# Splat-LOAM: Gaussian Splatting LiDAR Odometry and Mapping

Emanuele Giacomini[1] Luca Di Giammarino[1] Lorenzo De Rebotti[1] Giorgio Grisetti[1]
Martin R. Oswald[2]
[1]Sapienza University of Rome, [2]University of Amsterdam, Netherlands

## Abstract

*LiDARs provide accurate geometric measurements, making them valuable for ego-motion estimation and reconstruction tasks. Although its success, managing an accurate and lightweight representation of the environment still poses challenges. Both classic and NeRF-based solutions have to trade off accuracy over memory and processing times. In this work, we build on recent advancements in Gaussian Splatting methods to develop a novel LiDAR odometry and mapping pipeline that exclusively relies on Gaussian primitives for its scene representation. Leveraging spherical projection, we drive the refinement of the primitives uniquely from LiDAR measurements. Experiments show that our approach matches the current registration performance, while achieving SOTA results for mapping tasks with minimal GPU requirements. This efficiency makes it a strong candidate for further exploration and potential adoption in real-time robotics estimation tasks.*

## 1. Introduction

LiDAR sensors provide accurate spatial measurements of the environment, making them valuable for ego-motion estimation and reconstruction tasks. Since the measurements already capture the 3d structure, many LiDAR Simultaneous Localization And Mapping (SLAM) pipelines do not explicitly refine the underlying point representation [2, 3, 7, 9, 12, 37]. Moreover, a global map can be obtained by directly stacking the measurements together. However, this typically leads to extremely large point clouds that cannot be explicitly used for online applications. Several approaches attempted to use surfels [1, 31] and meshes [32]. Although these approaches manage to simultaneously estimate the sensor's ego-motion while optimizing the map representation, they result in a trade-off between accuracy, memory usage, and runtime.

The recent advent of NeRF [26] sparked fresh interest in Novel View Synthesis (NVS) tasks. Specifically, given a set of input views and triangulated points (i.e. obtainable via COLMAP [35, 36]), NeRF learns a continuous volumetric scene function. Color and density information is propagated throughout the radiance field by ray-casting pixels from the view cameras and, through a Multi-Layer Perceptron (MLP), the method learns a representation that ensures multi-view consistency. Concurrently, NeRFs inspired the computer vision community to tackle the problem of dense visual SLAM through *implicit* methods. The pioneering work in this context was iMAP [40], which stores the global appearance and geometry of the scene through a single MLP. Despite its success in driving a new research direction, the method suffered from issues related to the limited capacity of the model, leading to low reconstruction quality and catastrophic forgetting during the exploration of larger areas. These issues were later handled by shifting the paradigm and by moving some of the appearance and geometric information over a hierarchical feature grid [54] or neural point clouds [22, 33]. Similarly, these techniques were employed on LiDAR measurements to provide more accurate and lighter explicit dense representations [8, 16, 30, 39, 52]. However, these methods still require tailored sampling techniques to estimate the underlying Signed Distance Function (SDF) accurately. This bottleneck still poses problems for online execution.

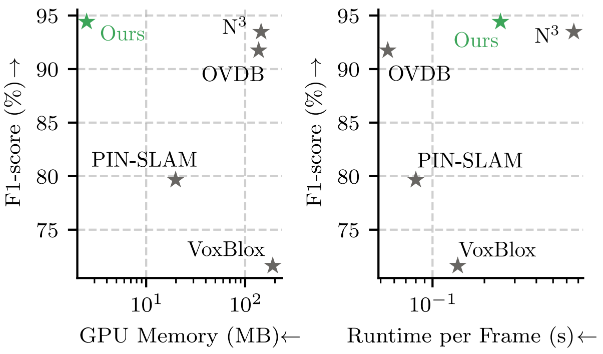

Figure 1. **Performance overview of Splat-LOAM.** F1 score to Active Memory (MB) and Runtime (s). The plots provide a quantitative comparison between state-of-the-art mapping pipelines, while PIN-SLAM and ours also perform online odometry.

A recent, explicit alternative to NeRFs is 3D Gaussian Splatting (3DGS) [20]. This approach leverages 3D

Gaussian-shaped primitives and a differentiable, tile-based rasterizer to generate an appearance-accurate representation. Furthermore, having no need to model empty areas and no neural components, 3DGS earned a remarkable result in accuracy and training speeds. Additionally, several approaches further enhanced the reconstruction capabilities of this representation [6, 13, 15]. Being fast and accurate, this representation is now sparking interest in dense visual SLAM. Recently, 3D Gaussians were employed in several works, yielding superior results over implicit solutions [24, 47, 53].

One issue with Gaussian Splatting relates to the primitive initialization. In regions where few or no points are provided by SFM, adaptive densification tends to fail, often yielding under-reconstructed regions. LiDAR priors are particularly useful in solving this issue due to explicit spatial measurements that can be used to initialize the local representation [14, 46].

However, to our knowledge, no attempt has been made to evaluate the this representation for LiDAR-only scenarios. This technique could prove interesting for visual NVS initialization and LiDAR mapping as it could produce a lightweight, dense, and consistent representation. These insights led us to the development of Splat-LOAM, the first LiDAR Odometry and Mapping pipeline that only leverages Gaussian primitives as its surface representation. Our system demonstrates results on par with other SOTA pipelines at a fraction of the computational demands, proving as an additional research direction for real-time perception in autonomous systems.

## 2. Related Work

**Classic LiDAR Odometry and Mapping.** *Feature*-based methods that leverage specific points or groups of points to perform incremental registration. For instance, [38, 49] track feature points on sharp edges and planar surface patches, enabling high-frequency odometry estimation. On the other hand, *Direct* methods leverage the whole cloud to perform registration. Specifically, these methods can be categorized based on the subjects of the alignment. *Scan-to-Scan* methods matches subsequent clouds, either explicitly [2, 3, 37] or through neural methods [21, 44], while *Scan-to-Model* methods match clouds with either a local or a global map. Typically, the map is represented using points [7, 12], surfels [1, 31], meshes [32]. Another explored solution project the measurements onto a spherical projection plane to leverage visual techniques for ego-motion estimation [9, 10, 28, 51].

**Implicit Methods.** Concerning mapping only methods, Zhong *et al.* [52] proposed the first method for LiDAR implicit mapping that, given a set of point clouds and the corresponding sensor poses, used hierarchical feature grids to estimate the SDF of the scene. Their results demonstrated once again the advantages of such representation for surface reconstruction in terms of accuracy and memory footprint. A similar approach from Song *et al.* [39] improves the mapping accuracy by introducing SDF normal guided sampling and a hierarchical, voxel-guided sampling strategy for local optimization. Building on these advancements, several LiDAR odometry and mapping techniques were proposed. Deng *et al.* [8] presented the first implicit LiDAR LOAM system using an octree-based feature representation to encode the scene's SDF, used both for tracking and mapping. Similarly, Isaacson *et al.* [16] proposes a hierarchical feature grid to store the SDF information while using point-to-plane ICP to register new clouds. Pan *et al.* [30] leverages a neural point cloud representation to ensure a globally consistent estimate. The new clouds are registered using a correspondence-free point-to-implicit model approach. These methods prove that implicit representation can offer SOTA results in accuracy at the cost of potentially high execution times or memory requirements. Although targeting a different problem setting, related are also a series of visual neural SLAM methods with RGB or RGBD input [19, 22, 23, 33, 34, 47, 48, 54], see [42] for a survey.

**Gaussian Splatting.** Few works tackle the use of LiDAR within the context of Gaussian Splatting. Wu *et al.* [46] propose a multi-modal fusion system for SLAM. Specifically, by knowing the LiDAR to camera rigid pose, the initial pose estimate is obtained through point cloud registration and further refined via photometric error minimization. In this framework, the LiDAR points are leveraged to initialize the new 3D Gaussians. In a related approach, Hong *et al.* [14] proposes a LiDAR-Inertial-Visual SLAM system. The initial estimate is computed through a LiDAR-Inertial odometry. Points are partitioned using size-adaptive voxels to initialize 3D Gaussians using per-voxel covariances. Both the primitives and the trajectory are further refined via photometric error minimization. While these methods introduce LiDAR measurement, 3D Gaussians are still inherently processed by cameras. Recently, Chen *et al.*[5] applies 3D Gaussians for the task of LiDAR NVS for re-simulation. The authors propose the use of Periodic Vibrating 3D Gaussian primitives to account for dynamic objects present in the scene. The primitives are initialized using a lightweight MLP, and the rasterization is carried out in a spherical frame by computing a per-primitive plane orthogonal to the ray that connects the primitive's mean to the LiDAR frame, thus removing any distortion in the projection process. Focusing on a different formulation, Jiang *et al.* [17] propose a method of LiDAR NVS that leverages Periodic Vibrating 2D Gaussian primitives. The primitives are initialized by randomly sampling points and, further optimized using the losses described in [15], along with a Chamfer loss is introduced to constrain the 3D structures of the synthesized point

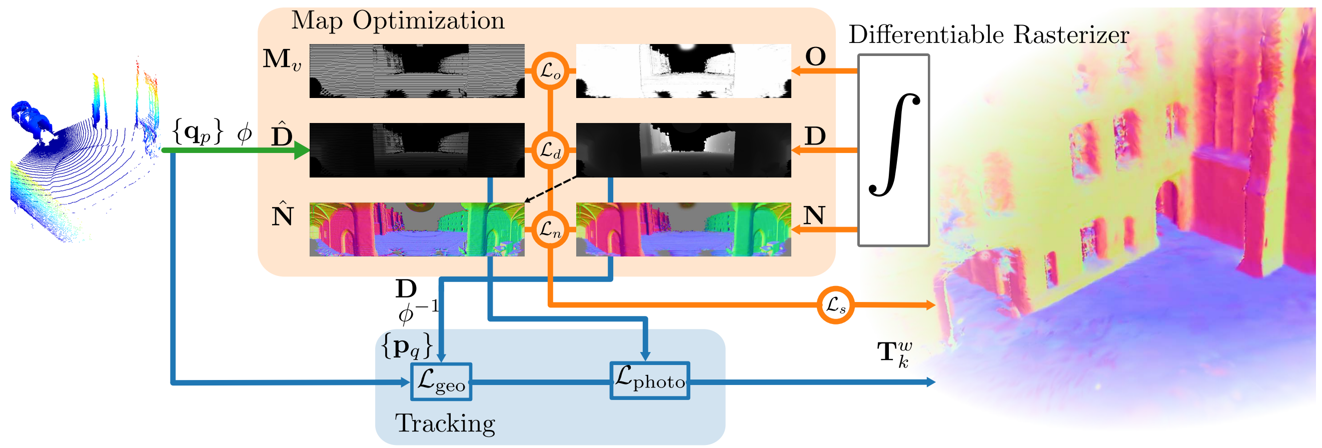

Figure 2. **Splat-LOAM Overview.** Given a LiDAR point cloud, we leverage the spherical projection to generate an image-like representation. Moreover, using an ad-hoc differentiable rasterizer, we guide the optimization for structural parameters of 2D Gaussians. The underlying representation is concurrently used to incrementally register new measurements.

clouds, and an additional ray-drop term to account for phenomena like non-returning laser pulses. This term is further refined through a U-Net that considers other factors, such as the distance of the surface from the sensor. Compared to this work, we provide a thorough methodology to render 2D Gaussians on spherical frames while accounting for coordinates singularity and a cloud registration technique to allow for simultaneous odometry and mapping. To our knowledge, Splat-LOAM is the first pure LiDAR Odometry and Mapping pipeline that leverages Gaussian primitives both for mapping and tracking. In sum, our contributions are

- A differentiable, tile-based rasterizer for 2D Gaussians over spherical manifolds.
- A mapping pipeline that allows the integration of LiDAR measurements in a 2D Gaussian representation.
- A tracking schema that leverages both 3D and 2D representations to register new measurements and estimate the sensor ego-motion.

## 3. Method

This section introduces our novel LiDAR odometry and mapping method based on 2D Gaussian primitives. We detail a mapping strategy for initializing, refining, and integrating these primitives alongside a registration method that leverages geometric and photometric cues from the continuous local model for ego-motion estimation. Additional details can be found in the Supplementary material.

### 3.1. Spherical Projection Model

While original Gaussian Splatting leverages pinhole-camera projection to render or refine 3D primitives, LiDAR input provides 360° panoramic input. To this end, we employ spherical projection to encode LiDAR measurements into an image-like representation that we use to guide the Gaussians optimization. A projection is a mapping $\phi : \mathbb{R}^3 \rightarrow \Gamma \subset \mathbb{R}^2$ from a world point $\mathbf{p} = (x, y, z)^T$ to image coordinates $\mathbf{u} = (u, v)^T$. Knowing the *range* $d = \|\mathbf{p}\|$ of an image point $\mathbf{u}$, we can calculate the inverse mapping $\phi^{-1} : \Gamma \times \mathbb{R} \rightarrow \mathbb{R}^3$, more explicitly $\mathbf{p} = \phi^{-1}(\mathbf{u}, d)$. We refer to this operation as back-projection. To ease the clarity of this work, it is worth mentioning that, compared to the classical pinhole camera, the optical reference frame is rearranged; the $x$-axis points forward, $y$-axis points to the left, and $z$-axis points upwards. Let $\mathbf{K}$ be a camera intrinsics matrix that can be computed directly from the point cloud (see Supplementary Material), with function $\psi$ mapping a 3D point to azimuth and elevation. Thus, the spherical projection of a point is given by

$$\phi(\mathbf{p}) = \mathbf{K}\psi(\mathbf{p}), \tag{1}$$

$$\psi(\mathbf{v}) = \begin{bmatrix} \operatorname{atan2}(v_y, v_x) \\ \operatorname{atan2}\left(v_z, \sqrt{v_x^2 + v_y^2}\right) \\ 1 \end{bmatrix}. \tag{2}$$

We used spherical projection to obtain a range map $\hat{\mathbf{D}}$ of the LiDAR point cloud $\{\mathbf{q}_p\}_{p=1}^{Q}$ of size Q.

### 3.2. 2D Gaussian Splatting

Our scene representation is based on 2D Gaussians due to their affinity to surface reconstruction [6, 15]. A 2D Gaussian is defined by its opacity $o \in [0, 1]$, centroid $\boldsymbol{\mu} \in \mathbb{R}^3$, origin plane defined by two tangential vectors $\mathbf{t}_\alpha \in \mathbb{R}^3$ and $\mathbf{t}_\beta \in \mathbb{R}^3$, and scaling vector $\mathbf{s} = (s_\alpha, s_\beta) \in \mathbb{R}^2$ that controls the variance of the Gaussian kernel [15]. Points on the splat's plane can be described in world space using:

$$\mathbf{H} = \begin{pmatrix} s_\alpha \mathbf{t}_\alpha & s_\beta \mathbf{t}_\beta & \mathbf{0} & \boldsymbol{\mu} \\ 0 & 0 & 0 & 1 \end{pmatrix}, \tag{3}$$

while the Gaussian kernel can be evaluated on a point $(\alpha, \beta)$ in splat space using:

$$\mathcal{G}(\alpha, \beta) = \exp\left(-\frac{\alpha^2 + \beta^2}{2}\right). \quad (4)$$

### 3.2.1. Rasterization

We render the Gaussians via $\alpha$-blending as in [15, 20]. Specifically, for each pixel $\mathbf{u}$, we integrate Gaussians from front to back to obtain the range $d$, normal $\mathbf{n}$, and opacity $o$ as follows:

$$d = \sum_{i=1}^{T} o_i \mathcal{G}_i d_i \prod_{j=1}^{i-1} (1 - o_j \mathcal{G}_j) \quad (5)$$

$$\mathbf{n} = \sum_{i=1}^{T} o_i \mathcal{G}_i \mathbf{t}_{n_i} \prod_{j=1}^{i-1} (1 - o_j \mathcal{G}_j) \quad (6)$$

$$o = \sum_{i=1}^{T} o_i \mathcal{G}_i \prod_{j=1}^{i-1} (1 - o_j \mathcal{G}_j) \quad (7)$$

Different from the pinhole model, the spherical projection model described in Sec. 3.1 is non-affine, resulting in strong distortion artifacts on Gaussians that cover a significant extent in the image plane, if rasterizing using the local affine approximation method proposed in [20].

To this extent, as in [15], our rasterizer leverages exact ray-splat intersections to render 2D Gaussians. This approach is particularly suited for our projection model as it explicitly relies on per-pixel directions to evaluate the Gaussian kernels, thus allowing us to account for the structure of our projection manifold surface.

The ray-splat intersection is based on [45], and differently from [15], we reformulate the ray parametrization to cover the whole azimuthal space around the sensor. Additionally, we propose a method to compute the image-space extent of each Gaussian that is robust to the azimuthal coordinate singularity, ensuring accurate rasterization for any Gaussian configuration.

### 3.2.2. Ray-splat Intersection

Let $\mathbf{v} = \phi^{-1}\left(\mathbf{K}^{-1}\mathbf{u}\right)$ be the normalized direction of pixel $\mathbf{u}$, parametrized as the intersection of two orthogonal planes

$$\mathbf{h}_x = \frac{\mathbf{v} \times \mathbf{u}_z}{\|\mathbf{v} \times \mathbf{u}_z\|} \;, \qquad \mathbf{h}_y = \mathbf{h}_x \times \mathbf{v} \;, \quad (8)$$

where $\mathbf{u}_z$ is the unit $z$-direction vector. This formulation is valid when $\mathbf{v} \neq \mathbf{u}_z$ which is a fair assumption in LiDAR measurements. Moreover, the planes are described in splat's space as

$$\mathbf{h}_\alpha = \left(\mathbf{T}_w^c \mathbf{H}\right)^T \mathbf{h}_x \;, \qquad \mathbf{h}_\beta = \left(\mathbf{T}_w^c \mathbf{H}\right)^T \mathbf{h}_y \;, \quad (9)$$

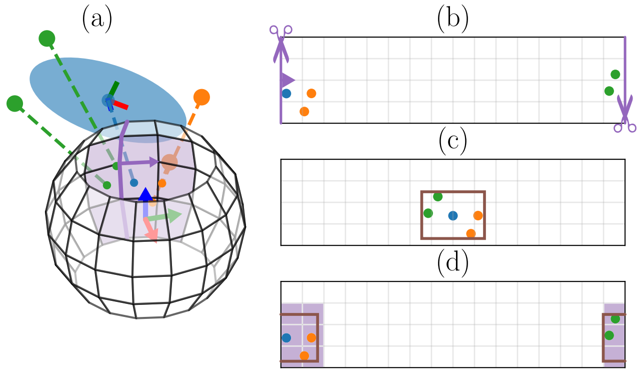

Figure 3. **Bounding box computation for near-singularity splats. (a)** shows the 3D configuration of a splat that approximately lies behind the camera. **(b)** shows the corresponding spherical image with the projected bounding box vertices. The distortion is removed by shifting the vertices along the horizontal direction to align the projected center to the image center. **(c)** being far from the coordinate singularity, we compute the maximum extent of the splat. **(d)** We revert the shift and propagate to the corresponding tiles via an offset from the central vertex, matching with the tiles highlighted on (a).

where $\mathbf{T}_w^k \in \mathbb{SE}(3)$ describes the *world* frame in the $k$-th sensor reference frame. We express the intersection of the two planes bundle as:

$$\mathbf{h}_\alpha \left(\alpha, \beta, 1, 1\right)^T = \mathbf{h}_\beta \left(\alpha, \beta, 1, 1\right)^T = 0 \;. \quad (10)$$

The solution $\alpha$ and $\beta$ is then computed by solving the homogeneous linear system:

$$\alpha(\mathbf{u}_s) = \frac{\mathbf{h}_\alpha^2 \mathbf{h}_\beta^4 - \mathbf{h}_\alpha^4 \mathbf{h}_\beta^2}{\mathbf{h}_\alpha^1 \mathbf{h}_\beta^2 - \mathbf{h}_\alpha^2 \mathbf{h}_\beta^1} \;, \quad \beta(\mathbf{u}_s) = \frac{\mathbf{h}_\alpha^4 \mathbf{h}_\beta^1 - \mathbf{h}_\alpha^1 \mathbf{h}_\beta^4}{\mathbf{h}_\alpha^1 \mathbf{h}_\beta^2 - \mathbf{h}_\alpha^2 \mathbf{h}_\beta^1} \;. \quad (11)$$

### 3.2.3. Bounding Box Computation

To efficiently rasterize Gaussians, each tile is informed of which Gaussians it should process. This is achieved by computing the per-Gaussian image-space extent from its projected central point and using it to inform each covered tile to rasterize the Gaussian. In our work, we compute the per-Gaussian extent by projecting the vertices of the $3\sigma$ splat-space bounding box on the spherical frame. Due to the horizontal periodic nature of the spherical frame, for Gaussians whose center is projected near the horizontal image boundaries and has a significant image-space size, one or more vertices of the bounding box may be projected on the opposite side of the image. If not handled, this situation can degrade the rasterization since not all the corresponding tiles may be informed, thus skipping rendering for some pixels and slowing the rendering process since the computed extent might almost cover the whole image plane.

We solve this by computing the Gaussian extent and tile propagation in a condition where the coordinate singularity cannot be harmful. As shown in Fig. 3, for each Gaussian, we project its bounding box and shift its projected points towards the center of the image plane while accounting for coordinate singularity. Similarly, after computing the Gaussian radius, the information is propagated to neighboring tiles while accounting for periodicity, ensuring correct rasterization and reducing the number of selected tiles.

## 3.3. Odometry And Mapping

Following the modern literature of RGB-D SLAM [22, 34, 47, 53], we rely on keyframing to optimize local maps. We choose this approach for the following advantages: first, continuous integration over the same model can have adverse effects on artifacts and, more importantly, on runtime. Instead of decreasing the local density, we reset to a new local model if certain conditions are met, while also restricting the number of frames joining the optimization stage to allow effective online processing Based on this, we define each local map as an individual Gaussian model $\mathbf{P}^s$ as

$$\mathbf{P}^s = \{\mathcal{G}(\boldsymbol{\mu}, \boldsymbol{\Sigma}, o) | i = 1, \ldots, N\} . \tag{12}$$

### 3.3.1. Local model initialization

We initialize a new local model using the input LiDAR point cloud whenever necessary, such as at system startup or when visibility conditions require it. As a first step, we generate a valid pixel mask via the indicator function $\mathbb{1}[\cdot]$ as

$$\mathbf{M}_v = \mathbb{1}[\hat{\mathbf{D}} > 0], \tag{13}$$

and compute the range gradients $\boldsymbol{\nabla}_{\mathbf{D}}$ to construct a weight map over the range image. We use a weighted sampling of $n_d$ points to prioritize complex regions. Splat positions are computed by back-projecting the range image, while their orientations are initialized by directing surface normals toward the sensor center to enhance initial visibility.

### 3.3.2. Local Model Refinement

We perform a limited number of refinement iterations $n_o$ on the most recent keyframes. Unlike [47, 53], we employ a geometric distribution-based sampling scheme that guarantees at least $40\%$ probability of selecting the most recent keyframe and progressively decreases the likelihood of choosing the older ones. To filter out artifacts caused by ray-drop phenomena in the LiDAR measurements, we only consider valid pixels to refine the local model parameters $\mathbf{x}$. We start by applying a densification strategy similar to the one described in Sec. 3.3.1, with two extra terms. The first one, $\mathbf{M}_n = \mathbb{1}[\mathbf{O}_i \leq \lambda_{d,o}]$, target newly discovered areas while the second, $\mathbf{M}_e = \mathbb{1}[|\mathbf{D}_i - \hat{\mathbf{D}}_i| \geq \lambda_{d,e}]$, target under-reconstructed regions.

To optimize the geometric consistency of the local model, we employ a loss term that minimizes the $L_1$ error:

$$\mathcal{L}_d = \sum_{\mathbf{u} \in \mathbf{M}_v} \rho_d \|\mathbf{D}(\mathbf{u}, \mathbf{x}) - \hat{\mathbf{D}}(\mathbf{u})\|, \tag{14}$$

where $\rho_d$ is a weight function dependent on the measurement's range. In addition, we employ a self-regularization term to align the splat's normals to the surface normals $\mathbf{N}$ estimated by the gradients of the range map $\mathbf{D}$ [15]:

$$\mathcal{L}_n = \sum_{\mathbf{u} \in \mathbf{M}_v} 1 - \mathbf{n}^T \mathbf{N}\big(\mathbf{D}(\mathbf{u}, \mathbf{x})\big) \tag{15}$$

Furthermore, to promote the expansion of splats over uniform surfaces, we introduce an additional term that operates on the opacity channel of the rasterized images. Specifically, we drive the splats to cover the areas of the image containing valid measurements by correlating the opacity image $\mathbf{O}$ with the valid mask $\mathbf{M}_v$.

$$\mathcal{L}_o = \sum_{\mathbf{u} \in \mathbf{M}_v} - \log \big(\mathbf{O}(\mathbf{u}, \mathbf{x})\big). \tag{16}$$

While $\mathcal{L}_o$ allows the splats to grow, it can also cause some splats to become extremely large in unobserved areas. We employ an additional regularization term on the scaling parameter of all primitives $N$ to compensate for this effect. We propose a novel regularization that allows the splat to extend up to a certain value $\tau_s$ before penalizing its growth:

$$\mathcal{L}_s = \begin{cases} \tau_s - \max(s_\alpha, s_\beta) & \text{if } \max(s_\alpha, s_\beta) > \tau_s \\ 0 & \text{otherwise} \end{cases} \tag{17}$$

We found this term to be more effective rather than directly minimizing the deviation from the average [24, 47, 53] as it allows for anisotropic splats that are particularly useful for mapping details and edges, and provides more control on the density and structure of the model.

The final mapping objective function is defined as:

$$\mathcal{L}_{\text{map}} = \mathcal{L}_d + \lambda_o \mathcal{L}_o + \lambda_n \mathcal{L}_n + \lambda_s \sum_{i=1}^{N} \mathcal{L}_{s_i} \ , \tag{18}$$

with $\lambda_o, \lambda_n, \lambda_s \in \mathbb{R}$ being loss weights. This update strategy allows our method to maintain a geometrically accurate scene representation, while selectively densifying under-reconstructed regions that require more resolution, using a limited number of iterations. To avoid catastrophic forgetting, we do not perform any opacity reset steps. Moreover, we prune Gaussians that are either transparent or very small, as they typically cannot be observed during rendering.

### 3.3.3. Frame-To-Model Registration

Each time a new keyframe is selected, we sample the local model by back-projecting the rasterized range image $\mathbf{D}$ onto a point cloud $\{\mathbf{p}_q\}_{q=1}^{W \times H}$ at its estimated pose. We design an ad-hoc tracking schema to benefit from both geometric and photometric consistencies provided by the LiDAR and the rendered local model. Hence, the total odometry loss is composed of the sum of both residuals:

$$\mathcal{L}_{\text{odom}} = \mathcal{L}_{\text{geo}} + \mathcal{L}_{\text{photo}}. \tag{19}$$

**Geometric Registration.** To associate geometric entities, we employ a PCA-based kd-tree [12]. The kd-tree is built on the back-projected rendered range map $\{\mathbf{p}_q\}_{q=1}^{W\times H}$ and segmented into tree leaves, corresponding to planar patches. Each leaf corresponds to $l = \langle \mathbf{p}_l, \mathbf{n}_l \rangle$, that is, the mean point $\mathbf{p}_l \in \mathbb{R}^3$ and the surface normal $\mathbf{n}_l \in \mathbb{R}^3$. The geometric loss $\mathcal{L}_{\text{geo}}$ represents the sum of the point-to-plane distance between the mean leaf $\mathbf{p}_l$ and the point of the current measurement point cloud $k$ with $\{\mathbf{q}_p\}_{p=1}^{Q}$, along the normal $\mathbf{n}_l$ expressed in the local reference frame $\mathbf{T}_w^k$. Specifically, we have:

$$\mathcal{L}_{\text{geo}} = \sum_{p,q\in\{a\}} \rho\left((\mathbf{T}_w^k \mathbf{n}_{l_q})^T (\mathbf{T}_w^k \mathbf{q}_p - \mathbf{p}_{q_i})\right), \tag{20}$$

where $w$ is the global frame, $k$ is the local frame, $\{a\}$ is the set of leaf associations with the point from the measurement point cloud, and $\rho$ is the Huber robust loss function.

**Photometric Registration.** Leveraging the rendered range map $\mathbf{D}$, we employ photometric registration for subpixel refinement, minimizing the photometric distance between the rendered and the spherical projected query point cloud $\hat{\mathbf{D}}$. The photometric loss is formulated as:

$$\mathcal{L}_{\text{photo}} = \sum_{\mathbf{u}} \left\| \rho\Big(\mathbf{D}(\mathbf{u}) - \hat{\mathbf{D}}\underbrace{\left(\phi(\mathbf{T}_w^k \phi^{-1}(\mathbf{u}, \hat{d}))\right)}_{\mathbf{u}'}\Big) \right\|^2. \tag{21}$$

The evaluation point $\mathbf{u}'$ in the query image $\hat{\mathbf{D}}$ is computed by first back-projecting the pixel $\mathbf{u}$, applying the transform $\mathbf{T}_w^k$, and then projecting it back. Pose updates $\delta$ are parameterized as local updates in the Lie algebra $\mathfrak{se}(3)$. Therefore, the transformation $\mathbf{T}_k^w$ of the local reference frame, expressed in the global reference frame, is updated as:

$$\mathbf{T}_k^w \leftarrow \mathbf{T}_k^w \exp(\delta). \tag{22}$$

This update is carried out using a second-order Gauss-Newton method. Local updates ensure that rotation updates are well-defined [11].

## 4. Experiments

In this section, we report the results of our method on several publicly available datasets. We evaluate our pipeline on tracking and mapping. We recall that, to our knowledge, this is the first Gaussian Splatting LiDAR Odometry and Mapping pipeline, and no direct competitors are available. We compared our approach with other well-known geometric and neural implicit methods. The experiments were executed on a PC with an Intel Core i9-14900K @ 3.20Ghz, 64GB of RAM, and an NVIDIA RTX 4090 GPU with 24 GB of VRAM.

For odometry evaluation, we include several baselines. The first one is a basic point-to-plane ICP odometry, followed by SLAMesh, that simultaneously estimates a mesh representation of the scene via Gaussian Processes and performs registration onto it [32]. Moreover, MAD-ICP leverages a forest of PCA-based KD-Trees to perform accurate registration. Furthermore, we include PIN-SLAM as SOTA baseline for implicit LiDAR SLAM [30]. It relies on neural point primitives and interleaves an incremental learning of the model's SDF and a correspondence-free, point-to-implicit registration schema.

For mapping evaluation, we include OpenVDB [27] and VoxBlox [29] as "classic" baselines. OpenVDB provides a robust volumetric data structure to handle 3D Points, while VoxBlox combines adaptive weights and grouped ray-casting for an efficient and accurate Truncated Signed Distance Function (TSDF) integration. Additionally, we include as neural-implicit approaches, $N^3$-Mapping [39] and PIN-SLAM [30]. $N^3$-Mapping is a neural-implicit non-projective SDF mapping approach that leverages normal guidance to produce more accurate SDFs, leading to SOTA results for offline LiDAR mapping. Moreover, PIN-SLAM is included as meshes can be extracted directly from the implicit representation via marching cubes. Below, we report the datasets and the evaluation metrics employed. We do not include SLAMesh [32] since it could not be run with ground-truth poses.

### 4.1. Datasets

We used the following four publicly available datasets:

- **Newer College Dataset (NC) [50]:** Collected with a handheld Ouster OS0-128 LiDAR in structured and vegetated areas. Ground truth was generated using the Leica BLK360 scanner, achieving centimeter-level accuracy over poses and points in the map.
- **A Vision Benchmark in Rome (VBR) [4]:** Recorded in Rome using OS1-64 (car-mounted) and OS0-128 (handheld) LiDARs, capturing large-scale urban scenarios with narrow streets and dynamic objects. Ground truth trajectories were obtained by fusing LiDAR, IMU, and RTK

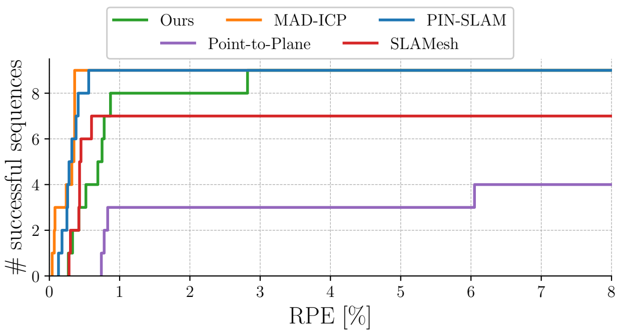

Figure 4. **RPE evaluation.** Number of successful sequences across RPE thresholds. It includes the sequences of Newer College [49], VBR [4], Oxford Spires [41] and Mai City [43].

| Dataset | Newer College[50] | | | | Oxford Spires[41] | | | | | | Mai-City[43] | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | quad-easy | | math-easy | | keble-02 | | bodleian-02 | | observatory-01 | | mai-01 | | mai-02 | |
| Approach | C-l1↓ | F-score↑ | C-l1↓ | F-score↑ | C-l1↓ | F-score↑ | C-l1↓ | F-score↑ | C-l1↓ | F-score↑ | C-l1↓ | F-score↑ | C-l1↓ | F-score↑ |
| OpenVDB[27] | 7.92 | 88.85 | 11.69 | 84.51 | 7.19 | 91.74 | **7.51** | 89.68 | 9.59 | 86.16 | 3.33 | 97.29 | 3.26 | 97.37 |
| VoxBlox[29] | 16.5 | 64.63 | 11.93 | 80.51 | 15.03 | 71.63 | 15.24 | 58.77 | 15.12 | 70.45 | 7.04 | 93.81 | 5.80 | 95.51 |
| $N^3$-Mapping[39] | 8.04 | 94.54 | fail | fail | **7.01** | 93.47 | 7.89 | **90.36** | **9.35** | **87.94** | **2.64** | **99.06** | **2.76** | **99.04** |
| PIN-SLAM[30] | 12.89 | 88.05 | 14.69 | 73.82 | 11.83 | 79.65 | 10.74 | 82.71 | 14.49 | 72.31 | 4.91 | 93.84 | 4.82 | 94.11 |
| Ours | **5.37** | **96.74** | **9.15** | **90.02** | 7.43 | **94.41** | 7.60 | 90.09 | 10.56 | 83.04 | 3.64 | 97.27 | 5.05 | 95.31 |

Table 1. **Reconstruction evaluation.** The pipelines were run with ground-truth poses. Voxel size is 20 cm and F-score is computed with 20 cm threshold. Splat-LOAM yield competitive mapping performance on both the Newer College[50] and Oxford Spires[41] dataset.

GNSS data, ensuring centimeter-level accuracy over the poses. This dataset is not used for mapping evaluation as no ground-truth maps are provided.

- **Oxford Spires [41]:** Recorded with a hand-held *Hesai QT64* LiDAR featuring a 360° horizontal FoV, 104° vertical FoV, 64 vertical channels, and 60 meters of range detection. Similar to [49], each sequence includes a prior map obtained via a survey-grade 3D imaging laser scanner, used for ground-truth trajectory estimation and mapping evaluation. Specifically, we choose the *Keble College*, *Bodleian Library*, and *Radcliffe Observatory* sequences to include both indoor and outdoor scenarios with different levels of detail.
- **Mai City [43]:** A synthetic dataset captured using a car-like simulated LiDAR with 120 meters of range detection. The measurements are generated via ray-casting on an underlying mesh, providing error-free, motion-undistorted data. We select the *01* and *02* sequences, which capture similar scenarios with different vertical resolutions.

## 4.2. Evaluation

We use Relative Pose Error (RPE) computed with progressively increasing delta steps to evaluate the odometry accuracy. Specifically, we adapt the deltas to the trajectory length to provide a more meaningful result [4].

Differently, to evaluate mapping quality, we use several metrics [25]: Accuracy (Acc) measures the mean distance of points on the estimated mesh with their nearest neighbors on the reference cloud. Completeness (Comp) measures the opposite distance, and Chamfer-l1 (C-l1) describes the mean of the two. Additionally, we use the F-score computed with 20 cm error threshold. Table 1 reports compact results for C-l1 and F1-score, additional tables are available in the Supplementary Material.

## 4.3. Ablation Study

Our approach employs Gaussian primitives for LiDAR odometry estimation and mapping, yielding results comparable to state-of-the-art methods while significantly enhancing computational efficiency. In this section, we evaluate some key aspects of our pipeline and evaluate their contributions.

**Memory and Runtime Analysis.** In Fig. 5, we report how the increment of active primitives affects the active GPU memory requirements and the per-iteration mapping frequency. It shows an experiment run over the large-scale *campus* sequence [4] where we set a maximum of 100 keyframes per local map and sample, at most, 50% of points for the incoming point cloud. It's possible to notice that the mapping FPS remains stable between 200k and 300k primitives. This is most likely related to the saturation of the rasterizer.

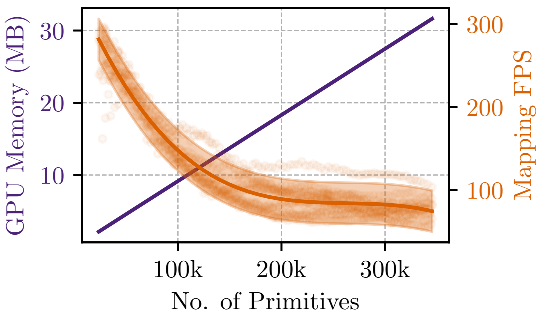

Figure 5. **Memory and Runtime Analysis.** The plot relates the used GPU Memory and the mapping iteration frequency with the number of active primitives. The measurements were obtained over the longest sequence we reported: campus [4].

**Odometry.** Fig. 7 shows the results for different tracking methods over our scene representation. Using both geometric and photometric components, we achieve a better result than using point-to-point or point-to-plane. The last three bars show an ablation of geometric and photometric loss components. The results are best for the joint use of both terms which support our design choice.

**Mesh generation.** We can sample the Gaussian scene to produce a mesh representation. Similar to [13], the meshing process involves sampling $n_m$ points from each keyframe rendered range and normal maps, and running Poisson reconstruction [18]. As shown in Tab. 2, we found

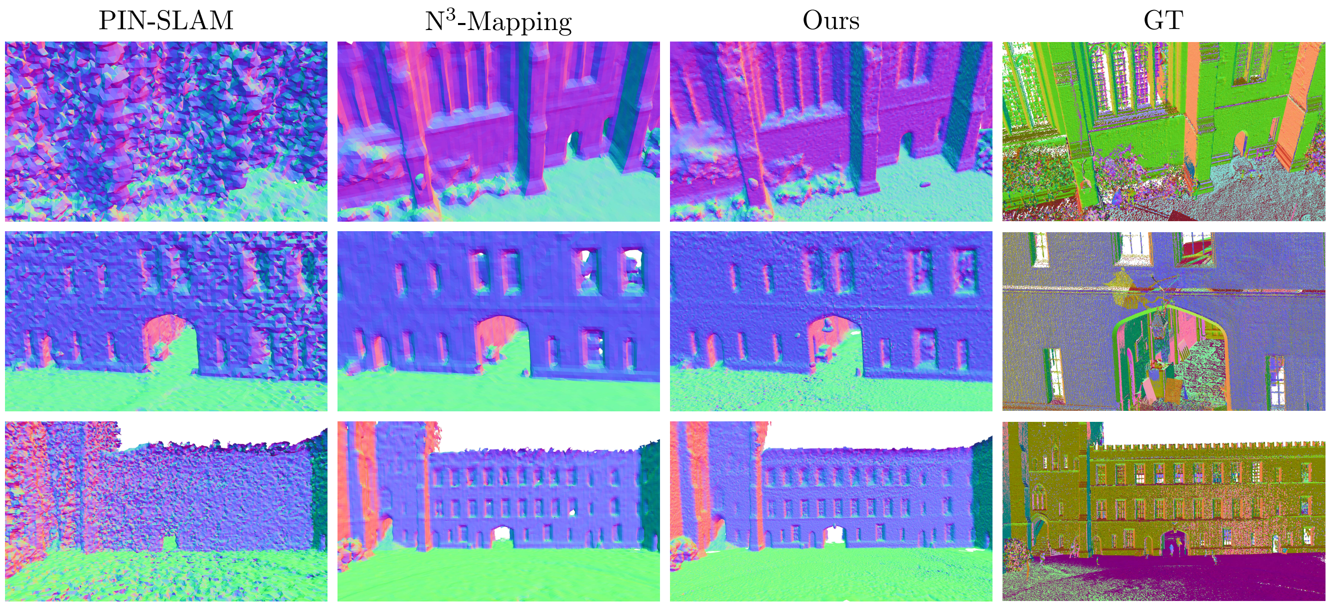

Figure 6. **Comparison of Mesh Reconstruction.** The figure shows reconstruction results for *quad-easy* sequence from the newer college dataset. Our method recovers a geometry with much higher data fidelity. PIN-SLAM lacks many details and exhibits a large level of noise. $N^3$-Mapping performs more similar to ours, but oversmoothes fine geometric details.

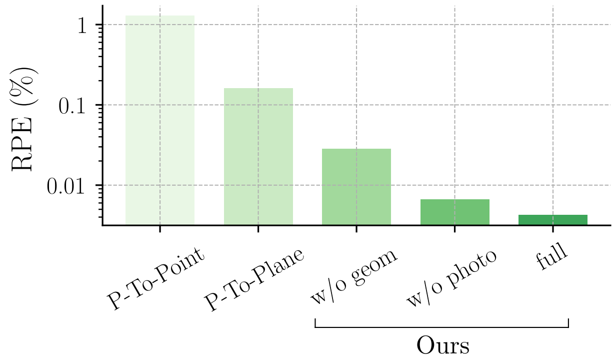

Figure 7. **Ablation on registration methods**. The plot reports the RPE (%) of several tested registration methods on the quad-easy sequence [50]. Enabling both geometric and photometric factors in sequence, provides a more robust estimate.

| Extraction Method | Acc↓ | Com↓ | C-l1↓ | F-score↑ |
|---|---|---|---|---|
| Marching Cubes [27] | 16.76 | 5.53 | 11.14 | 76.76 |
| Poisson (centers) | 10.15 | 6.70 | 8.43 | 92.33 |
| Ours | **6.64** | **4.09** | **5.37** | **96.74** |

Table 2. **Ablation on Meshing Methods.** We report mapping results with varying meshing methods on the quad-easy sequence [50].

that this method produces consistently better results than directly running TSDF integration and marching cubes over the rendered frames, or by running Poisson reconstruction directly on the Gaussian centers. Specifically, these method produces worse results in the regions that overlap multiple local models.

## 5. Conclusion

We present the first LiDAR Odometry and Mapping pipeline that leverages 2D Gaussian primitives as the sole scene representation. Through an ad-hoc tile-based Gaussian rasterizer for spherical images, we leverage LiDAR measurements to optimize the local model. Furthermore, we demonstrate the effectiveness of combining a geometric and photometric tracker to register new LiDAR point clouds over the Gaussian local model. The experiments show that our pipeline obtains tracking and mapping scores that are on par with the current SOTA at a fraction of the computational demands.

**Future Work.** We plan on improving Splat-LOAM by simultaneously estimating the sensor pose and velocity to compensate the rolling-shutter distortions caused by motion while acquisition, typical of the LiDAR measurements. Moreover, we plan on including Loop Closure (LC) to improve the pose estimates, along with the mapping accuracy.

## Acknowledgments

This work has been supported by PNRR MUR project PE0000013-FAIR.

## References

[1] Jens Behley and Cyrill Stachniss. Efficient Surfel-Based SLAM using 3D Laser Range Data in Urban Environments. In *Proc. of Robotics: Science and Systems (RSS)*. Robotics: Science and Systems Foundation, 2018. 1, 2

[2] P.J. Besl and Neil D. McKay. A method for registration of 3-D shapes. *IEEE TPAMI*, 14(2):239–256, 1992. 1, 2

[3] Dorit Borrmann, Jan Elseberg, Kai Lingemann, Andreas Nüchter, and Joachim Hertzberg. Globally consistent 3D mapping with scan matching. *Journal on Robotics and Autonomous Systems (RAS)*, 56(2):130–142, 2008. 1, 2

[4] Leonardo Brizi, Emanuele Giacomini, Luca Di Giammarino, Simone Ferrari, Omar Salem, Lorenzo De Rebotti, and Giorgio Grisetti. VBR: A Vision Benchmark in Rome. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, pages 15868–15874, 2024. 6, 7

[5] Qifeng Chen, Sheng Yang, Sicong Du, Tao Tang, Peng Chen, and Yuchi Huo. LiDAR-GS:Real-time LiDAR Re-Simulation using Gaussian Splatting, 2024. 2

[6] Pinxuan Dai, Jiamin Xu, Wenxiang Xie, Xinguo Liu, Huamin Wang, and Weiwei Xu. High-quality Surface Reconstruction using Gaussian Surfels. In *ACM SIGGRAPH 2024 Conference Papers*, pages 1–11, New York, NY, USA, 2024. Association for Computing Machinery. 2, 3

[7] Pierre Dellenbach, Jean-Emmanuel Deschaud, Bastien Jacquet, and François Goulette. CT-ICP: Real-time Elastic LiDAR Odometry with Loop Closure. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, pages 5580–5586, 2022. 1, 2

[8] Junyuan Deng, Qi Wu, Xieyuanli Chen, Songpengcheng Xia, Zhen Sun, Guoqing Liu, Wenxian Yu, and Ling Pei. NeRF-LOAM: Neural Implicit Representation for Large-Scale Incremental LiDAR Odometry and Mapping. In *ICCV*, pages 8184–8193, 2023. 1, 2

[9] Luca Di Giammarino, Leonardo Brizi, Tiziano Guadagnino, Cyrill Stachniss, and Giorgio Grisetti. MD-SLAM: Multi-cue Direct SLAM. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, pages 11047–11054, 2022. 1, 2

[10] Luca Di Giammarino, Emanuele Giacomini, Leonardo Brizi, Omar Salem, and Giorgio Grisetti. Photometric LiDAR and RGB-D Bundle Adjustment. *IEEE Robotics and Automation Letters (RA-L)*, 8(7):4362–4369, 2023. 2

[11] Chris Engels, Henrik Stewénius, and David Nistér. Bundle Adjustment Rules. *Photogrammetric computer vision*, 2(32), 2006. 6

[12] Simone Ferrari, Luca Di Giammarino, Leonardo Brizi, and Giorgio Grisetti. MAD-ICP: It is All About Matching Data – Robust and Informed LiDAR Odometry. *IEEE Robotics and Automation Letters (RA-L)*, 9(11):9175–9182, 2024. 1, 2, 6

[13] Antoine Guédon and Vincent Lepetit. SuGaR: Surface-Aligned Gaussian Splatting for Efficient 3D Mesh Reconstruction and High-Quality Mesh Rendering. In *CVPR*, pages 5354–5363, 2024. 2, 7

[14] Sheng Hong, Junjie He, Xinhu Zheng, and Chunran Zheng. LIV-GaussMap: LiDAR-Inertial-Visual Fusion for Real-Time 3D Radiance Field Map Rendering. *IEEE Robotics and Automation Letters (RA-L)*, 9(11):9765–9772, 2024. 2

[15] Binbin Huang, Zehao Yu, Anpei Chen, Andreas Geiger, and Shenghua Gao. 2D Gaussian Splatting for Geometrically Accurate Radiance Fields. In *ACM SIGGRAPH 2024 Conference Papers*, pages 1–11, New York, NY, USA, 2024. Association for Computing Machinery. 2, 3, 4, 5

[16] Seth Isaacson, Pou-Chun Kung, Mani Ramanagopal, Ram Vasudevan, and Katherine A. Skinner. LONER: LiDAR Only Neural Representations for Real-Time SLAM. *IEEE Robotics and Automation Letters (RA-L)*, 8(12):8042–8049, 2023. 1, 2

[17] Junzhe Jiang, Chun Gu, Yurui Chen, and Li Zhang. GS-LiDAR: Generating Realistic LiDAR Point Clouds with Panoramic Gaussian Splatting. In *ICLR*, 2025. 2

[18] Michael Kazhdan, Matthew Bolitho, and Hugues Hoppe. Poisson surface reconstruction. In *Proceedings of the Fourth Eurographics Symposium on Geometry Processing*, pages 61–70, Goslar, DEU, 2006. Eurographics Association. 7

[19] Nikhil Keetha, Jay Karhade, Krishna Murthy Jatavallabhula, Gengshan Yang, Sebastian Scherer, Deva Ramanan, and Jonathon Luiten. Splatam: Splat, track and map 3d gaussians for dense rgb-d slam. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024. 2

[20] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkuehler, and George Drettakis. 3D Gaussian Splatting for Real-Time Radiance Field Rendering. *ACM TOG*, 42(4):139:1–139:14, 2023. 1, 4

[21] Qing Li, Shaoyang Chen, Cheng Wang, Xin Li, Chenglu Wen, Ming Cheng, and Jonathan Li. LO-Net: Deep Real-Time Lidar Odometry. In *CVPR*, pages 8465–8474, 2019. 2

[22] Lorenzo Liso, Erik Sandström, Vladimir Yugay, Luc Van Gool, and Martin R. Oswald. Loopy-SLAM: Dense Neural SLAM with Loop Closures. In *CVPR*, pages 20363–20373, 2024. 1, 2, 5

[23] Hidenobu Matsuki, Riku Murai, Paul HJ Kelly, and Andrew J Davison. Gaussian splatting slam. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18039–18048, 2024. 2

[24] Hidenobu Matsuki, Riku Murai, Paul H. J. Kelly, and Andrew J. Davison. Gaussian Splatting SLAM. In *CVPR*, pages 18039–18048, 2024. 2, 5

[25] Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy networks: Learning 3d reconstruction in function space. In *CVPR*, 2019. 7

[26] 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. In *ECCV*, pages 405–421, Cham, 2020. Springer International Publishing. 1

[27] Ken Museth. VDB: High-resolution sparse volumes with dynamic topology. *ACM TOG*, 32(3):27:1–27:22, 2013. 6, 7, 8

[28] Austin Nicolai, Ryan Skeele, Christopher Eriksen, and Geoffrey A. Hollinger. Deep Learning for Laser Based Odometry

Estimation. In *RSS workshop Limits and Potentials of Deep Learning in Robotics*, page 1, 2016. 2

[29] Helen Oleynikova, Zachary Taylor, Marius Fehr, Roland Siegwart, and Juan Nieto. Voxblox: Incremental 3D Euclidean Signed Distance Fields for on-board MAV planning. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, pages 1366–1373, 2017. 6, 7

[30] Yue Pan, Xingguang Zhong, Louis Wiesmann, Thorbjörn Posewsky, Jens Behley, and Cyrill Stachniss. PIN-SLAM: LiDAR SLAM Using a Point-Based Implicit Neural Representation for Achieving Global Map Consistency. *IEEE Trans. on Robotics*, 40:4045–4064, 2024. 1, 2, 6, 7

[31] Jan Quenzel and Sven Behnke. Real-time Multi-Adaptive-Resolution-Surfel 6D LiDAR Odometry using Continuous-time Trajectory Optimization. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, pages 5499–5506, 2021. 1, 2

[32] Jianyuan Ruan, Bo Li, Yibo Wang, and Yuxiang Sun. SLAMesh: Real-time LiDAR Simultaneous Localization and Meshing. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, pages 3546–3552, 2023. 1, 2, 6, 4

[33] Erik Sandström, Yue Li, Luc Van Gool, and Martin R. Oswald. Point-SLAM: Dense Neural Point Cloud-based SLAM. In *ICCV*, pages 18387–18398, 2023. 1, 2

[34] Erik Sandström, Keisuke Tateno, Michael Oechsle, Michael Niemeyer, Luc Van Gool, Martin R. Oswald, and Federico Tombari. Splat-SLAM: Globally Optimized RGB-only SLAM with 3D Gaussians, 2024. 2, 5

[35] Johannes L. Schönberger and Jan-Michael Frahm. Structure-from-Motion Revisited. In *CVPR*, pages 4104–4113, 2016. 1

[36] Johannes L. Schönberger, Enliang Zheng, Jan-Michael Frahm, and Marc Pollefeys. Pixelwise View Selection for Unstructured Multi-View Stereo. In *ECCV*, pages 501–518, Cham, 2016. Springer International Publishing. 1

[37] A. Segal, D. Haehnel, and S. Thrun. Generalized-ICP. In *Proc. of Robotics: Science and Systems (RSS)*. Robotics: Science and Systems Foundation, 2009. 1, 2

[38] Tixiao Shan and Brendan Englot. LeGO-LOAM: Lightweight and Ground-Optimized Lidar Odometry and Mapping on Variable Terrain. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, pages 4758–4765, 2018. 2

[39] Shuangfu Song, Junqiao Zhao, Kai Huang, Jiaye Lin, Chen Ye, and Tiantian Feng. N$^3$-Mapping: Normal Guided Neural Non-Projective Signed Distance Fields for Large-Scale 3D Mapping. *IEEE Robotics and Automation Letters (RA-L)*, 9 (6):5935–5942, 2024. 1, 2, 6, 7

[40] Edgar Sucar, Shikun Liu, Joseph Ortiz, and Andrew J. Davison. iMAP: Implicit Mapping and Positioning in Real-Time. In *ICCV*, pages 6209–6218, 2021. 1

[41] Yifu Tao, Miguel Ángel Muñoz-Bañón, Lintong Zhang, Jiahao Wang, Lanke Frank Tarimo Fu, and Maurice Fallon. The Oxford Spires Dataset: Benchmarking Large-Scale LiDAR-Visual Localisation, Reconstruction and Radiance Field Methods, 2024. 6, 7, 3, 4, 5

[42] Fabio Tosi, Youmin Zhang, Ziren Gong, Erik Sandström, Stefano Mattoccia, Martin R. Oswald, and Matteo Poggi. How nerfs and 3d gaussian splatting are reshaping slam: a survey, 2024. 2

[43] Ignacio Vizzo, Xieyuanli Chen, Nived Chebrolu, Jens Behley, and Cyrill Stachniss. Poisson Surface Reconstruction for LiDAR Odometry and Mapping. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, pages 5624–5630, 2021. 6, 7, 3

[44] Guangming Wang, Xinrui Wu, Zhe Liu, and Hesheng Wang. PWCLO-Net: Deep LiDAR Odometry in 3D Point Clouds Using Hierarchical Embedding Mask Optimization. In *CVPR*, pages 15905–15914, 2021. 2

[45] Tim Weyrich, Simon Heinzle, Timo Aila, Daniel B. Fasnacht, Stephan Oetiker, Mario Botsch, Cyril Flaig, Simon Mall, Kaspar Rohrer, Norbert Felber, Hubert Kaeslin, and Markus Gross. A hardware architecture for surface splatting. *ACM TOG*, 26(3):90–es, 2007. 4

[46] Chenyang Wu, Yifan Duan, Xinran Zhang, Yu Sheng, Jianmin Ji, and Yanyong Zhang. MM-Gaussian: 3D Gaussian-based Multi-modal Fusion for Localization and Reconstruction in Unbounded Scenes. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, pages 12287–12293, 2024. 2

[47] Vladimir Yugay, Yue Li, Theo Gevers, and Martin R. Oswald. Gaussian-SLAM: Photo-realistic Dense SLAM with Gaussian Splatting, 2024. 2, 5

[48] Ganlin Zhang, Erik Sandström, Youmin Zhang, Manthan Patel, Luc Van Gool, and Martin R Oswald. Glorie-slam: Globally optimized rgb-only implicit encoding point cloud slam. *arXiv preprint arXiv:2403.19549*, 2024. 2

[49] Ji Zhang and Sanjiv Singh. LOAM: Lidar Odometry and Mapping in Real-time. In *Proc. of Robotics: Science and Systems (RSS)*. Robotics: Science and Systems Foundation, 2014. 2, 6, 7

[50] Lintong Zhang, Marco Camurri, David Wisth, and Maurice Fallon. Multi-Camera LiDAR Inertial Extension to the Newer College Dataset, 2021. 6, 7, 8, 3

[51] Xin Zheng and Jianke Zhu. Efficient LiDAR Odometry for Autonomous Driving. *IEEE Robotics and Automation Letters (RA-L)*, 6(4):8458–8465, 2021. 2

[52] Xingguang Zhong, Yue Pan, Jens Behley, and Cyrill Stachniss. SHINE-Mapping: Large-Scale 3D Mapping Using Sparse Hierarchical Implicit Neural Representations. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, pages 8371–8377, 2023. 1, 2

[53] Liyuan Zhu, Yue Li, Erik Sandström, Shengyu Huang, Konrad Schindler, and Iro Armeni. LoopSplat: Loop Closure by Registering 3D Gaussian Splats. In *Proc. of the International Conference on 3D Vision (3DV)*, 2025. 2, 5

[54] Zihan Zhu, Songyou Peng, Viktor Larsson, Weiwei Xu, Hujun Bao, Zhaopeng Cui, Martin R. Oswald, and Marc Pollefeys. NICE-SLAM: Neural Implicit Scalable Encoding for SLAM. In *CVPR*, pages 12776–12786, 2022. 1, 2