PINGS: Gaussian Splatting Meets Distance Fields within a Point-Based Implicit Neural Map

Yue Pan* Xingguang Zhong* Liren Jin* Louis Wiesmann*
Marija Popović‡ Jens Behley* Cyrill Stachniss*,†

* Center for Robotics, University of Bonn, Germany
‡ MAVLab, TU Delft, the Netherlands
† Lamarr Institute for Machine Learning and Artificial Intelligence, Germany

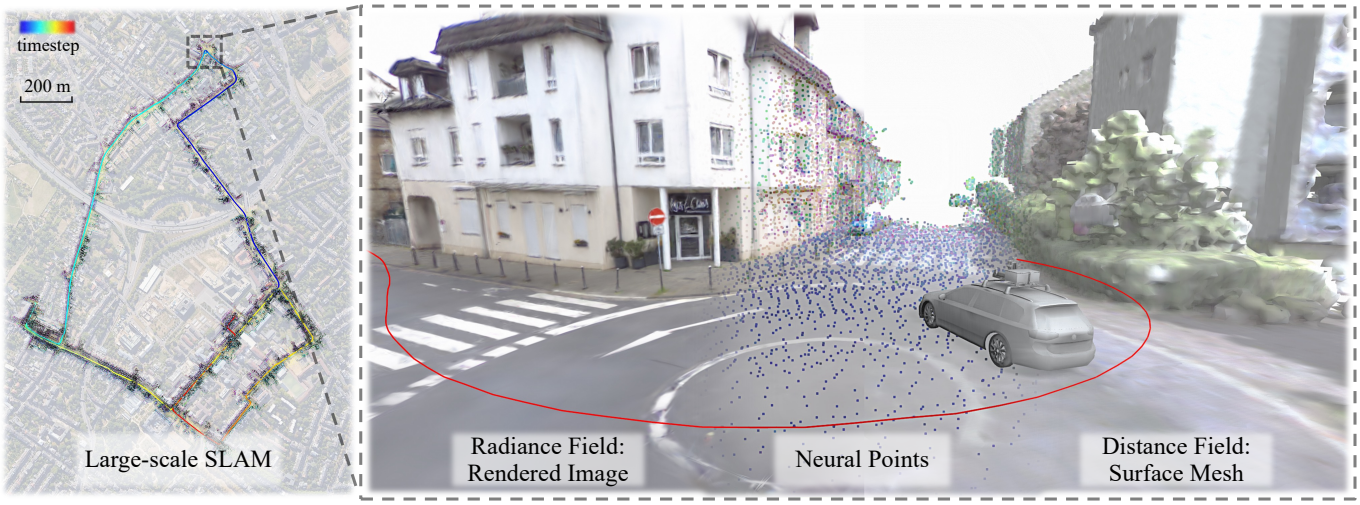


Fig. 1: We present PINGS, a novel LiDAR-visual SLAM system unifying distance field and radiance field mapping using an elastic point-based implicit neural representation. On the left, we show a globally consistent neural point map overlaid on a satellite image. The map was built using PINGS from around 10,000 LiDAR scans and 40,000 images collected by a robot car driving in an urban environment for around 5 km. The estimated trajectory is overlaid on the map and colorized according to the timestep. On the right, we show a zoomed-in view of a roundabout mapped by PINGS. It illustrates from left to right the rendered image from the Gaussian splatting radiance field, neural points colorized by the principal components of their geometric features, and the reconstructed mesh from the distance field (colorized by the radiance field). The red line indicates the local trajectory of the robot car (shown as the CAD model).

***Abstract*—Robots benefit from high-fidelity reconstructions of their environment, which should be geometrically accurate and photorealistic to support downstream tasks. While this can be achieved by building distance fields from range sensors and radiance fields from cameras, realising scalable incremental mapping of both fields consistently and at the same time with high quality is challenging. In this paper, we propose a novel map representation that unifies a continuous signed distance field and a Gaussian splatting radiance field within an elastic and compact point-based implicit neural map. By enforcing geometric consistency between these fields, we achieve mutual improvements by exploiting both modalities. We present a novel LiDAR-visual SLAM system called PINGS using the proposed map representation and evaluate it on several challenging large-scale datasets. Experimental results demonstrate that PINGS can incrementally build globally consistent distance and radiance fields encoded with a compact set of neural points. Compared to state-of-the-art methods, PINGS achieves superior photometric and geometric rendering at novel views by constraining the radiance field with the distance field. Furthermore, by utilizing dense photometric cues and multi-view consistency from the radiance field, PINGS produces more accurate distance fields, leading to improved odometry estimation and mesh reconstruction. We also provide an open-source implementation of PINGS.**

I. Introduction

The ability to perceive and understand the surroundings is fundamental for autonomous robots. At the core of this capability lies the ability to build a map — a digital twin of the robot's workspace that is ideally both geometrically accurate and photorealistic, enabling effective spatial awareness and operation of the robot [24, 41].

Previous works in robotics mainly focus on the incremental mapping of an occupancy grid or a distance field using range sensors, such as LiDAR or depth cameras, which enable localization [16], collision avoidance [14], or exploration [59]. Recently, PIN-SLAM [51] demonstrated that a compact point-based implicit neural representation can effectively model a continuous signed distance field (SDF) for LiDAR simultaneous localization and mapping (SLAM), enabling both accurate localization and globally consistent mapping.

However, occupancy voxel grids [16], occupancy fields [91], or distance fields [49, 51] fall short of providing photorealistic novel view rendering of the scene, which is crucial for applications requiring dense photometric information. This capability

can be achieved by building an additional radiance field with visual data using representations such as neural radiance field (NeRF) [45] or a 3D Gaussian splatting (3DGS) model [30]. Recent works demonstrated the potential of radiance fields, especially 3DGS, for various robotic applications including human-robot interaction [43], scene understanding [92, 97], simulation or world models for robotics learning [2, 12, 81], visual localization [4, 42], and active reconstruction [26, 27]. Nevertheless, these approaches often assume well-captured image collections in bounded scenes with offline processing, limiting their applicability for mobile robotic applications. Besides, radiance fields are not necessarily geometrically accurate, which can lead to issues in localization or planning.

In this paper, we investigate how to simultaneously build consistent, geometrically accurate, and photorealistic radiance fields as well as accurate distance fields for large-scale environments using LiDAR and camera data. Building upon PIN-SLAM's [51] point-based neural map for distance fields and inspired by Scaffold-GS [38], we propose a novel point-based model that additionally represents a Gaussian splatting radiance field. By enforcing mutual supervision between these fields during incremental mapping, we achieve both improved rendering quality from the radiance field and more accurate distance field for better localization and surface reconstruction.

The main contribution of this paper is a novel LiDAR-visual SLAM system, called PINGS, that incrementally builds continuous SDF and Gaussian splatting radiance fields by exploiting their mutual consistency within a point-based neural map. The distance field and radiance field infered from the elastic neural points enable robust pose estimation while maintaining global consistency through loop closure correction. The compact neural point map can be efficiently stored and loaded from disk, allowing accurate surface mesh reconstruction from the distance field and high-fidelity real-time novel view rendering from the radiance field, as shown in Fig. 1.

In sum, we make four key claims: (i) PINGS achieves better RGB and geometric rendering at novel views by constraining the Gaussian splatting radiance field using the signed distance field; (ii) PINGS builds a more accurate signed distance field for more accurate localization and surface reconstruction by leveraging dense photometric cues from the radiance field; (iii) PINGS enables large-scale globally consistent mapping with loop closures; (iv) PINGS builds a more compact map than previous methods for both radiance and distance fields.

Our open-source implementation of PINGS is publicly available at: https://github.com/PRBonn/PINGS.

II. Related Work

A. Point-based Implicit Neural Representation

Robotics has long relied on explicit map representations with discrete primitives like point clouds [87], surfels [3, 73], meshes [66], or voxel grids [22, 46] for core tasks like localization [65] and planning [59].

Recently, implicit neural representations have been proposed to model radiance fields [45] and geometric (occupancy or distance) fields [44, 49, 52] using multi-layer perceptrons (MLP). These continuous representations offer advantages like compact storage, and better handling of regions with sparse observations or occlusions, while supporting conversion to explicit representations for downstream tasks.

Instead of using a single MLP for the entire scene, recent methods use hybrid representations that combine local feature vectors with a shared shallow MLP. Point-based implicit neural representations [51, 79] store optimizable features in a neural point cloud, which has advantages over grid-based alternatives through its flexible spatial layout and inherent elasticity under transformations for example caused by loop closures.

Point-based implicit neural representations have been used for modeling either radiance fields or distance fields for various applications including differentiable rendering [8, 79], dynamic scene modeling [1], surface reconstruction [34], visual odometry [56, 86], and globally consistent mapping [51]. For example, PIN-SLAM [51] effectively represents local distance fields with neural points for odometry estimation and uses the elasticity of these neural points during loop closure correction.

In this paper, we propose a novel LiDAR-visual SLAM system that is built on top of PIN-SLAM [51] and encodes a Gaussian splatting radiance field within neural points while jointly optimizing it alongside the distance field. Compared to NeRF-based approaches [8, 79], this offers faster novel view rendering suitable for robotics applications.

B. Gaussian Splatting Radiance Field

NeRF [45] pioneered the use of MLPs to map 3D positions and view directions to color and volume density, encoding radiance fields through volume rendering-based training with posed RGB images. More recently, 3DGS [30] introduced explicit 3D Gaussian primitives to represent the radiance fields, achieving high-quality novel view synthesis. Compared to NeRF-based methods, 3DGS is more efficient by using primitive-based differentiable rasterization [82] instead of ray-wise volume rendering. The explicit primitives also enables editing and manipulation of the radiance field. These properties make 3DGS promising for robotics applications [2, 26, 35, 42, 43]. However, two main challenges limit its usage: geometric accuracy and scalability for incremental mapping. We discuss the related works addressing geometric accuracy in the following and addressing scalable mapping in Sec. II-C.

While 3DGS achieves high-fidelity photorealistic rendering, it often lacks the geometric accuracy. To tackle this limitation, SuGaR [18] uses a hybrid representation to extract meshes from 3DGS and align the Gaussian primitives with the surface meshes. To address the ambiguity in surface description, another solution is to flatten the 3D Gaussian ellipsoids to 2D disks [11, 23, 25, 85]. The 2D disks gradually align with surfaces during training, enabling more accurate depth and normal rendering. However, extracting surface meshes from these discrete primitives still requires either TSDF fusion [46] with rendered depth or Poisson surface reconstruction [28].

Another line of works [58, 84] model discrete Gaussian opacity as a continuous field, similar to NeRF-based surface reconstruction [70]. Several works [6, 39, 83] jointly train a

distance field with 3DGS and align the Gaussian primitives with the zero-level set of the distance field to achieve accurate surface reconstruction. However, these methods rely solely on image rendering supervision for both 3DGS and neural SDF training without direct 3D geometric constraints, leading to ambiguities in textureless or specular regions. The volume rendering-based SDF training also impacts efficiency.

While 3DGS originally uses structure-from-motion point clouds, robotic platforms with LiDAR can initialize primitives directly from LiDAR measurements [10, 21, 78]. Direct depth measurements can further supervise depth rendering to improve geometric accuracy and convergence speed [25, 42].

Our approach uniquely combines geometrically consistent 2D Gaussian disks with a neural distance field supervised by direct LiDAR measurements, enforcing mutual geometric consistency between the representations. This differs from GS-Fusion [72], which maintains decoupled distance and radiance fields without mutual supervision.

C. Large-Scale 3D Reconstruction

This paper focuses on online large-scale 3D reconstruction. There have been numerous works for the scalable occupancy or distance field mapping in the past decade, using efficient data structures such as an Octree [22, 93], voxel hashing [33, 48, 94], an VDB [67, 77], or wavelets [53].

Scalable radiance field mapping has also made significant progress recently. For large scale scenes captured by aerial images, recent works [36, 38, 55] demonstrate promising results using level-of-detail rendering and neural Gaussian compression. For driving scenes with short sequences containing hundreds of images, both NeRF-based [54, 81] and 3DGS-based [9, 13, 19, 80, 90, 95] approaches have demonstrated high-fidelity offline radiance field reconstruction, enabling closed-loop autonomous driving simulation [9, 81].

However, radiance field mapping for even larger scenes at ground level with thousands of images remains challenging due to scene complexity and memory constraints. BlockNeRF [61] addresses this by dividing scenes into overlapping blocks, training separate NeRFs per block, and consolidating them during rendering. Similarly, SiLVR [63] employs a submap strategy for scalable NeRF mapping. For 3DGS, hierarchical 3DGS [31] introduces a level-of-detail hierarchy that enables real-time rendering of city-scale scenes. The aforementioned methods require time-consuming structure-from-motion preprocessing and offline divide-and-conquer processing, limiting their applicability for online missions.

While there are several works on incremental mapping and SLAM with NeRF [49, 56, 60] or 3DGS [29, 42, 72, 96], they primarily focus on bounded indoor scenes and struggle with our target scenarios. Our proposed system enables incremental radiance and distance field mapping at the scale of previous offline methods [31, 61], while achieving globally consistent 3D reconstruction through loop closure correction.

III. Our Approach

Our approach, called PINGS, is a LiDAR-visual SLAM system that jointly builds globally consistent Gaussian splatting radiance fields and distance fields for large-scale scenes.

Notation. In the following, we denote the transformation from coordinate frame A to frame B as $\mathsf{T}_{BA} \in \mathrm{SE}(3)$, such that point $\boldsymbol{p}_B = \mathsf{T}_{BA}\boldsymbol{p}_A$, with rotation $\boldsymbol{R}_{BA} \in \mathrm{SO}(3)$ and translation $\boldsymbol{t}_{BA} \in \mathbb{R}^3$, where the rotation is also parameterized by a unit quaternion $\boldsymbol{q}$. At timestep t, each sensor frame S_t (LiDAR frame L_t or camera frame C_t) is related to the world frame W by pose T_{WS_t}, with T_{WS_0} fixed as identity. We denote the rotation of a vector $\boldsymbol{v} \in \mathbb{R}^3$ by a quaternion $\boldsymbol{q}$ as $\boldsymbol{q}\boldsymbol{v}\boldsymbol{q}^{-1}$ and the multiplication of two quaternions as $\boldsymbol{q}_1\boldsymbol{q}_2$.

Overview. We assume the robot is equipped with a LiDAR sensor and one or multiple cameras. At each timestep t, the input to our system is a LiDAR point cloud $\mathcal{P} = \{\boldsymbol{p} \in \mathbb{R}^3\}$ and M camera images $\mathcal{I} = \{\hat{I}_i \in \mathbb{R}^{H\times W\times 3} \mid i = 1, \dots, M\}$ collected by the robot. We assume the calibration of the LiDAR and cameras to be known but allow for the imperfect synchronization among the sensors. Our system aims to simultaneously estimate the LiDAR pose T_{WL_t} while updating a point-based implicit neural map $\mathcal{M}$, which models both a SDF and a radiance field, as summarized in Fig. 2.

A. Point-based Implicit Neural Map Representation

We define our point-based implicit neural map $\mathcal{M}$ as a set of neural points, given by:

$$\mathcal{M} = \{\boldsymbol{m}_i = (\boldsymbol{x}_i, \boldsymbol{q}_i, \boldsymbol{f}_i^g, \boldsymbol{f}_i^a, \tau_i^c, \tau_i^u) \mid i = 1, \dots, N\}, \quad (1)$$

where each neural point $\boldsymbol{m}_i$ is defined in the world frame W by a position $\boldsymbol{x}_i \in \mathbb{R}^3$ and a quaternion $\boldsymbol{q}_i \in \mathbb{R}^4$ representing the orientation of its own coordinate frame. Each neural point stores the optimizable geometric feature vector $\boldsymbol{f}_i^g \in \mathbb{R}^{F_g}$ and appearance feature vector $\boldsymbol{f}_i^a \in \mathbb{R}^{F_a}$. In addition, we keep track of each neural point's creation timestep τ_i^c and last update timestep τ_i^u to determine its active status and associate the neural point with the LiDAR pose T_{WL_τ} at the middle timestep $\tau_i = \lfloor(\tau_i^c + \tau_i^u)/2\rfloor$ between τ_i^c and τ_i^u, thus allowing direct map manipulation through pose updates.

We maintain a voxel hashing [47] data structure $\mathcal{V}$ with a voxel resolution v_p for fast neural point indexing and neighbor search, where each voxel stores at most one active neural point.

During incremental mapping, we dynamically update the neural point map based on point cloud measurements. For each newly measured point $\boldsymbol{p}_W$ in the world frame, we check its corresponding voxel in $\mathcal{V}$. If no active neural point exists in that voxel, we initialize a new neural point $\boldsymbol{m}$ with the position $\boldsymbol{x} = \boldsymbol{p}_W$, an identity quaternion $\boldsymbol{q} = (1, 0, 0, 0)$, and the feature vectors $\boldsymbol{f}^g = \boldsymbol{0},\ \boldsymbol{f}^a = \boldsymbol{0}$. Additionally, we define a local map $\mathcal{M}_l$ centered at the current LiDAR position $\boldsymbol{t}_{WL_t}$, which contains all active neural points within radius r_l. To avoid incorporating inconsistent historical observations caused by odometry drift, both map optimization and odometry estimation are operated only within this local map $\mathcal{M}_l$. After the map optimization at each timestep, we reassign the local map $\mathcal{M}_l$ into the global map $\mathcal{M}$.

Next, we describe how the neural points map $\mathcal{M}$ models both the SDF (Sec. III-B) and the radiance field (Sec. III-C).

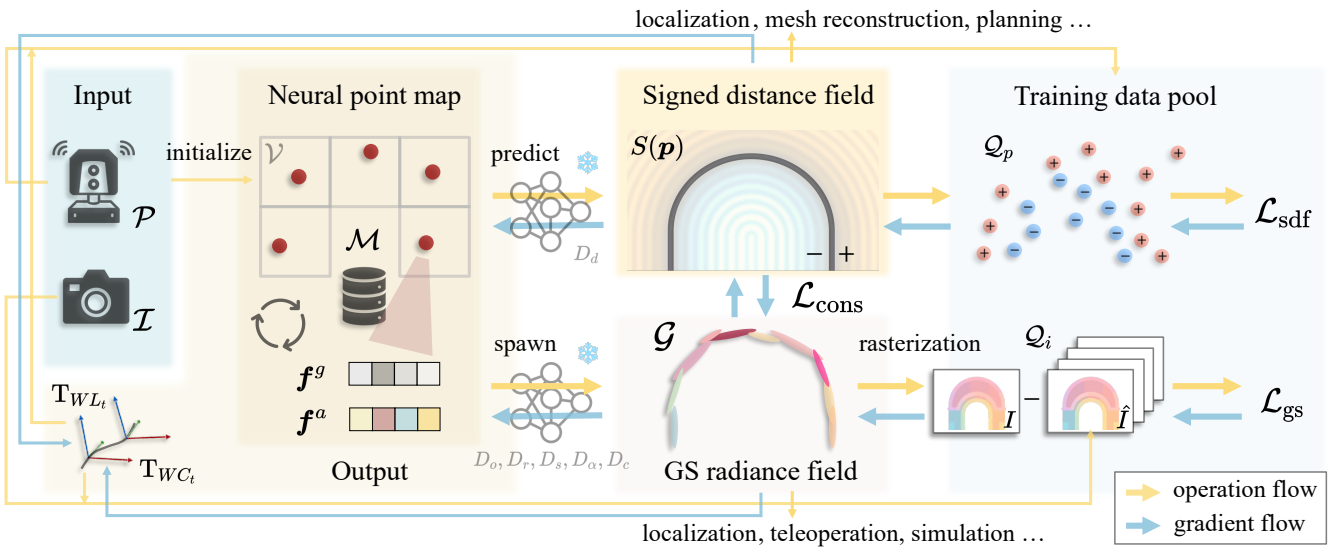


Fig. 2: Overview of PINGS: We take a stream of LiDAR point clouds $\mathcal{P}$ and camera images $\mathcal{I}$ as input. We initialize a neural point map $\mathcal{M}$ from $\mathcal{P}$ and maintain a training pool of SDF-labeled points $\mathcal{Q}_p$ and recent images $\mathcal{Q}_i$. The map uses a voxel hashing structure $\mathcal{V}$ where each neural point stores geometric features $\boldsymbol{f}^g$ and appearance features $\boldsymbol{f}^a$. These features are used to predict SDF values $S(\boldsymbol{p})$ at an arbitrary position $\boldsymbol{p}$ and spawn Gaussian primitives $\mathcal{G}$ through MLP decoders. We compute three kind of losses: (1) Gaussian splatting loss $\mathcal{L}_{\text{gs}}$ comparing rendered images through differentiable rasterization and reference images in the training pool, (2) SDF loss $\mathcal{L}_{\text{sdf}}$ comparing predicted SDF and labels of the sampled points in the training pool, and (3) consistency loss $\mathcal{L}_{\text{cons}}$ to align the geometry of both representations. The losses are backpropagated to optimize the neural point features $\boldsymbol{f}^g$ and $\boldsymbol{f}^a$. Meanwhile, we estimate LiDAR odometry by aligning the point cloud to current SDF and backpropagate $\mathcal{L}_{\text{gs}}$ to refine the camera poses. The final outputs are LiDAR poses T_{WL_t}, camera poses T_{WC_t}, and a compact neural point map $\mathcal{M}$ representing both SDF and Gaussian splatting radiance fields, enabling various robotic applications.

B. Neural Signed Distance Field

For the modeling and online training of a continuous SDF using the neural points, we follow the same strategy as in PIN-SLAM [51] and present a recap in this section.

We model the SDF value s at a query position $\boldsymbol{p}$ in the world frame W conditioned on its nearby neural points. For each neural point $\boldsymbol{m}_j$ in the k-nearest neighborhood $\mathcal{N}_p$ of $\boldsymbol{p}$, we define the relative coordinate $\boldsymbol{d}_j = \boldsymbol{q}_j(\boldsymbol{p}-\boldsymbol{x}_j)\boldsymbol{q}_j^{-1}$ denoting $\boldsymbol{p}$ in the local coordinate system of $\boldsymbol{m}_j$. Then, we feed the geometric feature vector $\boldsymbol{f}_j^g$ and the relative coordinate $\boldsymbol{d}_j$ to a globally shared SDF decoder D_d to predict the SDF s_j:

$$s_j = D_d(\boldsymbol{f}_j^g, \boldsymbol{d}_j). \tag{2}$$

As shown in Fig. 3 (a), the predicted SDF values s_j of the neighboring neural points at the query position $\boldsymbol{p}$ are then interpolated as the final prediction $s = S(\boldsymbol{p})$, given by:

$$S(\boldsymbol{p}) = \sum_{j \in \mathcal{N}_p} \frac{w_j}{\sum_{k \in \mathcal{N}_p} w_k} s_j, \tag{3}$$

with the interpolation weights $w_j = \|\boldsymbol{p} - \boldsymbol{x}_j\|^{-2}$.

To optimize the neural SDF represented by the neural point geometric features $\{\boldsymbol{f}_i^g\}_{i=1}^N$ and the SDF decoder D_d, we sample points along the LiDAR rays around the measured end points and in the free space. We take the projective signed distance along the ray as a pseudo SDF label for each sample point. For incremental learning, we maintain a training data pool $\mathcal{Q}_p$ containing sampled points from recent scans, with a maximum capacity and bounded by a distance threshold from the current robot position. At each timestep, we sample from the training data pool in batches and predict the SDF value at the sample positions. The SDF training loss $\mathcal{L}_{\text{sdf}}$ is formulated as a weighted sum of the binary cross entropy loss term $\mathcal{L}_{\text{bce}}$ and the Eikonal loss term $\mathcal{L}_{\text{eik}}$, given by:

$$\mathcal{L}_{\text{sdf}} = \lambda_{\text{bce}}\mathcal{L}_{\text{bce}} + \lambda_{\text{eik}}\mathcal{L}_{\text{eik}}. \tag{4}$$

The loss term $\mathcal{L}_{\text{bce}}$ applies a soft supervision on the SDF values by comparing the sigmoid activation of both the predictions and the pseudo labels. The Eikonal loss term $\mathcal{L}_{\text{eik}}$ regularizes the SDF gradients by enforcing the Eikonal constraint [17], which requires unit-length gradients $\|\nabla S(\boldsymbol{x})\| = 1$ for the sampled points. For more details regarding the SDF training, we refer readers to Pan et al. [51].

The incrementally built neural SDF map can then be used for LiDAR odometry estimation and surface mesh extraction.

C. Neural Gaussian Splatting Radiance Field

We use camera image streams $\mathcal{I}$ to construct a radiance field by spawning Gaussian primitives from our neural point map $\mathcal{M}$ and optimizing $\mathcal{M}$ via differentiable rasterization.

Neural Point-based Gaussian Spawning. Inspired by Scaffold-GS [38], we use our neural points as anchor points for spawning Gaussian primitives, see Fig. 3 (b). For each neural point $\boldsymbol{m}$ lying within the current camera frustum, we spawn K Gaussian primitives by feeding its feature vectors $(\boldsymbol{f}^g,\ \boldsymbol{f}^a)$ through globally shared MLP decoders. We parameterize each spawned Gaussian primitive $\boldsymbol{g}$ with its position $\boldsymbol{\mu} \in \mathbb{R}^3$ in the world frame, rotation $\boldsymbol{r} \in \mathbb{R}^4$ in the form of a unit quaternion, scale $\boldsymbol{s} \in \mathbb{R}^3$, opacity $\alpha \in [-1\,, 1]$, and RGB color $\boldsymbol{c} \in [0\,, 1]^3$.

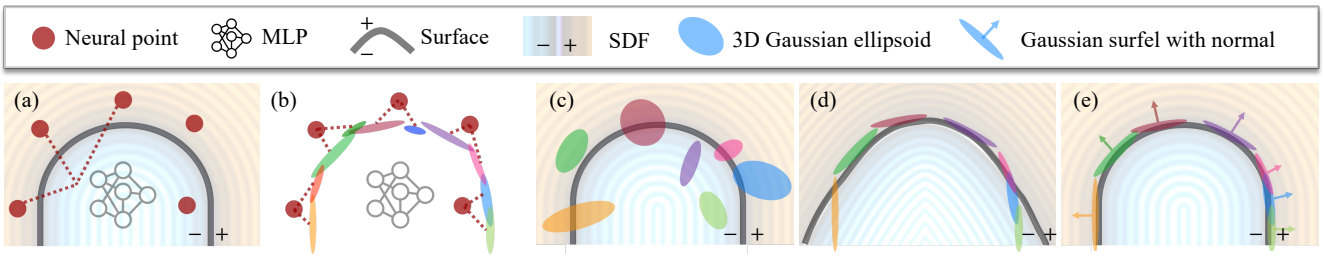


Fig. 3: Example of neural point-based SDF prediction, Gaussian primitives spawning, and the geometric consistency of PINGS: (a) SDF prediction at a query point through weighted interpolation of predictions from neighboring neural points. (b) Neural points spawning multiple Gaussian primitives to compose the radiance field. (c) Example of an accurate SDF but geometrically inaccurate radiance field with 3D Gaussian ellipsoids in regions with dense LiDAR coverage but sparse camera views, weak texture, or poor lighting. (d) Example of a geometrically accurate radiance field but inaccurate SDF in regions with rich visual data but sparse LiDAR measurements. (e) Our solution: flattening 3D Gaussian ellipsoids to surfels and enforcing geometric consistency by aligning surfel centers with the SDF zero-level set and aligning surfel normals with SDF gradients, resulting in accurate geometry for both fields.

Each neural point spawns Gaussian primitives in its local coordinate frame defined by its position $\boldsymbol{x}$ and orientation $\boldsymbol{q}$. The world-frame position $\boldsymbol{\mu}_i$ of each spawned primitive is:

$$\{\boldsymbol{\mu}_i = \boldsymbol{q}\boldsymbol{o}_i\boldsymbol{q}^{-1} + \boldsymbol{x} \mid \boldsymbol{o}_i \in D_o(\boldsymbol{f}^g)\}_{i=1}^{K}, \tag{5}$$

where D_o is the offset decoder that maps the geometric feature $\boldsymbol{f}^g$ to a set of K local offsets $\{\boldsymbol{o}_i\}_{i=1}^{K}$, which are then transformed into the world frame through quaternion rotation and translation. Likewise, the rotation $\boldsymbol{r}_i$ of each spawned Gaussian primitive is predicted by the rotation decoder D_r and then rotated by quaternion $\boldsymbol{q}$ as:

$$\{\boldsymbol{r}_i = \boldsymbol{q}\hat{\boldsymbol{r}}_i \mid \hat{\boldsymbol{r}}_i \in D_r(\boldsymbol{f}^g)\}_{i=1}^{K}. \tag{6}$$

The scale decoder D_s predicts each primitive's scale $\boldsymbol{s}_i$ as:

$$\{\boldsymbol{s}_i\}_{i=1}^{K} = D_s(\boldsymbol{f}^g). \tag{7}$$

We predict opacity values α in the range $[-1, 1]$ and treat only Gaussian primitives with positive opacity as being valid. To adaptively control spatial density of Gaussian primitives based on viewing distance, we feed the geometric feature $\boldsymbol{f}^g$ and the view distance $\delta_v = \|\boldsymbol{x} - \boldsymbol{t}_{WC}\|_2$ into the opacity decoder D_α. This implicitly encourages the network to predict fewer valid Gaussians for distant points and more for nearby points, reducing computational load. The opacity value α_i for each Gaussian primitive is predicted as:

$$\{\alpha_i\}_{i=1}^{K} = D_\alpha(\boldsymbol{f}^g, \delta_v). \tag{8}$$

For view-dependent color prediction, we take a different approach than the spherical harmonics used in 3DGS [30]. We feed the appearance feature $\boldsymbol{f}^a$ and the view direction $\boldsymbol{d}_v = (\boldsymbol{x} - \boldsymbol{t}_{WC})/\delta_v$ to the color decoder D_c to predict the color $\boldsymbol{c}_i$ of each Gaussian primitive, given by:

$$\{\boldsymbol{c}_i\}_{i=1}^{K} = D_c(\boldsymbol{f}^a, \boldsymbol{q}^{-1}\boldsymbol{d}_v\boldsymbol{q}), \tag{9}$$

where the view direction $\boldsymbol{d}_v$ is also transformed into the local coordinate system of the neural point.

Note that we treat position, rotation, scale, and opacity as geometric attributes of a Gaussian primitive, using the geometric feature $\boldsymbol{f}^g$ for their prediction, while using the appearance feature $\boldsymbol{f}^a$ to predict color.

Gaussian Splatting Rasterization. We gather all the valid Gaussians primitives $\mathcal{G}$ [30] spawned at the current viewpoint:

$$\mathcal{G} = \{\boldsymbol{g}_i = (\boldsymbol{\mu}_i, \boldsymbol{r}_i, \boldsymbol{s}_i, \boldsymbol{c}_i, \alpha_i) \mid i = 1, \ldots, N_g\}. \tag{10}$$

The distribution of each Gaussian primitive $\boldsymbol{g}_i$ in the world frame is represented as:

$$\mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i) = \exp\left(-\frac{1}{2}(\boldsymbol{x} - \boldsymbol{\mu}_i)^\top \boldsymbol{\Sigma}_i^{-1}(\boldsymbol{x} - \boldsymbol{\mu}_i)\right), \tag{11}$$

where the covariance matrix $\boldsymbol{\Sigma}_i$ is reparameterized as:

$$\boldsymbol{\Sigma}_i = \boldsymbol{R}(\boldsymbol{r}_i)\, \boldsymbol{S}(\boldsymbol{s}_i)\, \boldsymbol{S}(\boldsymbol{s}_i)^\top\, \boldsymbol{R}(\boldsymbol{r}_i)^\top, \tag{12}$$

where $\boldsymbol{R}(\boldsymbol{r}_i) \in SO(3)$ is the rotation matrix derived from the quaternion $\boldsymbol{r}_i$ and $\boldsymbol{S}(\boldsymbol{s}_i) = \mathrm{diag}(\boldsymbol{s}_i) \in \mathbb{R}^{3\times3}$ is the diagonal scale matrix composed of the scale $\boldsymbol{s}_i$ on each axis.

Using a tile-based rasterizer [98], we project the Gaussian primitives to the 2D image plane and sort them according to depth efficiently. The projected Gaussian distribution is:

$$\boldsymbol{\mu}' = \pi(\mathsf{T}_{CW}\boldsymbol{\mu}), \quad \boldsymbol{\Sigma}' = \boldsymbol{J}\boldsymbol{W}\boldsymbol{\Sigma}\boldsymbol{W}^\top\boldsymbol{J}^\top, \tag{13}$$

where $\boldsymbol{\mu}'$ and $\boldsymbol{\Sigma}'$ are the projected mean and covariance, π denotes the perspective projection, $\boldsymbol{J}$ is the Jacobian of the projective transformation, and $\boldsymbol{W}$ is the viewing transformation deduced from current camera pose T_{WC}. The rendered RGB image $\boldsymbol{I}$ at each pixel $\boldsymbol{u}$ is computed via alpha blending:

$$\boldsymbol{I}(\boldsymbol{u}) = \sum_{i \in \mathcal{G}(\boldsymbol{u})} w_i \boldsymbol{c}_i, \tag{14}$$

where the weight w_i of each of the depth-sorted Gaussian primitives $\mathcal{G}(\boldsymbol{u})$ covering pixel $\boldsymbol{u}$ is given by:

$$w_i = T_i\sigma_i, \; T_i = \prod_{j=1}^{i-1}(1 - \sigma_j), \; \sigma_i = \mathcal{N}(\boldsymbol{u}; \boldsymbol{\mu}_i', \boldsymbol{\Sigma}_i')\alpha_i, \tag{15}$$

where σ_i is the projected opacity of the i-th Gaussian primitive, computed using the 2D Gaussian density function $\mathcal{N}(\boldsymbol{u}; \boldsymbol{\mu}_i', \boldsymbol{\Sigma}_i')$ evaluated at pixel $\boldsymbol{u}$ with the projected mean $\boldsymbol{\mu}_i'$ and covariance $\boldsymbol{\Sigma}_i'$.

Gaussian Surfels Training. To achieve accurate and multi-view consistent geometry, we adopt Gaussian Surfels [11], a state-of-the-art 2DGS representation [23], by flattening 3D

Gaussian ellipsoids into 2D disks (last dimension of scale $s^z = 0$). For each pixel $\boldsymbol{u}$, we compute the surfel depth $d(\boldsymbol{u})$ as the ray-disk intersection distance, and obtain the normal $\boldsymbol{n}$ as the third column of the rotation matrix $\mathsf{R}(\boldsymbol{r})$. Using alpha blending, we render the depth map D and the normal map N using the weights w_i calculated in Eq. (15):

$$\mathsf{D}(\boldsymbol{u}) = \sum_{i \in \mathcal{G}(\boldsymbol{u})} w_i d_i(\boldsymbol{u}), \quad \mathsf{N}(\boldsymbol{u}) = \sum_{i \in \mathcal{G}(\boldsymbol{u})} w_i \boldsymbol{n}_i. \tag{16}$$

Given the training view with the RGB image $\widehat{\mathsf{I}}$ and the sparse depth map $\widehat{\mathsf{D}}$ projected from the LiDAR point cloud, we define the Gaussian splatting loss $\mathcal{L}_{\text{gs}}$ combining the photometric rendering $\mathcal{L}_{\text{photo}}$, depth rendering $\mathcal{L}_{\text{depth}}$, and area regularization $\mathcal{L}_{\text{area}}$ terms, given by:

$$\mathcal{L}_{\text{gs}} = \lambda_{\text{photo}}\mathcal{L}_{\text{photo}} + \lambda_{\text{depth}}\mathcal{L}_{\text{depth}} + \lambda_{\text{area}}\mathcal{L}_{\text{area}}, \tag{17}$$

$$\mathcal{L}_{\text{photo}} = 0.8 \cdot L_1\left(\mathsf{I}, \widehat{\mathsf{I}}\right) + 0.2 \cdot L_{\text{ssim}}\left(\mathsf{I}, \widehat{\mathsf{I}}\right), \tag{18}$$

$$\mathcal{L}_{\text{depth}} = L_1\left(\mathsf{D}, \widehat{\mathsf{D}}\right), \tag{19}$$

$$\mathcal{L}_{\text{area}} = \sum_{\boldsymbol{g}_i \in \mathcal{G}} s_i^x \cdot s_i^y, \tag{20}$$

where L_1 is the L1 loss, L_{ssim} is the structural similarity index measure (SSIM) loss [71], s_i^x and s_i^y are the scales of the Gaussian surfel $\boldsymbol{g}_i$. The area loss term $\mathcal{L}_{\text{area}}$ encourages minimal overlap among the surfels covering the surface.

To handle inaccurate camera poses resulting from imperfect LiDAR odometry and camera-LiDAR synchronization, we jointly optimize the camera poses on a manifold during radiance field training [42]. We also account for real-world lighting variations by optimizing per-frame exposure parameters [31].

D. Joint Optimization with Geometric Consistency

To enforce mutual alignment between the surfaces represented by the SDF and Gaussian splatting radiance field, we futhermore propose to jointly optimize the geometric consistency. This joint optimization helps resolve geometric ambiguities in the radiance field through the direct surface description of SDF, while simultaneously refining SDF's accuracy in regions with sparse LiDAR measurements using the dense photometric cues and multi-view consistency from the radiance field, see Fig. 3 (c), (d), and (e).

For each sampled Gaussian surfel, we randomly sample points along its normal direction $\boldsymbol{n}$ from the center $\boldsymbol{\mu}$ with random offsets $\epsilon \sim U(-\epsilon_{\max}, \epsilon_{\max})$. We enforce geometric consistency between the SDF and Gaussian surfels through a two-part consistency loss $\mathcal{L}_{\text{cons}}$, given by:

$$\mathcal{L}_{\text{cons}} = \lambda_{\text{cons}}^{\text{d}}\mathcal{L}_{\text{cons}}^{\text{d}} + \lambda_{\text{cons}}^{\text{v}}\mathcal{L}_{\text{cons}}^{\text{v}}, \tag{21}$$

$$\mathcal{L}_{\text{cons}}^{\text{d}} = \sum_{\boldsymbol{g}_i \in \mathcal{G}} \left| S\left(\boldsymbol{\mu}_i + \epsilon_i \boldsymbol{n}_i\right) - \epsilon_i \right|, \tag{22}$$

$$\mathcal{L}_{\text{cons}}^{\text{v}} = \sum_{\boldsymbol{g}_i \in \mathcal{G}} \left(1 - \frac{\nabla S\left(\boldsymbol{\mu}_i + \epsilon_i \boldsymbol{n}_i\right)^{\mathsf{T}} \boldsymbol{n}_i}{\left\| \nabla S\left(\boldsymbol{\mu}_i + \epsilon_i \boldsymbol{n}_i\right) \right\|} \right), \tag{23}$$

where $\mathcal{L}_{\text{cons}}^{\text{d}}$ enforces SDF values to match sampled offsets via an L1 loss, and $\mathcal{L}_{\text{cons}}^{\text{v}}$ aligns SDF gradients ∇S at the sampled points with surfel normals $\boldsymbol{n}$ using cosine distance.

We define the total loss $\mathcal{L}$ given by the sum of the SDF loss $\mathcal{L}_{\text{sdf}}$ in Eq. (4), Gaussian splatting loss $\mathcal{L}_{\text{gs}}$ in Eq. (17), and the geometric consistency loss $\mathcal{L}_{\text{cons}}$ in Eq. (21):

$$\mathcal{L} = \mathcal{L}_{\text{sdf}} + \mathcal{L}_{\text{gs}} + \mathcal{L}_{\text{cons}}. \tag{24}$$

We jointly optimize the neural point features $\{\boldsymbol{f}_i^g, \boldsymbol{f}_i^a\}_{i=1}^N$, decoder parameters, camera poses, and exposure correction parameters to minimize the total loss $\mathcal{L}$.

E. PINGS LiDAR-Visual SLAM System

We devise a LiDAR-visual SLAM system called PINGS using the proposed map representation, built on top of the LiDAR-only PIN-SLAM [51] system. PINGS alternates between two main steps: (i) mapping: incremental learning of the local neural point map $\mathcal{M}_l$, which jointly models the SDF and Gaussian splatting radiance field, and (ii) localization: odometry estimation using the learned SDF. In addition, loop closure detection and pose graph optimization run in parallel.

We initialize PINGS with 600 iterations of SDF training using only the first LiDAR scan. At subsequent timesteps, we jointly train the SDF and radiance field for 100 iterations. To prevent catastrophic forgetting during incremental mapping, we freeze decoder parameters after 30 timesteps and only update neural point features. We found the decoders converge on learning the interpretation capability within these 30 frames.

We maintain sliding window-like training pools $\mathcal{Q}_p$ and $\mathcal{Q}_i$ containing SDF-labeled sample points and image data whose view frustum overlaps with the local map $\mathcal{M}_l$, respectively. Each training iteration samples one image from $\mathcal{Q}_i$ and 8192 points from $\mathcal{Q}_p$ for optimization.

We estimate LiDAR odometry by aligning each new scan to the SDF's zero level set using an efficient Gauss-Newton optimization [74] that requires only SDF values and gradients queried at source point locations, eliminating the need for explicit point correspondences. Initial camera poses are derived from the LiDAR odometry and extrinsic calibration, then refined via gradient descent during the radiance field optimization to account for imperfect camera-LiDAR synchronization, as described in Sec. III-C.

In line with PIN-SLAM, we detect loop closures using the layout and features of the local neural point map. We then conduct pose graph optimization to correct the drift of the LiDAR odometry and get globally consistent poses. We move the neural points along with their associated LiDAR frames to keep a globally consistent map. Suppose $\Delta\mathsf{T}$ is the pose correction matrix of LiDAR frame L_i after pose graph optimization, we update the position $\boldsymbol{x}$ and orientation $\boldsymbol{q}$ of each neural point associated with L_i as:

$$\boldsymbol{x} \leftarrow \Delta\mathsf{T}\boldsymbol{x}, \quad \boldsymbol{q} \leftarrow \Delta\boldsymbol{q}\boldsymbol{q}, \tag{25}$$

where $\Delta\boldsymbol{q}$ is the rotation part of $\Delta\mathsf{T}$ in the form of a quaternion. Since the positions, rotations, and colors of the spawned Gaussian primitives are predicted in the local frames

of their anchor neural points, see Eq. (5), Eq. (6), and Eq. (9), they automatically transform with their anchor neural points, thus maintaining the global consistency of the radiance field.

PINGS aims to build static distance and radiance fields without artifacts from dynamic objects. Since measured points with large SDF values in stable free space likely correspond to dynamic objects [57], we identify neural points representing dynamic objects through SDF thresholding. We disable Gaussian primitive spawning for these points, effectively preventing dynamic objects from being rendered from the radiance field.

IV. Experimental Evaluation

The main focus of this paper is an approach for LiDAR-visual SLAM that unifies Gaussian splatting radiance fields and signed distance fields by leveraging their mutual consistency within a point-based implicit neural map representation.

We present our experiments to show the capabilities of our method called PINGS. The results of our experiments support our key claims, which are: (i) PINGS achieves better RGB and geometric rendering at novel views by constraining the Gaussian splatting radiance field using the SDF; (ii) PINGS builds a more accurate SDF for more accurate localization and surface reconstruction by leveraging dense photometric cues from the radiance field; (iii) PINGS enables large-scale globally consistent mapping with loop closures; (iv) PINGS builds a more compact map than previous methods for both radiance and distance fields.

A. Experimental Setup

1) Datasets: We evaluate PINGS on self-collected in-house car datasets and the Oxford Spires dataset [64]. Our in-house car datasets were collected using a robot car equipped with four Basler Ace cameras providing 360° visual coverage and an Ouster OS1-128 LiDAR (45° vertical FOV, 128 beams) mounted horizontally, both operating at 10 Hz. We calibrate the LiDAR-camera system using the method proposed by Wiesmann et al. [75] and generate reference poses through offline LiDAR bundle adjustment [76], incorporating RTK-GNSS data, point cloud alignment as well as constraints from precise geo-referenced terrestrial laser scans.

We evaluate the SLAM localization accuracy and scalability of PINGS on two long sequences from our dataset: a 5 km sequence with around 10,000 LiDAR scans and 40,000 images, and a second sequence which is a bit shorter. Both sequences traverse the same area in opposite directions on the same day. For better quantitative evaluation of the radiance field mapping quality, we select five subsequences of 150 LiDAR scans and 600 images each, as shown in Fig. 4. Having sequences captured in opposite driving directions and lane-level lateral displacement allows us to evaluate novel view rendering from substantially different viewpoints from the training views (out-of-sequence testing views), which is a critical capability for downstream tasks such as planning and simulation.

We evaluate surface reconstruction accuracy on the Oxford Spires dataset [64], which provides a millimeter-accurate reference map from a Leica RTC360 terrestrial laser scanner. The data was collected using a handheld system equipped with three global-shutter cameras and a 64-beam LiDAR.

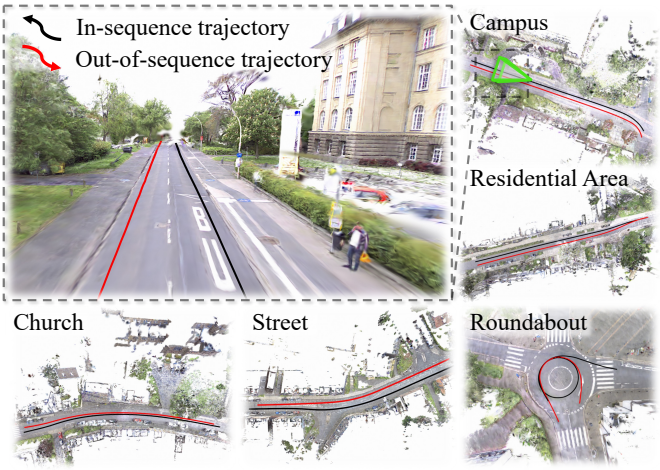


Fig. 4: Visualization of the five scenes from our *in-house car dataset* used for novel view rendering evaluation. For each scene, we show a bird's eye view rendering from the radiance field built by PINGS, with a detailed zoom-in of the *Campus* scene. The robot car traversed each area twice in opposite directions with lane-level lateral displacement. The black trajectory provided images for training (sampled) and in-sequence testing (unsampled), while the red trajectory provided out-of-sequence testing views.

2) Parameters and Implementation Details: For mapping parameters, we set the local map radius r_l to 80 m, voxel resolution v_p to 0.3 m, and maximum sample offset for consistency loss ϵ_{max} to $0.5\,v_p$. The training data pool $\mathcal{Q}_d$ has a capacity of $2 \cdot 10^7$ SDF-labeled sample points, and $\mathcal{Q}_i$ has a capacity of 200 images. During map optimization, we use Adam [32] with learning rates of 0.002 for neural point features, 0.001 for decoder, camera poses and exposure corrections parameters. The neural point feature dimensions F_g and F_a are set to 32 and 16, respectively. All decoders use shallow MLPs with one hidden layer of 128 neurons. Each neural point spawns $K = 8$ Gaussian primitives. For decoder activations, we use sigmoid for SDF decoder D_d and color decoder D_c, tanh for offset decoder D_o and opacity decoder D_α, and exponential for scale decoder D_s. The Gaussian spawning offset is scaled to $[-2v_p, 2v_p]$, and the scale output is clamped to a maximum of $2v_p$. The rotation decoder D_r output is normalized to valid unit quaternions. The weights for different loss terms are set to: $\lambda_{\text{bce}} = 1.0$, $\lambda_{\text{eik}} = 0.5$, $\lambda_{\text{photo}} = 1.0$, $\lambda_{\text{depth}} = 0.01$, $\lambda_{\text{area}} = 0.001$, and $\lambda^d_{\text{cons}} = \lambda^v_{\text{cons}} = 0.02$.

For training and testing, we use image resolutions of $512 \times 1,032$ for the in-house car dataset and 540×720 for the Oxford Spires dataset. The experiments are carried out on a single NVIDIA A6000 GPU.

B. Novel View Rendering Quality Evaluation

We evaluate novel view rendering quality on five subsequences from the in-house car dataset. For quantitative evaluation, we employ standard metrics: PSNR [45], SSIM [71], and LPIPS [88] to assess photorealism, along with Depth-L1 error to measure geometric accuracy. We compute these metrics

TABLE I: Quantitative comparison of rendering quality on the *in-house car dataset*. We evaluate rendering photorealism using PSNR, SSIM, and LPIPS metrics, and geometric accuracy using Depth-L1 error (in m). Best results are shown in **bold**, second best are underscored.

Sequence	Method	In-Sequence Testing View				Out-of-Sequence Testing View			
		PSNR↑	SSIM↑	LPIPS↓	Depth-L1↓	PSNR↑	SSIM↑	LPIPS↓	Depth-L1↓
Church	3DGS	18.02	0.62	0.52	1.45	16.37	0.59	0.52	1.22
	GSS	18.04	0.63	0.51	1.37	16.44	0.60	0.52	1.25
	Neural Point + 3DGS	20.89	0.71	0.41	0.80	19.56	0.70	0.42	0.78
	Neural Point + GGS	22.56	0.75	0.36	**0.43**	20.48	0.74	0.38	0.47
	PINGS (Ours)	**22.93**	**0.78**	**0.33**	**0.43**	**20.79**	**0.76**	**0.34**	**0.46**
Residential Area	3DGS	17.60	0.58	0.52	2.22	14.63	0.53	0.56	2.78
	GSS	17.56	0.59	0.51	2.26	14.80	0.54	0.55	2.68
	Neural Point + 3DGS	21.10	0.71	0.38	0.89	18.34	0.65	0.42	0.93
	Neural Point + GSS	22.33	0.73	0.35	**0.53**	19.31	0.69	0.38	0.69
	PINGS (Ours)	**22.67**	**0.77**	**0.30**	**0.53**	**19.48**	**0.71**	**0.34**	**0.68**
Street	3DGS	16.39	0.56	0.55	2.09	15.73	0.57	0.53	2.30
	GSS	16.85	0.59	0.53	1.87	16.01	0.59	0.52	2.15
	Neural Point + 3DGS	19.74	0.68	0.42	0.81	18.02	0.64	0.44	0.79
	Neural Point + GSS	22.13	0.75	0.35	0.29	19.09	0.69	0.40	0.49
	PINGS (Ours)	**22.45**	**0.78**	**0.32**	**0.28**	**19.34**	**0.71**	**0.37**	**0.47**
Campus	3DGS	17.38	0.57	0.52	2.70	14.88	0.49	0.58	3.60
	GSS	17.34	0.59	0.51	2.36	14.96	0.51	0.57	3.42
	Neural Point + 3DGS	20.04	0.67	0.40	1.06	17.83	0.60	0.44	1.19
	Neural Point + GSS	21.82	0.72	0.35	0.65	18.71	0.64	0.41	**0.79**
	PINGS (Ours)	**22.40**	**0.76**	**0.31**	**0.64**	**18.91**	**0.66**	**0.38**	0.80
Roundabout	3DGS	21.20	0.71	0.39	0.87	18.97	0.69	0.40	0.85
	GSS	21.74	0.72	0.38	0.55	19.10	0.69	0.40	0.60
	Neural Point + 3DGS	21.44	0.75	0.35	0.72	19.23	0.72	0.37	0.78
	Neural Point + GSS	**23.54**	**0.82**	**0.28**	**0.47**	20.22	**0.78**	0.31	0.55
	PINGS (Ours)	23.45	**0.82**	**0.28**	**0.47**	**20.23**	0.77	**0.30**	**0.54**

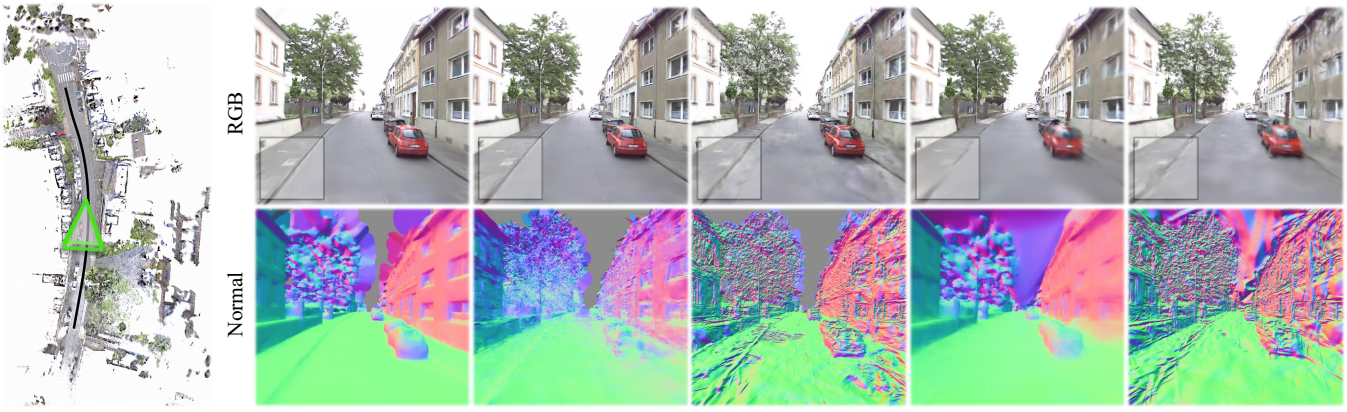


Fig. 5: Qualitative comparison of rendering quality on the *in-house car dataset*. Left: Bird's eye view rendering of the *Church* scene, showing the training view trajectory (black line) and the test viewpoint for comparison (green camera frustum). Right: RGB and normal map renderings from different methods at the test viewpoint, with detailed comparison of curb and sidewalk rendering in the highlighted box.

for both in-sequence and out-of-sequence testing views. We consider the following methods for comparison:

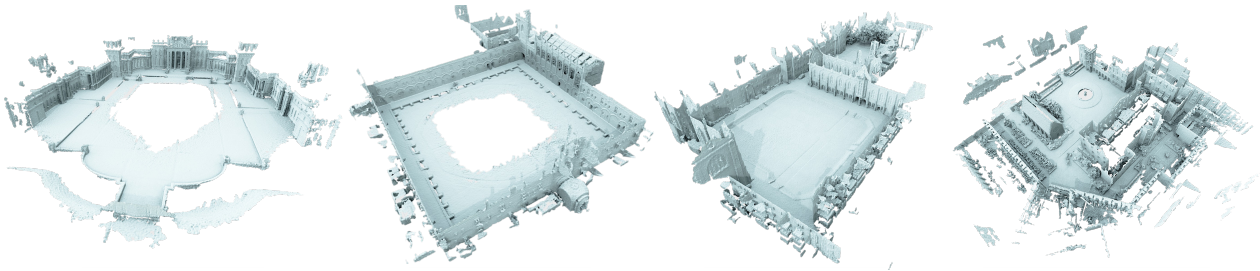
- *3DGS* [30]: An incremental training variant of 3DGS initialized with LiDAR measurements and supervised with the depth rendering loss $\mathcal{L}_{\text{depth}}$ as defined in Sec. III-C.
- *GSS* [11]: Gaussian surfels splatting, a state-of-the-art 2D Gaussian representation using surfels instead of 3D ellipsoids. It uses the same setup as the *3DGS* baseline but adds the depth-normal consistency loss from GSS [11].
- *Neural Point+3DGS*: Our extension of Scaffold-GS [38] that enables incremental training and adds supervision of neural point geometric features through the SDF branch, as detailed in Sec. III-C.
- *Neural Point+GSS*: A variant that replaces the 3D Gaussian in *Neural Point+3DGS* with 2D Gaussian surfels.
- *PINGS*: Our complete framework that extends *Neural Point+GSS* by introducing geometric consistency loss $\mathcal{L}_{\text{cons}}$ into the joint training, as described in Sec. III-D.

For fair comparison, we disable the localization part and use the ground truth pose for all the compared methods. For 3DGS and GSS, we initialize their Gaussian primitive density to match the total number of Gaussians spawned by PINGS.

We show the quantitative comparison on five sequences in Tab. I as well as show the qualitative comparison of the RGB and normal map rendering results on the *church* scene at a novel view in Fig. 5. Our method PINGS achieves

TABLE II: Quantitative evaluation of surface reconstruction quality on the *Oxford-Spires dataset*. We use the metrics include accuracy error (in m), completeness error (in m), and Chamfer distance (in m), as well as precision, recall and F-score (with $0.1\,\mathrm{m}$ threshold). † denotes methods requiring offline batch processing. Best results are shown in **bold**, second best are underscored.

Sequence	Method	Accuracy ↓	Completeness ↓	Chamfer Distance ↓	Precision ↑	Recall ↑	F-score ↑
Blenheim Palace 05	OpenMVS†	0.126	1.045	0.586	0.574	0.381	0.458
	Nerfacto†	0.302	0.676	0.489	0.388	0.257	0.309
	GSS	0.204	0.254	0.229	0.271	0.261	0.266
	VDB-Fusion	0.098	**0.123**	0.111	0.646	**0.746**	0.692
	PIN-SLAM	0.078	0.136	0.107	0.768	0.712	0.739
	PINGS (Ours)	**0.072**	0.133	**0.102**	**0.794**	0.726	**0.758**
Christ Church 02	OpenMVS†	**0.046**	5.381	2.714	**0.886**	0.266	0.410
	Nerfacto†	0.219	4.435	2.327	0.532	0.254	0.343
	GSS	0.174	0.292	0.233	0.407	0.301	0.346
	VDB-Fusion	0.098	**0.243**	0.171	0.655	**0.524**	0.582
	PIN-SLAM	0.069	0.252	0.160	0.812	0.497	0.617
	PINGS (Ours)	0.067	0.251	**0.159**	0.815	0.502	**0.622**
Keble College 04	OpenMVS†	**0.067**	0.342	0.205	**0.918**	0.718	**0.806**
	Nerfacto†	0.137	0.150	0.144	0.654	0.709	0.680
	GSS	0.171	0.162	0.167	0.424	0.518	0.466
	VDB-Fusion	0.103	**0.101**	0.102	0.639	**0.821**	0.719
	PIN-SLAM	0.096	0.108	0.102	0.701	0.793	0.744
	PINGS (Ours)	0.093	0.106	**0.099**	0.705	0.799	0.749
Observatory Quarter 01	OpenMVS†	**0.048**	0.622	0.335	**0.902**	0.618	**0.734**
	Nerfacto†	0.197	0.398	0.298	0.587	0.598	0.592
	GSS	0.179	0.184	0.181	0.377	0.443	0.407
	VDB-Fusion	0.123	**0.109**	0.116	0.573	**0.737**	0.645
	PIN-SLAM	0.105	0.129	0.117	0.654	0.677	0.665
	PINGS (Ours)	0.102	0.124	**0.113**	0.659	0.705	0.681



Blenheim Palace Christ Church Keble College Observatory Quarter

Fig. 6: Qualitative results of the surface mesh reconstruction by PINGS on the *Oxford-Spires dataset*. The meshes are extracted using marching cubes algorithm from the SDF with a resolution of $0.1\,\mathrm{m}$.

superior performance in both photorealistic rendering quality and depth rendering accuracy on the in-house car dataset, and consistently outperforms the baselines for both in-sequence and out-of-sequence testing views. Analysis of the results reveals several insights: (i) The adoption of GSS over 3DGS leads to improved geometric rendering quality and enhanced out-of-sequence rendering photorealism; (ii) Our approach of spawning Gaussians from neural points and jointly training with the distance field provides better optimization control and reduces floating Gaussians in free space, resulting in superior rendering quality; (iii) The addition of geometric consistency constraints from SDF enables better surface alignment of Gaussian surfels, further enhancing both geometric accuracy and photorealistic rendering quality, as evidenced by the smoother normal maps produced by PINGS compared to *Neural Point+GSS*. These improvements are less significant in the *Roundabout* scene, where the dense viewpoint coverage from the vehicle's circular trajectory provides strong multi-view constraints, reducing the benefit of additional geometric constraints from the SDF.

In sum, this experiment validates that PINGS achieves better RGB and geometric rendering at novel views by constraining the Gaussian splatting radiance field using the SDF.

C. Surface Reconstruction Quality Evaluation

We evaluate surface reconstruction quality on four sequences from the Oxford-Spires dataset. We follow the benchmark [64] to report the metrics including accuracy error, completeness error, and Chamfer distance, as well as precision, recall and F-score calculated with a threshold of $0.1\,\mathrm{m}$. We compare the performance of PINGS with five state-of-the-art methods, including OpenMVS [5], Nerfacto [62], GSS [11], VDB-Fusion [67], and PIN-SLAM [51]. To ensure fair comparison of geometric mapping quality, we disable the localization modules of PIN-SLAM and PINGS and use ground truth poses across all methods. For GSS, after completing the radiance field mapping, we render depth maps at each frame and apply TSDF fusion [67] for mesh extraction. Results of the vision-based offline processing methods (OpenMVS and Nerfacto) are taken from the benchmark [64]. For the remaining methods (GSS, VDB-Fusion, PIN-SLAM, and PINGS),

TABLE III: Localization performance comparison of PINGS against state-of-the-art odometry/SLAM methods on the *in-house car dataset*. We report average relative translation error (ARTE) [%] and absolute trajectory error (ATE) [m]. Odometry methods are shown above the midrule, SLAM methods below. Best results are shown in **bold**, second best are underscored.

Method	Seq. 1 (5.0 km)		Seq. 2 (3.7 km)	
	ARTE [%] ↓	ATE [m] ↓	ARTE [%] ↓	ATE [m] ↓
F-LOAM [69]	1.96	28.52	1.93	27.00
KISS-ICP [68]	1.49	8.17	1.38	8.22
PIN odometry [51]	0.95	4.51	0.98	5.64
PINGS odometry	0.73	5.17	0.59	4.78
SuMa [3]	5.55	39.90	4.42	44.78
MULLS [50]	2.23	40.37	1.64	33.82
PIN-SLAM [51]	1.00	3.17	0.98	4.44
PINGS (Ours)	**0.68**	**1.99**	**0.58**	**3.47**

we extract surface meshes from their SDFs using marching cubes [37] at a resolution of 0.1 m.

We show the qualitative results of PINGS on the four sequences in Fig. 6. Quantitative comparisons in Tab. II demonstrate that PINGS achieves superior performance, particularly in terms of Chamfer distance and F-score metrics. Notably, when using identical neural point resolution, PINGS consistently outperforms PIN-SLAM across all metrics through its joint optimization of the radiance field and geometric consistency constraints. This improvement validates that incorporating dense photometric cues and multi-view consistency from the radiance field improves the SDF accuracy, ultimately enabling surface mesh reconstruction with higher quality.

D. SLAM Localization Accuracy Evaluation

We compare the pose estimation performance of PINGS against state-of-the-art LiDAR odometry/SLAM systems on two full sequences of the in-house car dataset. The compared methods include F-LOAM [69], KISS-ICP [68], SuMa [3], MULLS [50], and PIN-SLAM [51]. For evaluation metrics, we use average relative translation error (ARTE) [15] to assess odometry drift and absolute trajectory error (ATE) [89] to measure the global pose estimation accuracy. The results shown in Tab. III demonstrate that PINGS achieves both lower odometry drift and superior global localization accuracy than the compared approaches. Compared to PIN-SLAM, the improvement stems from the refined SDF obtained through joint optimization with the radiance field and geometric consistency constraints. The improved SDF leads to more accurate LiDAR odometry and relocalization during loop closure correction.

E. Large-Scale Globally Consistent Mapping

Fig. 1 demonstrates the globally consistent SLAM capabilities of PINGS on a challenging 5 km sequence from our in-house car dataset. In Fig. 7, we show the effect of loop closure correction. Without loop closure correction, odometry drift accumulates over time, causing neural points to be inconsistently placed when revisiting previously mapped regions. This results in visual artifacts in the radiance field rendering, such as duplicate objects and trees incorrectly appearing on the road. After conducting loop closure correction and updating the map, both the neural point map and RGB rendering achieve

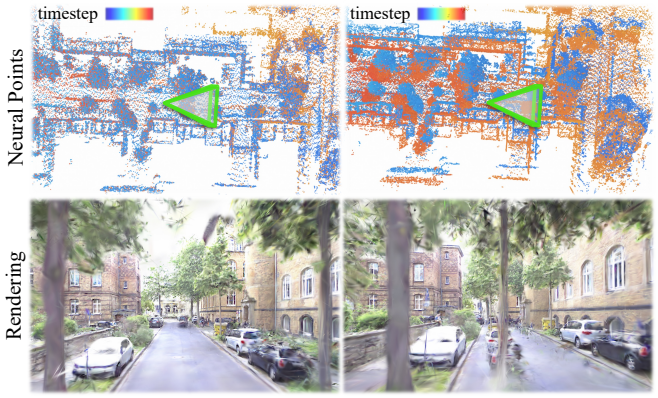


Fig. 7: Demonstration of the effect of loop closure correction on the *in-house car dataset*. When the vehicle revisits a previously mapped region, we compare the neural point map (colored by timestep) and RGB rendering (viewed from the green frustum) with and without correcting the loop closure. Without loop closure correction, the misaligned neural points create visual artifacts like trees appearing on the road. After applying loop closure correction and map update, we achieve globally consistent neural point map and RGB rendering.

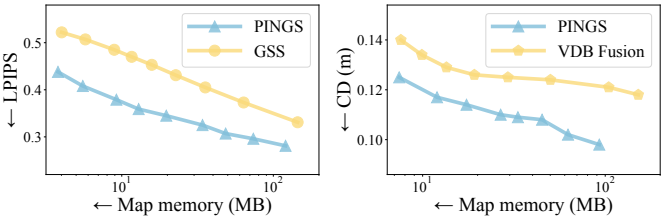


Fig. 8: Map memory efficiency analysis comparing mapping quality versus memory usage. Left: Radiance field comparison between PINGS and GSS on the *in-house car dataset* using LPIPS metric. Right: Distance field comparison between PINGS and VDB-Fusion on the *Oxford Spires dataset* using Chamfer distance. Points represent results at different map resolution, with points closer to the bottom-left corner indicating better quality-memory trade-off.

global consistency. These results validate that PINGS can build globally consistent maps at large scale through loop closure correction by leveraging the elasticity of neural points.

F. Map Memory Efficiency Evaluation

Fig. 8 depicts the map memory usage in relation to the rendering quality for the radiance field and the surface reconstruction quality for the distance field. Experiment results validate that storing the neural points and decoder MLPs in PINGS is more memory-efficient than directly storing the Gaussian primitives or the discrete SDF in voxels. With equivalent memory usage, PINGS achieves superior performance across both metrics: better novel view rendering photorealism (lower LPIPS) compared to GSS [11] on the in-house car dataset, and better surface reconstruction accuracy (lower Chamfer distance) compared to the discrete TSDF-based method VDB-Fusion [67] on the Oxford Spires dataset. Moreover, while GSS and VDB-Fusion each model only a single field type, our PINGS framework efficiently represents both radiance field and SDF within a single map. The efficiency of PINGS comes from the globally-shared decoder MLPs that learn common

patterns, and locally-defined neural points that compactly encode multiple Gaussian primitives and continuous SDF values through feature vectors instead of storing them explicitly.

V. Limitations

Our current approach has three main limitations. First, although our SDF mapping and LiDAR odometry modules operate at sensor frame rate, the computationally intensive radiance field mapping results in an overall processing time of around five seconds per frame on an NVIDIA A6000 GPU. This performance bottleneck could potentially be addressed through recent advances in efficient optimization schemes for Gaussian splatting training [20, 40]. Second, PINGS relies solely on online per-scene optimization without using any pre-trained priors. Incorporating such priors [7] into our unified map representation could improve both mapping quality and convergence speed. Finally, though PINGS can filter dynamic objects using the distance field, it lacks explicit 4D modeling capabilities. This limitation is noticeable in highly dynamic environments and when objects transition between static and dynamic states. Future work could address this challenge by incorporating object detection priors [9, 80] to enable accurate 4D mapping of both radiance and distance fields.

VI. Conclusion

In this paper, we present a new LiDAR-visual SLAM system making use of a novel map representation that unifies a continuous signed distance field and a Gaussian splatting radiance field within an elastic and compact set of neural points. By introducing mutual geometric consistency constraints between these fields, we jointly improve both representations. The distance field provides geometric structure to guide radiance field optimization, while the radiance field's dense photometric cues and multi-view consistency enhance the distance field's accuracy. Our experimental results on challenging large-scale datasets show that our method can incrementally construct globally consistent maps that outperform baseline methods in the novel view rendering fidelity, surface reconstruction quality, odometry estimation accuracy, and map memory efficiency.

Acknowledgements

This work has partially been funded by the European Union under the grant agreements No 101070405 (DigiForest), by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy, EXC-2070 – 390732324 – PhenoRob, under STA 1051/5-1 within the FOR 5351 – 459376902 (AID4Crops), and by the German Federal Ministry of Education and Research (BMBF) in the project "Robotics Institute Germany", grant No. 16ME0999.

References

[1] J. Abou-Chakra, F. Dayoub, and N. Sünderhauf. ParticleNeRF: A Particle-Based Encoding for Online Neural Radiance Fields. In *Proc. of the IEEE Winter Conf. on Applications of Computer Vision (WACV)*, 2024.

[2] J. Abou-Chakra, K. Rana, F. Dayoub, and N. Suenderhauf. Physically Embodied Gaussian Splatting: A Realtime Correctable World Model for Robotics. In *Proc. of the Conf. on Robot Learning (CoRL)*, 2024.

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

[4] M. Bortolon, T. Tsesmelis, S. James, F. Poiesi, and A. Del Bue. 6DGS: 6D Pose Estimation from a Single Image and a 3D Gaussian Splatting Model. In *Proc. of the Europ. Conf. on Computer Vision (ECCV)*, 2024.

[5] D. Cernea. OpenMVS: Multi-View Stereo Reconstruction Library, 2020.

[6] H. Chen, C. Li, and G.H. Lee. NeuSG: Neural implicit surface reconstruction with 3d gaussian splatting guidance. *arXiv preprint*, arXiv:2023.00846, 2023.

[7] Y. Chen, J. Wang, Z. Yang, S. Manivasagam, and R. Urtasun. G3R: Gradient Guided Generalizable Reconstruction. In *Proc. of the Europ. Conf. on Computer Vision (ECCV)*, 2024.

[8] Z. Chen, Z. Li, L. Song, L. Chen, J. Yu, J. Yuan, and Y. Xu. NeuRBF: A Neural Fields Representation with Adaptive Radial Basis Functions. In *Proc. of the IEEE/CVF Intl. Conf. on Computer Vision (ICCV)*, 2023.

[9] Z. Chen, J. Yang, J. Huang, R.d. Lutio, J.M. Esturo, B. Ivanovic, O. Litany, Z. Gojcic, S. Fidler, M. Pavone, L. Song, and Y. Wang. OmniRe: Omni Urban Scene Reconstruction. *arXiv preprint*, arXiv:2408.16760, 2024.

[10] J. Cui, J. Cao, F. Zhao, Z. He, Y. Chen, Y. Zhong, L. Xu, Y. Shi, and J. Yu. LetsGo: Large-Scale Garage Modeling and Rendering via LiDAR Assisted Gaussian Primitives. *ACM Trans. on Graphics (TOG)*, 43(6):1–18, 2024.

[11] P. Dai, J. Xu, W. Xie, X. Liu, H. Wang, and W. Xu. High-quality Surface Reconstruction using Gaussian Surfels. In *Proc. of the Intl. Conf. on Computer Graphics and Interactive Techniques (SIGGRAPH)*, 2024.

[12] D. Driess, I. Schubert, P. Florence, Y. Li, and M. Toussaint. Reinforcement Learning with Neural Radiance Fields. In *Proc. of the Conf. on Neural Information Processing Systems (NeurIPS)*, 2022.

[13] T. Fischer, J. Kulhanek, S.R. Bulò, L. Porzi, M. Pollefeys, and P. Kontschieder. Dynamic 3D Gaussian Fields for Urban Areas. In *Proc. of the Conf. on Neural Information Processing Systems (NeurIPS)*, 2024.

[14] D. Fox, W. Burgard, and S. Thrun. The dynamic window approach to collision avoidance. *IEEE Journal of Robotics and Automation*, 4(1):23–33, 1997.

[15] A. Geiger, P. Lenz, and R. Urtasun. Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[16] G. Grisetti, C. Stachniss, and W. Burgard. Improved Techniques for Grid Mapping with Rao-Blackwellized Particle Filters. *IEEE Trans. on Robotics (TRO)*, 23(1):34–46, 2007.

[17] A. Gropp, L. Yariv, N. Haim, M. Atzmon, and Y. Lipman.

Implicit Geometric Regularization for Learning Shapes. In *Proc. of the Intl. Conf. on Machine Learning (ICML)*, 2020.

[18] A. Guédon and V. Lepetit. Sugar: Surface-aligned gaussian splatting for efficient 3d mesh reconstruction and high-quality mesh rendering. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2024.

[19] G. Hess, C. Lindström, M. Fatemi, C. Petersson, and L. Svensson. SplatAD: Real-Time Lidar and Camera Rendering with 3D Gaussian Splatting for Autonomous Driving. *arXiv preprint*, arXiv:2411.16816, 2024.

[20] L. Höllein, A. Božič, M. Zollhöfer, and M. Nießner. 3DGS-LM: Faster Gaussian-Splatting Optimization with Levenberg-Marquardt. *arXiv preprint*, arXiv:2409.12892, 2024.

[21] S. Hong, J. He, X. Zheng, C. Zheng, and S. Shen. 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.

[22] A. Hornung, K. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard. OctoMap: An Efficient Probabilistic 3D Mapping Framework Based on Octrees. *Autonomous Robots*, 34(3):189–206, 2013.

[23] B. Huang, Z. Yu, A. Chen, A. Geiger, and S. Gao. 2D Gaussian Splatting for Geometrically Accurate Radiance Fields. In *Proc. of the Intl. Conf. on Computer Graphics and Interactive Techniques (SIGGRAPH)*, 2024.

[24] N. Hughes, Y. Chang, S. Hu, R. Talak, R. Abdulhai, J. Strader, and L. Carlone. Foundations of spatial perception for robotics: Hierarchical representations and real-time systems. *Intl. Journal of Robotics Research (IJRR)*, 43(10):1457–1505, 2024.

[25] C. Jiang, R. Gao, K. Shao, Y. Wang, R. Xiong, and Y. Zhang. LI-GS: Gaussian Splatting with LiDAR Incorporated for Accurate Large-Scale Reconstruction. *arXiv preprint*, arXiv:2409.12899, 2024.

[26] L. Jin, X. Zhong, Y. Pan, J. Behley, C. Stachniss, and M. Popovic. ActiveGS: Active Scene Reconstruction using Gaussian Splatting. *IEEE Robotics and Automation Letters (RA-L)*, 10(5):4866–4873, 2025.

[27] R. Jin, Y. Gao, H. Lu, and F. Gao. GS-Planner: A Gaussian-Splatting-based Planning Framework for Active High-Fidelity Reconstruction. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2024.

[28] M. Kazhdan and H. Hoppe. Screened poisson surface reconstruction. *ACM Trans. on Graphics*, 32(3):1–13, 2013.

[29] N. Keetha, J. Karhade, K.M. Jatavallabhula, G. Yang, S. Scherer, D. Ramanan, and J. Luiten. SplaTAM: Splat Track & Map 3D Gaussians for Dense RGB-D SLAM. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2024.

[30] B. Kerbl, G. Kopanas, T. Leimkühler, and G. Drettakis. 3D Gaussian Splatting for Real-Time Radiance Field Rendering. *ACM Trans. on Graphics (TOG)*, 42(4):1–14, 2023.

[31] B. Kerbl, A. Meuleman, G. Kopanas, M. Wimmer, A. Lanvin, and G. Drettakis. A Hierarchical 3D Gaussian Representation for Real-Time Rendering of Very Large Datasets. *ACM Trans. on Graphics (TOG)*, 43(4):1–15, 2024.

[32] D. Kingma and J. Ba. Adam: A Method for Stochastic Optimization. In *Proc. of the Intl. Conf. on Learning Representations (ICLR)*, 2015.

[33] M. Klingensmith, I. Dryanovski, S. Srinivasa, and J. Xiao. Chisel: Real time large scale 3d reconstruction onboard a mobile device using spatially hashed signed distance fields. In *Proc. of Robotics: Science and Systems (RSS)*, 2015.

[34] T. Li, X. Wen, Y.S. Liu, H. Su, and Z. Han. Learning Deep Implicit Functions for 3D Shapes with Dynamic Code Clouds. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[35] Y. Li, Z. Kuang, T. Li, G. Zhou, S. Zhang, and Z. Yan. ActiveSplat: High-Fidelity Scene Reconstruction through Active Gaussian Splatting. *arXiv preprint*, arXiv:2410.21955, 2024.

[36] Y. Liu, C. Luo, L. Fan, N. Wang, J. Peng, and Z. Zhang. Citygaussian: Real-time high-quality large-scale scene rendering with gaussians. In *Proc. of the Europ. Conf. on Computer Vision (ECCV)*, 2024.

[37] W. Lorensen and H. Cline. Marching Cubes: a High Resolution 3D Surface Construction Algorithm. In *Proc. of the Intl. Conf. on Computer Graphics and Interactive Techniques (SIGGRAPH)*, 1987.

[38] T. Lu, M. Yu, L. Xu, Y. Xiangli, L. Wang, D. Lin, and B. Dai. Scaffold-GS: Structured 3d gaussians for view-adaptive rendering. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2024.

[39] X. Lyu, Y.T. Sun, Y.H. Huang, X. Wu, Z. Yang, Y. Chen, J. Pang, and X. Qi. 3DGSR: Implicit Surface Reconstruction with 3D Gaussian Splatting. *ACM Trans. on Graphics (TOG)*, 43(6):1–12, 2024.

[40] S.S. Mallick, R. Goel, B. Kerbl, M. Steinberger, F.V. Carrasco, and F. De La Torre. Taming 3DGS: High-Quality Radiance Fields with Limited Resources. In *Proc. of the Intl. Conf. on Computer Graphics and Interactive Techniques (SIGGRAPH)*, 2024.

[41] R. Mascaro and M. Chli. Scene Representations for Robotic Spatial Perception. *Annual Review of Control, Robotics, and Autonomous Systems*, 8(5):1–27, 2024.

[42] H. Matsuki, R. Murai, P.H. Kelly, and A.J. Davison. Gaussian splatting SLAM. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2024.

[43] M.H. Maximum Wilder-Smith, Vaishakh Patil. Radiance Fields for Robotic Teleoperation. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2024.

[44] L. Mescheder, M. Oechsle, M. Niemeyer, S. Nowozin,

and A. Geiger. Occupancy networks: Learning 3d reconstruction in function space. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[45] B. Mildenhall, P. Srinivasan, M. Tancik, J. Barron, R. Ramamoorthi, and R. Ng. NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis. In *Proc. of the Europ. Conf. on Computer Vision (ECCV)*, 2020.

[46] R.A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A.J. Davison, P. Kohli, J. Shotton, S. Hodges, and A. Fitzgibbon. KinectFusion: Real-Time Dense Surface Mapping and Tracking. In *Proc. of the Intl. Symp. on Mixed and Augmented Reality (ISMAR)*, 2011.

[47] M. Nießner, M. Zollhöfer, S. Izadi, and M. Stamminger. Real-Time 3D Reconstruction at Scale Using Voxel Hashing. *ACM Trans. on Graphics (TOG)*, 32(6):1–11, 2013.

[48] H. Oleynikova, Z. Taylor, M. Fehr, R. Siegwar, and J. 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)*, 2017.

[49] J. Ortiz, A. Clegg, J. Dong, E. Sucar, D. Novotny, M. Zollhoefer, and M. Mukadam. isdf: Real-time neural signed distance fields for robot perception. In *Proc. of Robotics: Science and Systems (RSS)*, 2022.

[50] Y. Pan, P. Xiao, Y. He, Z. Shao, and Z. Li. MULLS: Versatile LiDAR SLAM Via Multi-Metric Linear Least Square. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2021.

[51] Y. Pan, X. Zhong, L. Wiesmann, T. Posewsky, J. Behley, and C. Stachniss. PIN-SLAM: LiDAR SLAM Using a Point-Based Implicit Neural Representation for Achieving Global Map Consistency. *IEEE Trans. on Robotics (TRO)*, 40:4045–4064, 2024.

[52] J.J. Park, P. Florence, J. Straub, R. Newcombe, and S. Lovegrove. DeepSDF: Learning Continuous Signed Distance Functions for Shape Representation. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[53] V. Reijgwart, C. Cadena, R. Siegwart, and L. Ott. Efficient volumetric mapping of multi-scale environments using wavelet-based compression. In *Proc. of Robotics: Science and Systems (RSS)*, 2023.

[54] K. Rematas, A. Liu, P.P. Srinivasan, J.T. Barron, A. Tagliasacchi, T. Funkhouser, and V. Ferrari. Urban radiance fields. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[55] K. Ren, L. Jiang, T. Lu, M. Yu, L. Xu, Z. Ni, and B. Dai. Octree-GS: Towards Consistent Real-time Rendering with LOD-structured 3D Gaussians. *arXiv preprint*, arXiv:2403.17898, 2024.

[56] E. Sandström, Y. Li, L. Van Gool, and M. R. Oswald. Point-SLAM: Dense Neural Point Cloud-based SLAM. In *Proc. of the IEEE/CVF Intl. Conf. on Computer Vision (ICCV)*, 2023.

[57] L. Schmid, O. Andersson, A. Sulser, P. Pfreundschuh, and R. Siegwart. Dynablox: Real-time Detection of Diverse Dynamic Objects in Complex Environments. *IEEE Robotics and Automation Letters (RA-L)*, 8(10):6259–6266, 2023.

[58] G. Song, C. Cheng, and H. Wang. GVKF: Gaussian Voxel Kernel Functions for Highly Efficient Surface Reconstruction in Open Scenes. In *Proc. of the Conf. on Neural Information Processing Systems (NeurIPS)*, 2024.

[59] C. Stachniss, G. Grisetti, and W. Burgard. Information Gain-based Exploration Using Rao-Blackwellized Particle Filters. In *Proc. of Robotics: Science and Systems (RSS)*, 2005.

[60] E. Sucar, S. Liu, J. Ortiz, and A.J. Davison. imap: Implicit mapping and positioning in real-time. In *Proc. of the IEEE/CVF Intl. Conf. on Computer Vision (ICCV)*, 2021.

[61] M. Tancik, V. Casser, X. Yan, S. Pradhan, B. Mildenhall, P.P. Srinivasan, J.T. Barron, and H. Kretzschmar. Block-NeRF: Scalable Large Scene Neural View Synthesis. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[62] M. Tancik, E. Weber, E. Ng, R. Li, B. Yi, J. Kerr, T. Wang, A. Kristoffersen, J. Austin, K. Salahi, A. Ahuja, D. McAllister, and A. Kanazawa. Nerfstudio: A Modular Framework for Neural Radiance Field Development. In *Proc. of the Intl. Conf. on Computer Graphics and Interactive Techniques (SIGGRAPH)*, 2023.

[63] Y. Tao, Y. Bhalgat, L.F.T. Fu, M. Mattamala, N. Chebrolu, and M. Fallon. SiLVR: Scalable Lidar-Visual Reconstruction with Neural Radiance Fields for Robotic Inspection. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2024.

[64] Y. Tao, M.A. Munoz-Banon, L. Zhang, J. Wang, L.F.T. Fu, and M. Fallon. The Oxford Spires Dataset: Benchmarking Large-Scale LiDAR-Visual Localisation, Reconstruction and Radiance Field Methods. *arXiv preprint*, arXiv:2411.10546, 2024.

[65] S. Thrun, D. Fox, W. Burgard, and F. Dellaert. Robust Monte Carlo Localization for Mobile Robots. *Artificial Intelligence*, 128(1-2), 2001.

[66] I. Vizzo, X. Chen, N. Chebrolu, J. Behley, and C. Stachniss. Poisson Surface Reconstruction for LiDAR Odometry and Mapping. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2021.

[67] I. Vizzo, T. Guadagnino, J. Behley, and C. Stachniss. VDBFusion: Flexible and Efficient TSDF Integration of Range Sensor Data. *Sensors*, 22(3):1296, 2022.

[68] I. Vizzo, T. Guadagnino, B. Mersch, L. Wiesmann, J. Behley, and C. Stachniss. KISS-ICP: In Defense of Point-to-Point ICP – Simple, Accurate, and Robust Registration If Done the Right Way. *IEEE Robotics and Automation Letters (RA-L)*, 8(2):1029–1036, 2023.

[69] H. Wang, C. Wang, C. Chen, and L. Xie. F-LOAM: Fast LiDAR Odometry and Mapping. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems*

(IROS), 2021.

[70] P. Wang, L. Liu, Y. Liu, C. Theobalt, T. Komura, and W. Wang. Neus: Learning neural implicit surfaces by volume rendering for multi-view reconstruction. In *Proc. of the Conf. on Neural Information Processing Systems (NeurIPS)*, 2021.

[71] Z. Wang, A.C. Bovik, H.R. Sheikh, and E.P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600–612, 2004.

[72] J. Wei and S. Leutenegger. GSfusion: Online RGB-D Mapping Where Gaussian Splatting Meets TSDF Fusion. *IEEE Robotics and Automation Letters (RA-L)*, 9(12):11865–11872, 2024.

[73] T. Whelan, S. Leutenegger, R.S. Moreno, B. Glocker, and A. Davison. ElasticFusion: Dense SLAM Without A Pose Graph. In *Proc. of Robotics: Science and Systems (RSS)*, 2015.

[74] L. Wiesmann, T. Guadagnino, I. Vizzo, N. Zimmerman, Y. Pan, H. Kuang, J. Behley, and C. Stachniss. LocNDF: Neural Distance Field Mapping for Robot Localization. *IEEE Robotics and Automation Letters (RA-L)*, 8(8):4999–5006, 2023.

[75] L. Wiesmann, T. Läbe, L. Nunes, J. Behley, and C. Stachniss. Joint Intrinsic and Extrinsic Calibration of Perception Systems Utilizing a Calibration Environment. *IEEE Robotics and Automation Letters (RA-L)*, 9(10):9103–9110, 2024.

[76] L. Wiesmann, E. Marks, S. Gupta, T. Guadagnino, J. Behley, and C. Stachniss. Efficient LiDAR Bundle Adjustment for Multi-Scan Alignment Utilizing Continuous-Time Trajectories. *arXiv preprint*, arXiv:2412.11760, 2024.

[77] L. Wu, C.L. Gentil, and T. Vidal-Calleja. VDB-GPDF: Online Gaussian Process Distance Field with VDB Structure. *IEEE Robotics and Automation Letters (RA-L)*, 10(1):374–381, 2024.

[78] Y. Xie, Z. Huang, J. Wu, and J. Ma. GS-LIVM: Real-Time Photo-Realistic LiDAR-Inertial-Visual Mapping with Gaussian Splatting. *arXiv preprint*, arXiv:2410.17084, 2024.

[79] Q. Xu, Z. Xu, J. Philip, S. Bi, Z. Shu, K. Sunkavalli, and U. Neumann. Point-NeRF: Point-Based Neural Radiance Fields. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[80] Y. Yan, H. Lin, C. Zhou, W. Wang, H. Sun, K. Zhan, X. Lang, X. Zhou, and S. Peng. Street Gaussians for Modeling Dynamic Urban Scenes. In *Proc. of the Europ. Conf. on Computer Vision (ECCV)*, 2024.

[81] Z. Yang, Y. Chen, J. Wang, S. Manivasagam, W.C. Ma, A.J. Yang, and R. Urtasun. UniSim: A Neural Closed-Loop Sensor Simulator. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2023.

[82] W. Yifan, F. Serena, S. Wu, C. Öztireli, and O. Sorkine-Hornung. Differentiable Surface Splatting for Point-based Geometry Processing. *ACM Trans. on Graphics (TOG)*, 38(6):1–14, 2019.

[83] M. Yu, T. Lu, L. Xu, L. Jiang, Y. Xiangli, and B. Dai. GSDF: 3DGS meets SDF for Improved Rendering and Reconstruction. In *Proc. of the Conf. on Neural Information Processing Systems (NeurIPS)*, 2024.

[84] Z. Yu, T. Sattler, and A. Geiger. Gaussian Opacity Fields: Efficient Adaptive Surface Reconstruction in Unbounded Scenes. *ACM Trans. on Graphics (TOG)*, 43(6):1–13, 2024.

[85] B. Zhang, C. Fang, R. Shrestha, Y. Liang, X. Long, and P. Tan. RaDe-GS: Rasterizing Depth in Gaussian Splatting. *arXiv preprint*, arXiv:2406.01467, 2024.

[86] G. Zhang, E. Sandström, Y. Zhang, M. Patel, L. Van Gool, and M.R. Oswald. GLORIE-SLAM: Globally optimized rgb-only implicit encoding point cloud slam. In *Proc. of the Europ. Conf. on Computer Vision (ECCV)*, 2024.

[87] J. Zhang and S. Singh. LOAM: Lidar Odometry and Mapping in Real-time. In *Proc. of Robotics: Science and Systems (RSS)*, 2014.

[88] R. Zhang, P. Isola, A.A. Efros, E. Shechtman, and O. Wang. The Unreasonable Effectiveness of Deep Features as a Perceptual Metric. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[89] Z. Zhang and D. Scaramuzza. A Tutorial on Quantitative Trajectory Evaluation for Visual(-Inertial) Odometry. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2018.

[90] C. Zhao, S. Sun, R. Wang, Y. Guo, J.J. Wan, Z. Huang, X. Huang, Y.V. Chen, and L. Ren. TCLC-GS: Tightly coupled lidar-camera gaussian splatting for surrounding autonomous driving scenes. In *Proc. of the Europ. Conf. on Computer Vision (ECCV)*, 2024.

[91] L. Zhao, Y. Wang, and S. Huang. Occupancy-SLAM: Simultaneously Optimizing Robot Poses and Continuous Occupancy Map. In *Proc. of Robotics: Science and Systems (RSS)*, 2022.

[92] Y. Zheng, X. Chen, Y. Zheng, S. Gu, R. Yang, B. Jin, P. Li, C. Zhong, Z. Wang, L. Liu, et al. GaussianGrasper: 3D Language Gaussian Splatting for Open-vocabulary Robotic Grasping. *IEEE Robotics and Automation Letters (RA-L)*, 9(9):7827–7834, 2024.

[93] X. Zhong, Y. Pan, J. Behley, and C. Stachniss. SHINE-Mapping: Large-Scale 3D Mapping Using Sparse Hierarchical Implicit Neural Representations. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2023.

[94] X. Zhong, Y. Pan, C. Stachniss, and J. Behley. 3D LiDAR Mapping in Dynamic Environments using a 4D Implicit Neural Representation. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2024.

[95] X. Zhou, Z. Lin, X. Shan, Y. Wang, D. Sun, and M.H. Yang. DrivingGaussian: Composite Gaussian Splatting

for Surrounding Dynamic Autonomous Driving Scenes. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2024.

[96] L. Zhu, Y. Li, E. Sandström, S. Huang, K. Schindler, and I. Armeni. LoopSplat: Loop Closure by Registering 3D Gaussian Splats. In *Proc. of the Intl. Conf. on 3D Vision (3DV)*, 2025.

[97] X. Zuo, P. Samangouei, Y. Zhou, Y. Di, and M. Li. FMGS: Foundation model embedded 3d gaussian splatting for holistic 3d scene understanding. *Intl. Journal of Computer Vision (IJCV)*, 130:1–17, 2024.

[98] M. Zwicker, H. Pfister, J. Van Baar, and M. Gross. EWA volume splatting. In *Proc. of Visualization*, 2001.