4.

1.1 Baselines

Nerfacto. We use the Nerfacto model from Nerfstudio [42] version 1.0.2 in our experiments. We predict normals using the proposed method from Ref-NeRF [45] and use rendered normal and depth maps for Poisson surface reconstruction.
Depth-Nerfacto. We use the depth supervised variant of Nerfacto with a direct loss on ray termination distribution for sensor depth supervision as described in DS-NeRF [4]. Besides this, we use the same settings as for Nerfacto.
Neusfacto. We use default settings provided by Neus-facto from SDFStudio [54].
MonoSDF. We use a modified version of the Neus-facto model with additional regularization for sensor depths and monocular normal estimates. We use the recommended settings from MonoSDF [55] and set the sensor depth loss multiplier to 0.1 and normal loss multiplier to 0.05.
Splatfacto. The Splatfacto model from Nerfstudio version 1.0.2 and `gsplat` [52] version 0.1.7 serve as our baseline 3DGS model. This is a faithful re-implementation of the original 3DGS work [16]. We keep all the default settings for the baseline comparison.
SuGaR. We use the official SuGaR [11] source-code. The original code-base, written as an extension to the original 3DGS work [16], supports only COLMAP based datasets (that is, datasets containing a COLMAP database file). We made slight modifications to the original source-code to support non-COLMAP based

formats to import camera information and poses directly from a pre-made .json files. We use default settings for training as described in [11]. We use the SDF trained variant in all experiments. We extract both the coarse and refined meshes for evaluation, although the difference in geometry metrics are small between them.

1.2 Datasets

MuSHRoom. We use the official train and evaluation splits from the MuSHRoom dataset. We report evaluation metrics on a) images obtained from uniformly sampling every 10 frames from the training camera trajectory and b) images obtained from a different camera trajecotry. We use the globally optimized COLMAP [37] for both evaluation sequences. We use a total of 5 million points for mesh extraction for Poisson surface reconstruction.

ScanNet++. We use the "b20a261fdf" and "8b5caf3398" scenes in our experiments. We use the iPhone sequences with COLMAP registered poses. The sequences contain 358 and 705 registered images respectively. We uniformly load every 5th frame from the sequences from which we reserve every 10th frame for evaluation.

Replica. We use the room0, room1, room2, office0, and office1 sequences from the Replica dataset. Each sequence has 2000 densely captured images. We uniformly load every 25th frame from the trajectories from which we reserve every 5th frame for evaluation. This results in 64 training and 16 evaluation images for all sequences.

1.3 Depth Evaluation Metrics

For the ScanNet++ and MuSHRoom datasets, we follow [21,28,30,39,44,49,57] and report the depth estimation quality with: Absolute Relative Distance (*Abs Rel*); Squared Relative Distance (*Sq Rel*); *RMSE*; *RMSE log*; *Threshold accuracy* ($\delta < t$). The *Abs Rel* metric provides a measure of the average magnitude of the relative error between the predicted depth values and the ground truth depth values. Unlike the *Abs Rel* metric, the *Sq Rel* considers the squared relative error between the predicted and ground truth depth values. The *RMSE* metric calculates the square root of the average of the squared differences between the predicted and the ground-truth values. It gives a measure of the magnitude of the error made by the predictions. The *RMSE log* metric is similar to *RMSE* but applied in the logarithmic domain, which can be particularly useful for very large depth values. The *Threshold accuracy* measures the percentage of predicted depth values within a certain threshold factor, δ of the ground-truth depth values. These metrics are defined in Tab. 1

Metric	Definition
Absolute Relative Distance (Abs Rel)	$\frac{1}{N}\sum_{i=1}^{N}\frac{\lvert d_i^{\text{pred}}-d_i^{\text{gt}}\rvert}{d_i^{\text{gt}}}$
Squared Relative Distance (Sq Rel)	$\frac{1}{N}\sum_{i=1}^{N}\frac{(d_i^{\text{pred}}-d_i^{\text{gt}})^2}{d_i^{\text{gt}}}$
RMSE	$\sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(d_i^{\text{pred}}-d_i^{\text{gt}}\right)^2}$
RMSE log	$\sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(\log d_i^{\text{pred}}-\log d_i^{\text{gt}}\right)^2}$
Threshold accuracy, δ	$\frac{1}{N}\sum_{i=1}^{N}\left[\max\left(\frac{d_i^{\text{pred}}}{d_i^{\text{gt}}},\frac{d_i^{\text{gt}}}{d_i^{\text{pred}}}\right)<\delta\right]$

Table 1: Depth Evaluation Metrics. We show definitions for our depth evaluation metrics. d_i^{pred} and d_i^{gt} are predicted and ground-truth depths for the i-th pixel. δ is the threshold factor (*e.g.* $\delta < 1.25$, $\delta < 1.25^2$, $\delta < 1.25^3$).

1.4 Depth Losses

For depth supervision, we compare the following variants of loss functions:

$$\mathcal{L}_{\text{MSE}} = \frac{1}{|\hat{D}|}\sum(\hat{D}-D)^2 \tag{1}$$

$$\mathcal{L}_{1} = \frac{1}{|\hat{D}|}\sum\|\hat{D}-D\|_1 \tag{2}$$

$$\mathcal{L}_{\text{LogL1}} = \frac{1}{|\hat{D}|}\sum\log(1+\|\hat{D}-D\|_1) \tag{3}$$

$$\mathcal{L}_{\text{HuberL1}} = \begin{cases}\|D-\hat{D}\|_1, & \text{if } \|D-\hat{D}\|_1 \le \delta,\\ \frac{(D-\hat{D})^2+\delta^2}{2\delta}, & \text{otherwise.}\end{cases} \tag{4}$$

$$\mathcal{L}_{\text{DSSIML1}} = \alpha\frac{1-\text{SSIM}(I,\hat{I})}{2}+(1-\alpha)|I-\hat{I}| \tag{5}$$

$$\mathcal{L}_{\text{EAS}} = g_{\text{rgb}}\frac{1}{|\hat{D}|}\sum\|\hat{D}-D\|_1 \tag{6}$$

$$\mathcal{L}_{\hat{D}} = g_{\text{rgb}}\frac{1}{|\hat{D}|}\sum\log(1+\|\hat{D}-D\|_1) \tag{7}$$

$$\mathcal{L}_{\hat{D}_{\text{smooth}}} = \frac{1}{|\hat{D}|}\sum_{i,j}(|\hat{D}_{i,j}-\hat{D}_{i+1,j}|+|\hat{D}_{i,j}-\hat{D}_{i,j+1}|) \tag{8}$$

$$\mathcal{L}_{\text{depth}} = \mathcal{L}_{\hat{D}}+\mathcal{L}_{\hat{D}_{\text{smooth}}} \tag{9}$$

Where $\delta = 0.2\max(\|D-\hat{D}\|_1)$, $g_{\text{rgb}} = \exp(-\nabla I)$, $D/\hat{D}$ are the ground truth and rendered depths, and $I/\hat{I}$ is the ground truth/rendered rgb image. We compare the performance of these losses as supervision in Tab. 3.

2 Additional Quantitative and Qualitative Results

We provide additional quantitative results for depth and normal supervised 3DGS. In Table 2 we show depth estimation performance when compared to Faro laser scanner ground truth depth maps instead of iPhone depth captures for the MuSHRoom dataset. This corresponds to Table 2 in the main paper, which compares depth metrics on iPhone depth captures for the same scenes and baselines. When comparing to laser scanner depths, our method still out performs other baselines on depth estimation.

Table 2: Depth evaluation metrics compared to ground truth Faro scanner depths for the MuSHRoom dataset. Instead of evaluating on noisy captured iPhone depth maps, we rely on more accurate depths reconstructed from a laser scanner. We show that our depth regularization strategy greatly outperforms other baselines. Results are averaged over 10 scenes.

Methods	(a) Test within a sequence							(b) Test with a different sequence						
	Abs Rel ↓	Sq Rel ↓	RMSE ↓	RMSE log ↓	$\delta < 1.25 \uparrow$	$\delta < 1.25^2 \uparrow$	$\delta < 1.25^3 \uparrow$	Abs Rel ↓	Sq Rel ↓	RMSE ↓	RMSE log ↓	$\delta < 1.25 \uparrow$	$\delta < 1.25^2 \uparrow$	$\delta < 1.25^3 \uparrow$
Nerfacto [42]	0.1472	0.1979	0.6105	0.1326	0.8825	0.9648	0.9829	0.1452	0.1832	0.6385	0.1313	0.8841	0.9665	0.9844
Depth-Nerfacto [42]	0.1390	0.1171	0.5021	0.1298	0.8846	0.9722	0.9871	0.1349	0.1076	0.5163	0.1262	0.8923	0.9759	0.9897
MonoSDF [55]	0.1090	0.0987	0.4874	0.1127	0.8348	0.9381	0.9766	0.1100	0.1098	0.5092	0.1137	0.8262	0.9380	0.9772
Splatfacto (no cues) [16]	0.0832	0.0545	0.3847	0.1023	0.8975	0.9578	0.9790	0.0806	0.0539	0.3861	0.1005	0.9051	0.9592	0.9783
Splatfacto+Only Depth (ours)	0.0371	0.0308	0.3080	0.0427	0.9552	0.9782	0.9891	0.0378	0.0308	0.3135	0.0426	0.9547	0.9803	0.9914
Splatfacto+Depth+Normal (ours)	0.0364	0.0302	0.3033	0.0417	0.9560	0.9783	0.9893	0.0369	0.0297	0.3057	0.0415	0.9564	0.9807	0.9916

In Fig. 1 we visualize mesh reconstruction results on challenging scenes from the MuSHRoom dataset. The qualitative results indicate that our simple depth and normal regularization strategy is competitive against non-3DGS based baselines. Adding depth and normal supervision to 3DGS greatly enhances geometry reconstruction. We also additionaly visualize various depth and RGB frames from sequences from the MuSHRoom dataset in Fig. 2, Fig. 3 and Fig. 4. We consistently generate more realistic depth maps and mitigate artifacts in RGB frames compared to the baseline 3DGS method.

3 Additional Ablation studies

3.1 Ablations on Sensor Depth Losses

We compare the performance of various depth losses described in Section 1.4 in Table 3. There are several interesting observations, firstly that the logarithmic depth loss L_{LogL1} outperforms other popular variants like L_{L1} or L_{MSE} on depth and RGB synthesis. Also, the gradient aware logarithmic depth variant $L_{\hat{D}}$ outperforms the simpler variant. This validates our assumption that captured sensor depths, like those from iPhone cameras, tend to contain noise and inaccuracies at edges or sharp boundaries. Therefore, the gradient aware variant mitigates these inaccurate sensor readings. Furthermore, we visualize various depth losses in Fig. 5. Adding an additional smoothing prior on rendered depths with $L_{\hat{D}_{\text{smooth}}}$ improves qualitative results on flat surfaces.

Table 3: Ablation study on different sensor depth losses on MuSHRoom iPhone sequences with two evaluation methods: (a) test set obtained by sampling uniformly within the training sequence and (b) test set obtained from a wholly different camera trajectory. Results are averaged over 2 scenes ("activity", "kokko"). We observe that a logarithmic penalty combined with a smoothing prior performs the best in depth estimation and novel view syntheis metrics, validating our choice of depth supervision using $\mathcal{L}_{\text{depth}}$.

(a) Test split obtained by sampling uniformly every 10 frames within the training sequence.

Methods	Depth estimation							Novel view synthesis		
	Abs Rel ↓	Sq Rel ↓	RMSE ↓	RMSE log ↓	$\delta < 1.25$ ↑	$\delta < 1.25^2$ ↑	$\delta < 1.25^3$ ↑	PSNR ↑	SSIM ↑	LPIPS ↓
$\mathcal{L}_{\text{MSE}}$	0.0587	0.0229	0.2313	0.0618	0.9534	0.9900	0.9966	22.32	0.7995	0.1653
$\mathcal{L}_1$	0.0419	0.0233	0.2286	0.0435	0.9629	0.9909	0.9965	22.46	0.8041	0.1594
$\mathcal{L}_{\text{DSSIML1}}$	0.0476	0.0331	0.2773	0.0523	0.9476	0.9821	0.9929	21.77	0.7802	0.1879
$\mathcal{L}_{\text{LogL1}}$	0.0430	0.0267	0.2414	0.0444	0.9609	0.9898	0.9961	22.48	0.8053	0.1580
$\mathcal{L}_{\text{HuberL1}}$	0.0536	0.0239	0.2335	0.0561	0.9579	0.9903	0.9962	22.39	0.8017	0.1625
$\mathcal{L}_{\text{EAS}}$	0.0954	0.0572	0.3581	0.1103	0.8726	0.9528	0.9784	22.18	0.7951	0.1780
$\mathcal{L}_{\hat{D}_{\text{smooth}}}$	0.0993	0.0614	0.3680	0.1163	0.8667	0.9445	0.9704	21.97	0.7897	0.1859
$\mathcal{L}_{\text{LogL1}} + \mathcal{L}_{\text{EAS}}$	0.0444	0.0268	0.2444	0.0463	0.9591	0.9892	0.9958	22.47	0.8039	0.1614
$\mathcal{L}_{\text{LogL1}} + \mathcal{L}_{\hat{D}_{\text{smooth}}}$	0.0444	0.0269	0.2449	0.0467	0.9575	0.9882	0.9956	22.40	0.8015	0.1646
$\mathcal{L}_{\hat{D}}$	0.0338	0.0212	0.2170	0.0350	0.9691	0.9922	0.9968	22.49	0.8031	0.1630
$\mathcal{L}_{\hat{D}} + \mathcal{L}_{\text{EAS}}$	0.0340	0.0205	0.2148	0.0355	0.9683	0.9916	0.9966	22.34	0.8006	0.1673
$\mathcal{L}_{\text{depth}}$	0.0340	0.0213	0.2175	0.0353	0.9688	0.9920	0.9967	22.48	0.8025	0.1640

(b) Test split obtained from a different camera trajectory with no overlap with the training sequence.

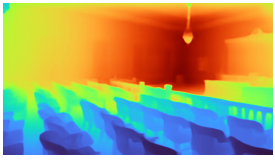
Methods	Depth estimation							Novel view synthesis		
	Abs Rel ↓	Sq Rel ↓	RMSE ↓	RMSE log ↓	$\delta < 1.25$ ↑	$\delta < 1.25^2$ ↑	$\delta < 1.25^3$ ↑	PSNR ↑	SSIM ↑	LPIPS ↓
$\mathcal{L}_{\text{MSE}}$	0.0572	0.0282	0.2506	0.0570	0.9585	0.9878	0.9964	19.37	0.7088	0.2329
$\mathcal{L}_1$	0.0449	0.0248	0.2364	0.0449	0.9639	0.9888	0.9965	19.45	0.7164	0.2253
$\mathcal{L}_{\text{DSSIML1}}$	0.0482	0.0330	0.2775	0.0527	0.9495	0.9791	0.9915	18.98	0.7040	0.2430
$\mathcal{L}_{\text{LogL1}}$	0.0451	0.0269	0.2454	0.0453	0.9629	0.9881	0.9961	19.50	0.7183	0.2228
$\mathcal{L}_{\text{HuberL1}}$	0.0526	0.0267	0.2483	0.0533	0.9617	0.9881	0.9962	19.45	0.7128	0.2285
$\mathcal{L}_{\text{EAS}}$	0.0724	0.0442	0.3142	0.0819	0.9329	0.9709	0.9852	19.30	0.7108	0.2351
$\mathcal{L}_{\hat{D}_{\text{smooth}}}$	0.0761	0.0496	0.3229	0.0895	0.9272	0.9685	0.9804	19.22	0.7097	0.2429
$\mathcal{L}_{\text{LogL1}} + \mathcal{L}_{\text{EAS}}$	0.0447	0.0256	0.2437	0.0456	0.9625	0.9878	0.9958	19.48	0.7185	0.2223
$\mathcal{L}_{\text{LogL1}} + \mathcal{L}_{\hat{D}_{\text{smooth}}}$	0.0437	0.0249	0.2413	0.0445	0.9641	0.9886	0.9962	19.48	0.7193	0.2261
$\mathcal{L}_{\hat{D}}$	0.0427	0.0252	0.2335	0.0420	0.9632	0.9888	0.9967	19.53	0.7187	0.2286
$\mathcal{L}_{\hat{D}} + \mathcal{L}_{\text{EAS}}$	0.0424	0.0252	0.2383	0.0427	0.9614	0.9880	0.9964	19.45	0.7207	0.2283
$\mathcal{L}_{\text{depth}}$	0.0421	0.0243	0.2324	0.0420	0.9631	0.9891	0.9967	19.51	0.7202	0.2270

3.2 Ablation on Monocular Depth Supervision

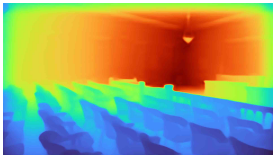
We also investigate monocular depth supervision with the Zoe [1] and DepthAnything [50] monocular depth estimation models in Table 4 with the "Courtroom" and "Ballroom" sequences from Tanks & Temples dataset. The Depth Anything model generally performs better.

Table 4: Comparison with monocular depth from Zoe-Depth and DepthAnything.

(a) Visualization of predicted depth



Predicted Depth from Zoe-Depth [1]



Predicted Depth from Depth Anything [50]

(b) Comparison of different depth prediction networks and depth loss weight λ_{depth}. Results are averaged over 2 scenes.

Depth Prediction Network	Depth loss weight	PSNR ↑	SSIM ↑	LPIPS ↓
-	Ours (RGB)	23.29	0.8029	0.1456
Depth Anything	$\lambda_{\text{depth}} = 0.01$	**23.42**	0.8048	0.1403
	$\lambda_{\text{depth}} = 0.1$	23.11	0.8008	0.1442
Zoe Depth	$\lambda_{\text{depth}} = 0.01$	23.38	**0.8049**	0.1393
	$\lambda_{\text{depth}} = 0.05$	23.21	0.8042	**0.1391**
	$\lambda_{\text{depth}} = 0.1$	23.04	0.8012	0.1431

Fig. 1: Qualitative comparison on mesh reconstruction. Comparison of baseline methods on sequences from the MuSHRoom dataset.

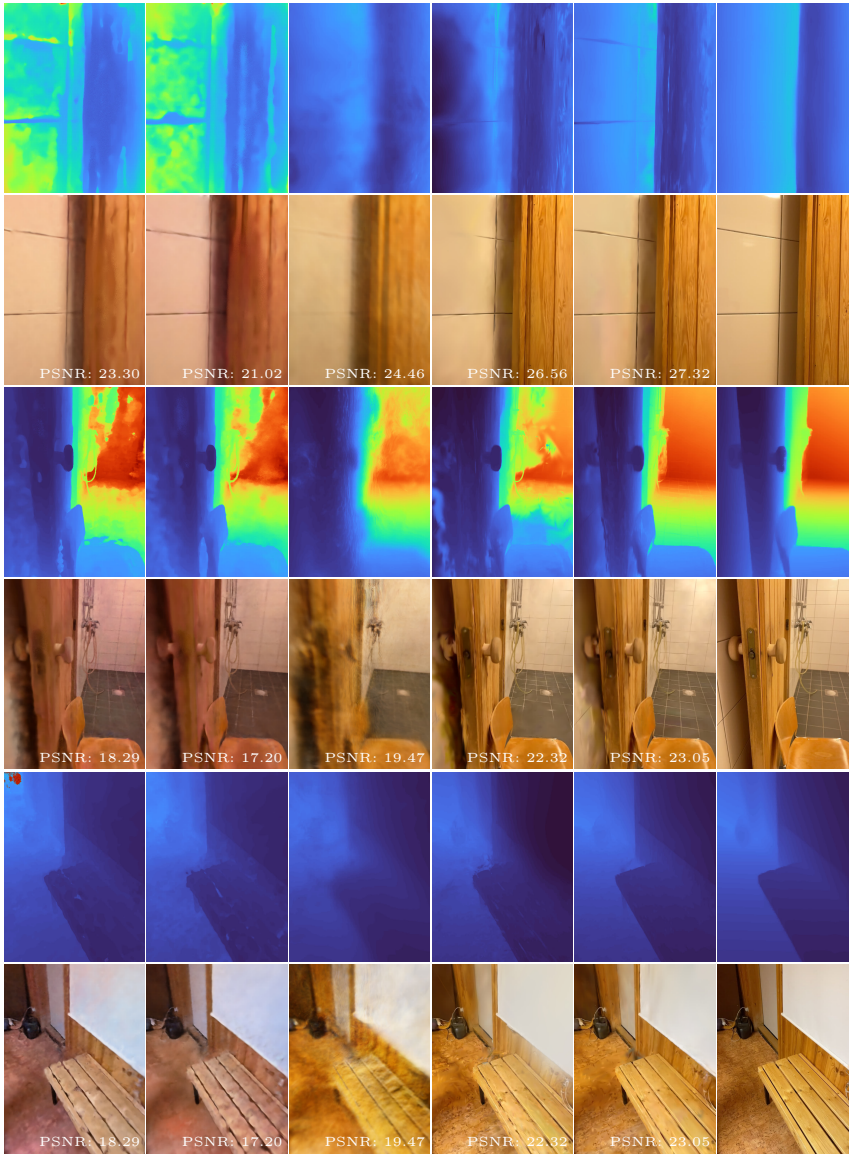


Fig. 2: Qualitative comparison of rendered depth and RGB images. Comparison of baseline methods on the "sauna" sequence from the MuSHRoom dataset.

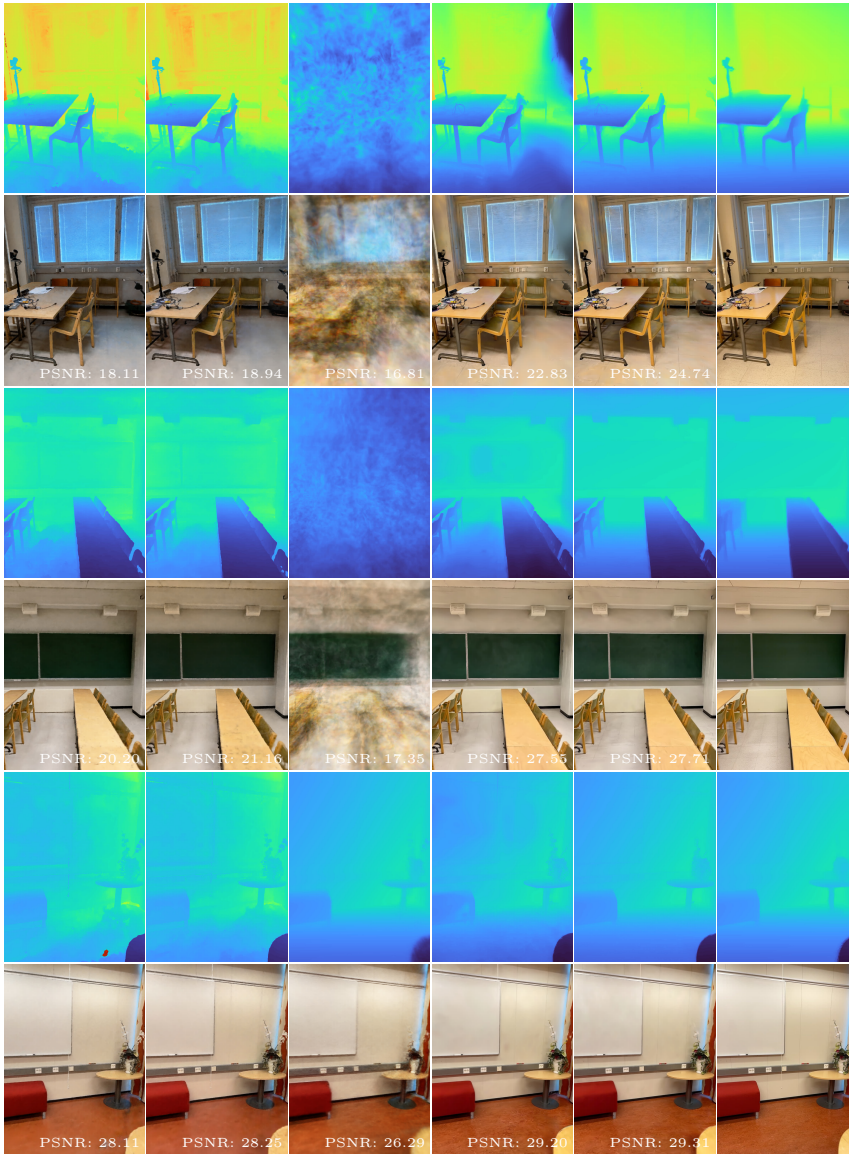


Fig. 3: Qualitative comparison of rendered depth and RGB images. Comparison of baseline methods on the "classroom" and "coffee room" sequences from the MuSHRoom dataset.

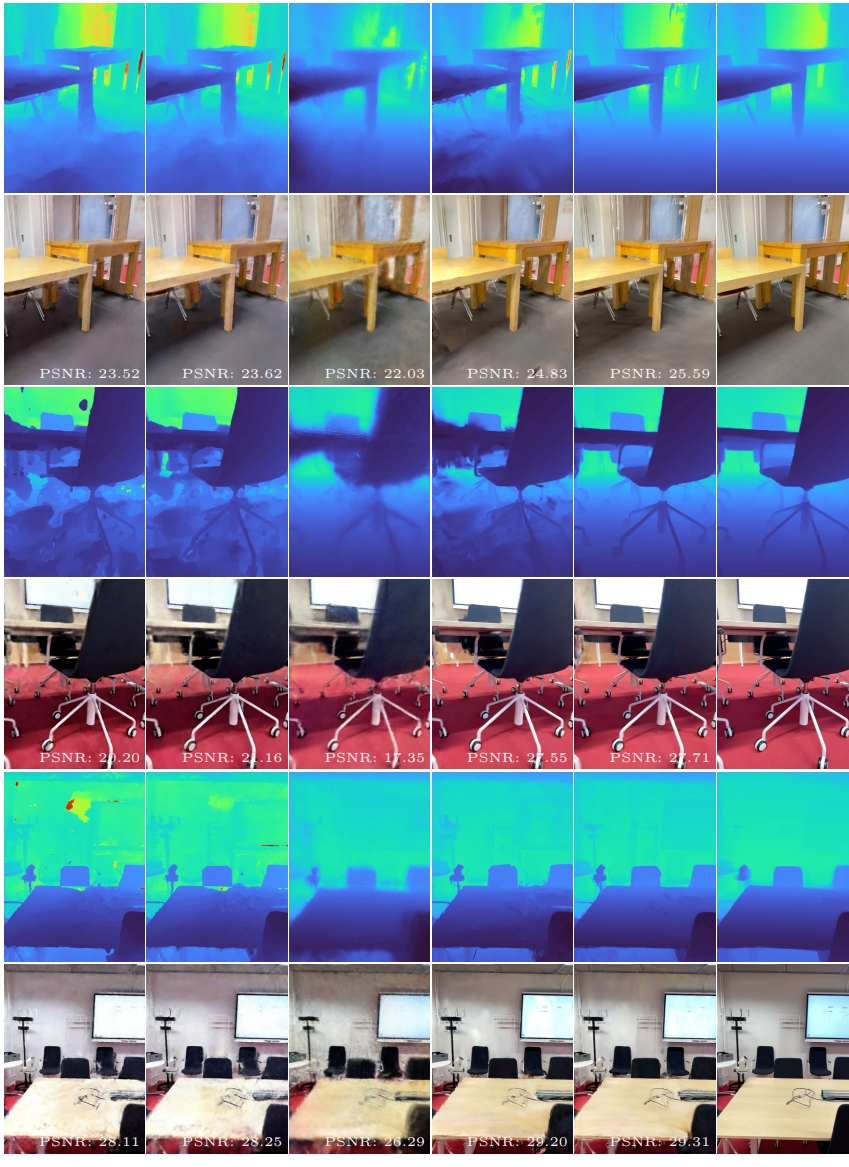


Fig. 4: Qualitative comparison of rendered depth and RGB images. Comparison of baseline methods on the "koivu" sequence from the MuSHRoom dataset.

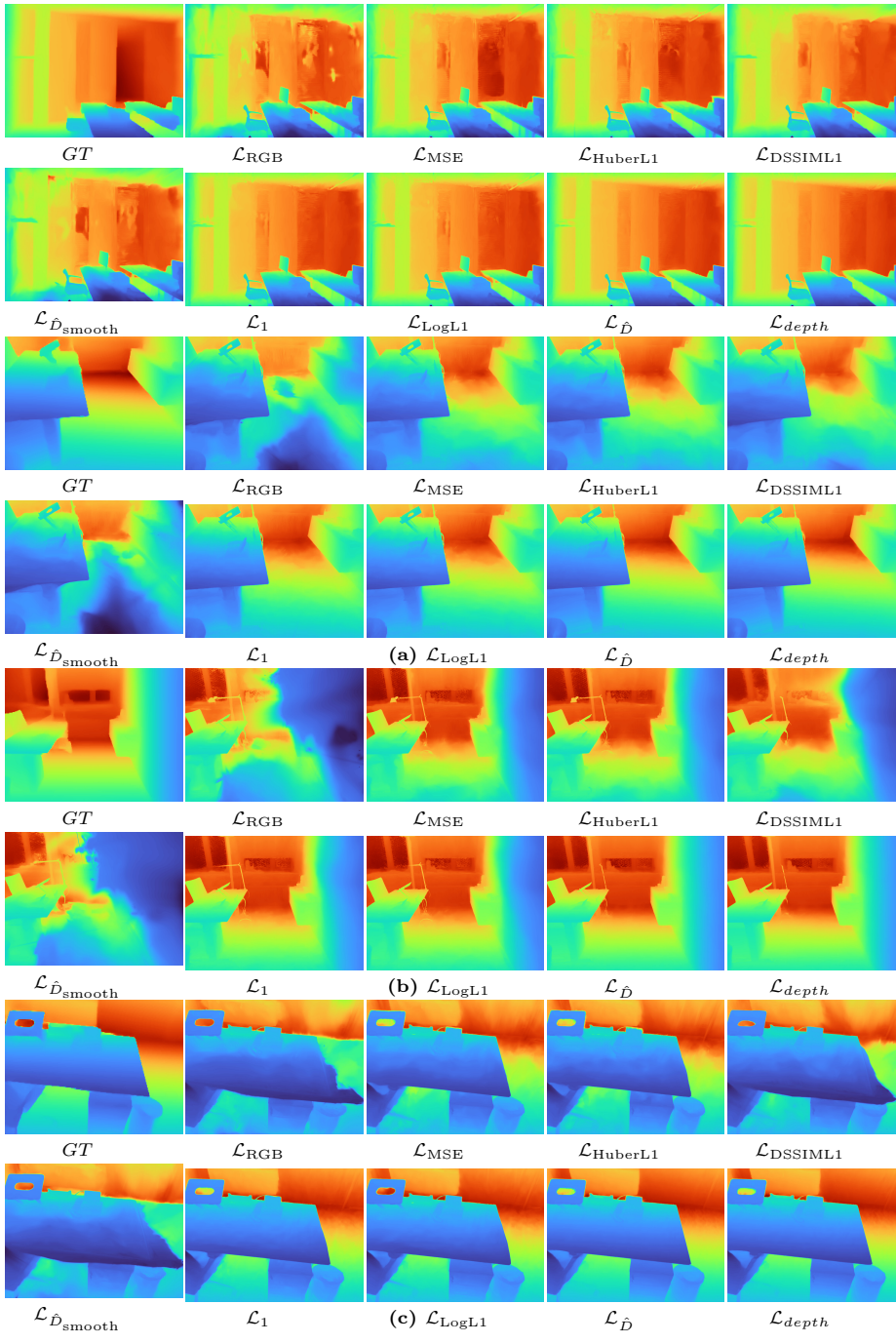


Fig. 5: Ablation on sensor depth losses. Qualitative comparison of rendered depths supervised with different depth losses on ScanNet++. *GT* indicates the iPhone capture depth. Our proposed logarithmic and smoothed loss $\mathcal{L}_{\text{depth}}$ qualitatively gives the best depth reconstruction results.