DN-LinPrim: Depth- and Normals-regularized LinPrim

Karl Freund Miguel Trasobares
{karl.freund, miguel.trasobares}@tum.de
Technical University of Munich
Arcisstraße 21, 80333 Munich

Abstract

3D Gaussian Splatting (3DGS) has established a new standard for fast and efficient Novel View Synthesis. Building upon it, LinPrim introduced a way to incorporate geometrically bounded primitives in the form of polyhedra, leading to comparable results with 3DGS while requiring fewer primitives. However, just as 3DGS, LinPrim solely relies on photometric supervision, leading to visual reconstructions that are high quality but lack geometric accuracy. We extend LinPrim by introducing dense geometric priors in the form of depth- and normal-regularization. We show that our method outperforms LinPrim in geometric accuracy while reaching parity with other geometrically-regularized 3DGS approaches and maintaining comparable photometric results to all.[1]

1. Introduction

The field of novel view synthesis has been revolutionized by *NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis* (NeRFs) [6] and, more recently, *3D Gaussian Splatting for Real-Time Radiance Field Rendering* (3DGS) [4]. While 3DGS allows for real-time rendering via splatting, the spherical or ellipsoidal nature of Gaussians can struggle to represent sharp and planar structures found in structural environments. *LinPrim: Linear Primitives for Differentiable Volumetric Rendering* (LinPrim) [11] addresses this by introducing linear pirmitives, oriented octahedra defined by a geometric center and 3 axis-aligned distances as a replacement for Gaussians. With the primitives being volumetrically bounded, this representation of the environment is better suited for reconstructing structural geometry. However, like 3DGS, LinPrim relies on photometric loss (L1 and SSIM) during optimization. Although accurate RGB renderings can be generated from incorrect geometry, it can lead to "floater" artifacts and noisy surfaces when viewed from unseen angles, as well as imperfect geometrical placement of primitives.

In this work, we address these limitations by integrating geometric regularization in the form of explicit depth and surface normal priors into the LinPrim optimization pipeline. Our contributions are:

1. We implement a pipeline which generates dense depth and surface normals priors from RGB images.
2. Geometric Regularization: We add depth and normal regularizaiton to LinPrim's existing optimization process to enforce geometric reconstruction accuracy.
3. We demonstrate that our method can produce high-quality 3D reconstructions with accurate geometry and surface normals in a real-world dataset.

2. Related Work

Explicit Volumetric Rendering 3D Gaussian Splatting [4] represented a paradigm shift from MLP-based Neural Radiance Fields [6] away from implicit representations encoded in continuous functions to explicit geometric primitives. *LinPrim: Linear Primitives for Differentiable Volumetric Rendering* (LinPrim) [11] extends this by utilizing volumetrically bounded polyhedra as primitives, in contrast to unbounded 3D Gaussian distributions in 3DGS.

Depth Priors in View Synthesis Pure RGB supervision is often insufficient for clean geometry. Previous works have integrated depth priors into NeRFs by predicting depth maps directly from sparse point clouds gained from Structure from Motion algorithms [8]. For 3DGS, *Depth-Regularized Optimization for 3D Gaussian Splatting in Few-Shot Images* [1] has utilized monocular depth prediction, while *DN-Splatter: Depth and Normal Priors for Gaussian Splatting and Meshing* (DN-Splatter) [10] consider depth maps obtained from sensors to regularize the position and scale of Gaussians.

Surface Normal Priors in View Synthesis Complementary to depth, surface normals provide geometrical signals

[1] Website: https://adl4cv-dn-linprim.github.io/dn-linprim.github.io/

to constrain local surface orientation, especially in textureless regions where photometric signals are weak. Recent advances in explicit rendering have integrated normal supervision to improve geometric reconstruction, as well as photometric quality [1, 8]. *Normal-GS: 3D Gaussian Splatting with Normal-Involved Rendering* [12] utilizes self-regularization by comparing normals gathered throughout the rendering process against normals derived from a rendered depth map. Similarly, DN-Splatter [10] performs employs monocular normal estimation and normals computation from depth maps from sensor data as priors.

3. Method

As in LinPrim [11], our method optimizes the features of a set of linear primitives (octahedra) from a set of RGB images I_i with known camera intrinsics K. As preprocessing, images are undistorted, while the COLMAP [9] Structure-from-Motion (SfM) algorithm provides the camera extrinsics (R_i, t_i) per image and a sparse point cloud $P \in R^{n\times 3}$ as a byproduct. First, we employ these 3D points P and their metric depth p_z at their 2D projections $(u, v) = \Pi(p, K)$ onto each image plane to get a sparse depth map D_{sparse} per image. This sparse depth map is used as a reference to obtain a dense depth prior D^*_{dense}. Secondly, we employ these depth maps to compute the normal prior N_i. During optimization, the depth and normals of the primitives are rendered via α-blending onto the image plane and compared to the priors, resulting in additional loss terms.

3.1. Geometric Signal Computation

We employ a state-of-the-art monocular depth network, *Depth Anything V2* (DAv2) [13], as an estimator to predict a dense relative depth map per view D_{dense} that is scale-and-shift invariant. In order to guide the optimization, the depth signal must match the scene in metric scale. To resolve scale ambiguity, the estimated relative depth maps D_{dense} are scale-and-shift aligned via weighted Least-squares Minimization at the projected point positions to obtain metric scaled depth maps $^*_{\text{dense}}$,

$$D^*_{\text{dense}} = s^* \cdot D_{\text{dense}} + t^*, \tag{1}$$

with optimal scale s^* and shift t^* derived from

$$\arg\min_{s,t} \sum_{p\in D_{\text{sparse}}} \|w(p)\,(D_{\text{sparse}}(p) - D_{\text{dense}}(p; s, t))\|_2^2, \tag{2}$$

where $D_{\text{dense}}(p; s, t)$ is the scaled and shifted bilinear interpolated depth value at the projected position of p within the image plane and $w \in (0, 1)$ is the confidence assigned to each pixel's depth via linear error-based scaling $w_p = 1 - (\epsilon_p/\epsilon_{max})$ to avoid the influence of outliers, with ϵ_p being the reprojection error provided by COLMAP for point p.

3.2. Surface Normals Signal Estimation

For each training image I_i, surface normals are computed from the aligned dense depth map D^*_{dense}. Each pixel position $p(u, v)$ is unprojected to 3D $p \in \mathbb{R}^3$ using its aligned depth,

$$\mathbf{p}(u, v) = \Pi^{-1}\big((u, v), D^*_{\text{dense}}(u, v); K\big). \tag{3}$$

3D local tangent vectors are approximated via forward finite differences:

$$\mathbf{t}_u = \nabla_u \mathbf{p}, \quad \mathbf{t}_v = \nabla_v \mathbf{p}. \tag{4}$$

Finally, the per-pixel surface normal is obtained as the normalized cross product,

$$\mathrm{N}^*(u, v) = \frac{\mathbf{t}_u \times \mathbf{t}_v}{\|\mathbf{t}_u \times \mathbf{t}_v\|}. \tag{5}$$

3.3. Rendering Process

LinPrim [11] utilizes a rasterization pipeline to render each pixel's color (RGB) via α-blending the linear primitives along the ray based on their individual density ρ_i. We use the same differentiable point-based rendering technique as LinPrim to render the depth D and normal vector N per pixel, along with the opacity α_i and transmittance T_i computation per ray-intersected primitive:

$$\alpha_i = 1 - e^{-\rho_i \Delta z_i}, \tag{6}$$

$$T_i = \prod_{j=1}^{i-1} (1 - \alpha_j), \tag{7}$$

where Δz_i is the distance along the ray inside the primitve. We consider the intersection 3D point reported by the *Möller-Trumbore intersection algorithm (MTIA)* [7] closest to the image plane as the entry point p^{inter} of the ray $r(t) = o + t \cdot d$ into the primitive.

Depth Rendering For every primitive intersected by the ray r, the depth $d_i = p_z^{inter}$ at the intersection point is accumulated as:

$$D = \sum_{i=1}^{N} T_i \alpha_i d_i, \tag{8}$$

Surface Normal Rendering Similarly, for every primitive intersected by the ray r, the normal n_i at the intersection point p^{inter} is computed as the normal of the triangle with vertices (v_0, v_1, v_2) that p^{inter} lies on:

$$t_u = v_1 - v_0, \quad t_v = v_2 - v_0, \quad n_i = \frac{t_u \times t_v}{\|t_u \times t_v\|_2}. \tag{9}$$

To ensure consistency with the viewing direction, the normal is flipped to face the camera. The normal at the surface is accumulated as:

$$N = \frac{\sum_{i=1}^{N} T_i \alpha_i n_i}{\| \sum_{i=1}^{N} T_i \alpha_i n_i \|_2} . \tag{10}$$

3.4. Geometric Optimization

In order to optimize the linear primitives' features we introduce per-pixel depth and normal losses. The depth loss $\mathcal{L}_D$ encourages the rasterized depths D to match the aligned depths D^*_{dense} through the L_1 norm's mean over all pixels:

$$\mathcal{L}_D = \frac{1}{|\mathcal{M}|} \sum_{p \in \mathcal{M}} \| D(p) - D^*_{\text{dense}}(p) \|_1, \tag{11}$$

where $\mathcal{M}$ is the set of all valid pixels with $D^*_{\text{dense}}(p) > 0$. Similarly, the surface normal loss $\mathcal{L}_{\text{N}}$ measures the average cosine similarity between the rendered normals normals N and the estimated normals N^* over all pixels:

$$\mathcal{L}_N = 1 - \frac{1}{|\mathcal{I}|} \sum_{p \in \mathcal{I}} \frac{N(p) \cdot N^*(p)}{\|N(p)\|_2 \cdot \|N^*(p)\|_2}, \tag{12}$$

where $\mathcal{I}$ is the set of all pixels (u, v) in the normal map. The final training objective is comprised of the loss terms used in 3DGS [4] and LinPrim [11], as well as the depth and normal losses:

$$\mathcal{L} = (1 - \lambda)\mathcal{L}_1 + \lambda \mathcal{L}_{\text{D-SSIM}} + \lambda_{\text{D}} \mathcal{L}_{\text{D}} + \lambda_{\text{N}} \mathcal{L}_{\text{N}}. \tag{13}$$

We set $\lambda_D = 0.1, \lambda_N = 0.05$ in all our experiments. Details can be found in 6.2.

4. Experiments

Dataset Our results are evaluated on the first 5 scenes from the ScanNet++ [14] dataset which include test views. The datasets focuses on indoor scenes with high-quality RGB views, as well as sparse 3D point clouds from COLMAP's [9] Structure from Motion algorithm. It offers predefined train and test data splits which we use for all experiments.

Baselines To evaluate our method against comparable state-of-the-art methods, we consider DN-Splatter [10] for utilizing depth- and normal-regularization, as well as the LinPrim base implementation. All methods are trained on complete scenes with full resolution RGB views at 30k iterations. Hyperparameters are chosen as they are found to be optimal in the aforementioned papers, namely $\lambda_d = 0.2, \lambda_n = 0.1, \lambda_s = 0.1$ for DN-Splatter, $\lambda = 0.2$ for LinPrim and third degree spherical harmonics for all methods. For our hyperparameters refer to 3.4. To achieve comparable results for depth regularization, instead of considering DN-Splatter's standard depth priors, we supply the aligned depth priors D^*_{dense} as in our model.

Evaluation Metrics We evaluate our method using standard metrics for both rendering and geometry. Photometric quality is measured via PSNR, SSIM, and LPIPS. Geometric quality is assessed using Absolute Relative Error (AbsRel) and Threshold Accuracy (δ_1) for depth, alongside Mean Angular Error (MAE) for surface normals.

4.1. Experiment Results

Table 1 summarizes quantitative results for complete scenes and 10-view subsets (see 6.1). Our method consistently outperforms both baselines across all metrics, demonstrating the effectiveness of our depth and normal guidance while not decreasing the image quality. Compared to LinPrim, our method recovers sharper and much more detailed geometric details while achieving slightly better reconstruction quality, while DN-Splatter tends to oversmooth the geometric details (see Figure 1). Nevertheless, despite the regularization, our method shows artifacts at flat surfaces of the scenes (walls, chairs, etc.), where the linear primitives protrude towards the camera.

4.2. Ablation Study

In order to understand the impact of our proposed components, we evaluate several variants of our method in scene. We first explore two strategies for depth accumulation in the rendering, and then analyze the contribution of each loss term to the overall reconstruction quality.

Depth Guidance We compare our proposed α-blended depth at the ray-primitive intersection with a simpler naive alternative that accumulates the depth at the geometric centers of the primitives, see Table 4. The naive loss variant performs better in terms of RGB reconstruction, while the depth at intersection variant performs better at geometric metrics.

Loss Contributions We study the effect of each loss term by evaluating a mixture of losses: depth loss $\mathcal{L}_{\text{D}}$, normal loss $\mathcal{L}_{\text{N}}$ and self-regularization loss $\mathcal{L}_{\text{N self reg}}$. At the self regularization we consider the normal signal $N^*(p)$ to be the normal map computed from the current rendered depth map D. The results are summarized in Table 5, we show that self regularization $\mathcal{L}_{\text{N self reg}}$ performs slightly better at RGB metrics while $\mathcal{L}_{\text{N}}$ performs at the surface normal metrics.

5. Conclusion

We introduced DN-LinPrim, a geometrically regularized extension of LinPrim, utilizing depth and normal priors derived from monocular RGB views. We demonstrate that it establishes geometric reconstruction improvements while delivering comparable photometric quality for real-world

Setting	Method	RGB Reconstruction			Depth Accuracy		Normals
		PSNR ↑	SSIM ↑	LPIPS ↓	AbsRel ↓	δ_1 ↑	MAE ↓
All Views	LinPrim	24.09	0.849	0.281	0.121	0.806	31.12°
	DN-Splatter	23.46	0.834	0.334	0.073	0.919	16.80°
	Ours	**24.28**	**0.853**	**0.275**	**0.056**	**0.959**	**15.73°**
10 Views	LinPrim	16.88	0.742	0.414	0.299	0.412	40.72°
	DN-Splatter	16.54	0.739	0.431	0.304	0.471	29.22°
	Ours	**17.94**	**0.759**	**0.379**	**0.118**	**0.850**	**22.26°**

Table 1. Mean results on complete scenes and 10 views from 5 ScanNet++ scenes. Bold indicates best performance. Optimized on 30k iterations at full image resolution. Refer to 6.1 for more details.

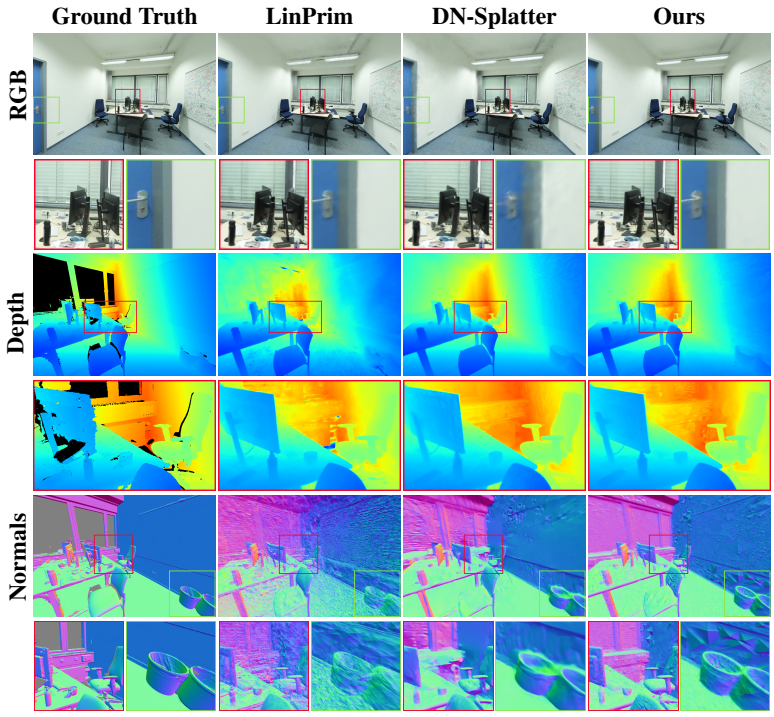

Figure 1. We compare RGB reconstruction (Top), Depth estimation (Middle), and Surface Normals (Bottom). Our method (right column) recovers sharper geometric details compared to LinPrim and displays more accurate photometric quality than DN-Splatter.

indoor scenes. Future work may consider extracting accurate mesh-based representations leveraging the accurate rendered depth and normals via Poisson surface reconstruction [3]. Additional improvements which consider the mesh quality during optimization such as done in *MILo: Mesh-In-the-Loop Gaussian Splatting for Detailed and Efficient Surface Reconstruction* [2] or which consider simpler primitives with constant density like *EVER: Exact Volumetric Ellipsoid Rendering for Real-time View Synthesis* [5] may also yield improved results in the future.

References

[1] Jaeyoung Chung, Jeongtaek Oh, and Kyoung Mu Lee. Depth-regularized optimization for 3d gaussian splatting in few-shot images, 2024. 1, 2

[2] Antoine Guédon, Diego Gomez, Nissim Maruani, Bingchen Gong, George Drettakis, and Maks Ovsjanikov. Milo: Mesh-in-the-loop gaussian splatting for detailed and efficient surface reconstruction, 2025. 4

[3] Michael Kazhdan, Matthew Bolitho, and Hugues Hoppe. Poisson surface reconstruction. In *Proceedings of the fourth Eurographics symposium on Geometry processing*, 2006. 4

[4] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering, 2023. 1, 3

[5] Alexander Mai, Peter Hedman, George Kopanas, Dor Verbin, David Futschik, Qiangeng Xu, Falko Kuester, Jonathan T. Barron, and Yinda Zhang. Ever: Exact volumetric ellipsoid rendering for real-time view synthesis, 2025. 4

[6] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis, 2020. 1

[7] Tomas Möller and Ben Trumbore. Fast, minimum storage ray/triangle intersection. In *ACM SIGGRAPH 2005 Courses*, pages 7–es. 2005. 2

[8] Barbara Roessle, Jonathan T. Barron, Ben Mildenhall, Pratul P. Srinivasan, and Matthias Nießner. Dense depth priors for neural radiance fields from sparse input views, 2022. 1, 2

[9] Johannes Lutz Schönberger and Jan-Michael Frahm. Structure-from-motion revisited. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 2, 3

[10] Matias Turkulainen, Xuqian Ren, Iaroslav Melekhov, Otto Seiskari, Esa Rahtu, and Juho Kannala. Dn-splatter: Depth and normal priors for gaussian splatting and meshing, 2024. 1, 2, 3

[11] Nicolas von Lützow and Matthias Nießner. Linprim: Linear primitives for differentiable volumetric rendering, 2025. 1, 2, 3

[12] Meng Wei, Qianyi Wu, Jianmin Zheng, Hamid Rezatofighi, and Jianfei Cai. Normal-gs: 3d gaussian splatting with normal-involved rendering, 2024. 2

[13] Lihe Yang, Bingyi Kang, Zilong Huang, Zhen Zhao, Xiaogang Xu, Jiashi Feng, and Hengshuang Zhao. Depth anything v2, 2024. 2

[14] Chandan Yeshwanth, Yueh-Cheng Liu, Matthias Nießner, and Angela Dai. Scannet++: A high-fidelity dataset of 3d indoor scenes, 2023. 3, 1

DN-LinPrim: Depth- and Normals-regularized LinPrim

Supplementary Material

6. Appendix

6.1. View Selection

In order to subsample 10 views from the full scenes, we first voxelize the COLMAP 3D points into 10cm occupancy cubes, then greedily select views that maximize the number of seen voxels included in the final view set. ScanNet++ scenes used: 39f36da05b, 5a269ba6fe, dc263dfbf0, 08bbbdcc3d, fb564c935d.

6.2. Ablation Studies

Geometric Loss Regularization Parameters λ_D, λ_N To evaluate model performance for different regularization parameters, experiments were carried out on the complete scene dc263dfbf0 from ScanNet++ [14] at half resolution for λ_{D} and quarter image resolution for λ_{N} with third degree spherical harmonics. Combined, our final configuration uses $\lambda_D = 0.1$ and $\lambda_N = 0.05$. For more details see Tables 2 and 3.

$\lambda_{\mathbf{D}}$	RGB Reconstruction			Depth Accuracy		Normals
	PSNR ↑	SSIM ↑	LPIPS ↓	AbsRel ↓	δ_1 ↑	MAE ↓
0.001	22.99	0.847	0.274	0.117	0.836	30.10
0.01	23.38	0.850	<u>0.270</u>	<u>0.077</u>	0.924	28.78
0.10	23.25	**0.853**	**0.269**	**0.074**	**0.943**	**27.55**
0.25	23.14	<u>0.852</u>	0.275	0.082	<u>0.938</u>	<u>28.09</u>
0.50	23.33	0.852	0.284	0.085	0.932	28.54
0.75	<u>23.50</u>	0.850	0.292	0.083	0.936	29.25
1.00	**23.57**	0.848	0.300	0.084	0.933	29.74

Table 2. Ablation study on the depth loss weight λ_D. Bold indicates best performance per column, underline indicates second-best.

$\lambda_{\mathbf{N}}$	RGB Reconstruction			Depth Accuracy		Normals
	PSNR ↑	SSIM ↑	LPIPS ↓	AbsRel ↓	δ_1 ↑	MAE ↓
0.01	<u>23.69</u>	<u>0.848</u>	<u>0.243</u>	0.078	<u>0.941</u>	20.36
0.05	**23.81**	**0.849**	**0.240**	<u>0.074</u>	**0.943**	**17.10**
0.10	23.63	0.847	0.245	**0.070**	0.939	<u>17.26</u>
0.25	23.21	0.837	0.265	0.087	0.910	20.02
0.50	22.41	0.816	0.307	0.135	0.816	24.33
0.75	21.55	0.794	0.348	0.184	0.748	27.60

Table 3. Ablation study on the normal regularization weight λ_N. Bold indicates best performance per column, underline indicates second-best.

Method	RGB Reconstruction			Depth Accuracy		Normals
	PSNR ↑	SSIM ↑	LPIPS ↓	AbsRel ↓	δ_1 ↑	MAE ↓
LinPrim	22.58	0.870	0.266	0.131	0.798	30.66
LinPrim + $\mathcal{L}_{\text{D naive}}$	<u>23.30</u>	<u>0.876</u>	0.260	0.082	0.941	26.54
LinPrim + $\mathcal{L}_{\text{D inter}}$	23.15	0.876	0.260	<u>0.078</u>	**0.944**	27.55
LinPrim + $\mathcal{L}_{\text{D naive}} + \mathcal{L}_{\text{N}}$	**23.48**	**0.876**	**0.259**	0.082	0.942	<u>17.21</u>
LinPrim + $\mathcal{L}_{\text{D inter}} + \mathcal{L}_{\text{N}}$ (ours)	22.86	0.875	<u>0.259</u>	**0.077**	0.940	**16.88**

Table 4. Quantitative comparison of depth supervision variants (naive and intersection) on scene dc263dfbf0. Bold indicates best performance, underline indicates second-best per column.

Method	RGB Reconstruction			Depth Accuracy		Normals
	PSNR ↑	SSIM ↑	LPIPS ↓	AbsRel ↓	δ_1 ↑	MAE ↓
LinPrim	22.58	0.870	0.266	0.131	0.798	30.66
LinPrim + $\mathcal{L}_{\text{D}}$	23.15	**0.876**	0.260	0.078	**0.944**	27.55
LinPrim + $\mathcal{L}_{\text{N self reg}}$	21.73	0.865	0.290	0.179	0.708	35.24
LinPrim + $\mathcal{L}_{\text{N}}$	22.12	0.870	0.267	0.124	0.818	<u>18.51</u>
LinPrim + $\mathcal{L}_{\text{D}} + \mathcal{L}_{\text{N self reg}}$	**23.33**	<u>0.875</u>	<u>0.265</u>	<u>0.077</u>	<u>0.943</u>	26.17
LinPrim + $\mathcal{L}_{\text{D}} + \mathcal{L}_{\text{N}}$ (ours)	22.86	0.875	**0.259**	**0.077**	0.940	**16.88**

Table 5. Quantitative comparison of loss variants on scene dc263dfbf0. For the depth loss $\mathcal{L}_{\text{D}}$ we consider the intersection variant always. Bold indicates best performance, underline indicates second-best per column.