S^{2}KAN-SLAM: Elastic Neural LiDAR SLAM with SDF Submaps and Kolmogorov-Arnold Networks

Zhong Wang[1], Lin Zhang[2], *Senior Member, IEEE*, and Hesheng Wang[1*], *Senior Member, IEEE*

***Abstract*—Traditional LiDAR SLAM approaches prioritize localization over mapping, yet high-precision dense maps are essential for numerous applications involving intelligent agents. Recent advancements have introduced methods leveraging neural fields to enhance mapping capabilities; however, these approaches still face several limitations. Firstly, concerning scene representation, they typically employ neural fields with high-dimensional features and multi-layer perceptron decoders utilizing non-continuous activation functions. This results in low learning efficiency and challenges in capturing high-frequency signals. Secondly, in terms of scene organization, these methods often treat the entire scene as a singular neural field, leading to inefficiencies, inflexibility, and difficulties in rectifying accumulated errors when mapping large-scale environments over extended periods. To tackle the first issue, we propose a lightweight continuous SDF regression approach by encoding the scene in single-valued embeddings and decoding SDF values from a Kolmogorov-Arnold Network. By minimizing discrepancies in measuring range, sampling distance, and decoded SDF values, we facilitate iterative frame-to-model tracking and bundle adjustment neural mapping. To mitigate the second challenge, we propose structuring the whole scene into multiple neural SDF submaps. By establishing node-node, node-submap, and loop closure constraints into a global pose graph, the system can create dense neural maps with global consistency across large-scale scenes. Experimental evaluations in both real-world and simulated settings indicate that our system achieves superior mapping completeness and accuracy, enhanced learning efficiency, reduced memory consumption, and greater flexibility compared to its counterparts.**

***Index Terms*—LiDAR SLAM, Kolmogorov-Arnold Network**

I. Introduction

Simultaneous localization and mapping (SLAM) is the foundation for intelligent agents to achieve autonomous interaction with the environment. LiDAR is one of the most common sensors used for building SLAM systems outdoor.

Traditional LiDAR SLAM systems are usually more focused on localization than on mapping. That is to say, their mapping is primarily for pose tracking, often resulting in sparse point cloud maps or feature maps. For example, the implicit maps of the LOAM series solutions consist of feature lines and planes, and their explicit maps are only sparse point clouds aggregated based on the estimated poses [1]–[3]. It should be noted that such a sparse representation is not suitable for downstream tasks such as autonomous navigation and augmented reality, which desire the SLAM system to output a dense map. While a few LiDAR SLAM methods based on dense occupancy grid representations can produce dense 3D maps, they are merely discrete occupancy probability encodings of the actual scene, limited by the given resolution, and cannot be used to generate high-resolution mesh maps [4], [5].

In recent years, novel view synthesis techniques using implicit neural representations have opened new doors for dense 3D reconstruction [6]–[10]. Based on such techniques, a series of SLAM systems represent the scene as an implicit neural radiance field and aim to simultaneously estimate the carrier pose and the neural scene [11]–[14]. Although such a representation has the advantage of being continuous and compact, its slow volume rendering process becomes the efficiency bottleneck. To address this issue, some methods compromise by explicitly representing the scene as a dense 3D grid, and achieve continuous scene representation through interpolation of the high-dimensional features at the grid vertices, greatly improving the volume rendering efficiency [15], [16]. However, these methods are mostly designed for RGB/RGB-D inputs and indoor room-scale scenes. When the input is a sparse LiDAR point cloud and the working environment is an outdoor large-scale scene, although very few methods have preliminarily attempted to migrate the neural SLAM techniques developed for indoor scenes to the outdoor case, there are still issues with scene representation and scene organization [17], [18].

In terms of scene representation, a common practice in existing methods with explicit grid representations is to attach high-dimensional features to the grid vertices and use composite linear interpolation to obtain features within the grid, which are then input to a multi-layer perceptron (MLP) to decode the corresponding RGB, depth, or signed distance function (SDF) values [15], [19]. This paradigm has been verified to be effective in indoor scenes, but when coming to the outdoor case, as the scale of the scene expands, this representation requires maintaining a large number of learnable parameters, which becomes a bottleneck for learning efficiency. In addition, research has shown that the discontinuous activation function in MLP also hinders their decoding of higher frequency features [20].

Concerning the scene organization, neural representation-

This work was supported in part by the National Natural Science Foundation of China under Grant 62225309, U24A20278, 62361166632, U21A20480, 62403311, and 62403308; in part by the Postdoctoral Fellowship Program of CPSF under Grant GZC20241013, in part by the China Postdoctoral Science Foundation under Grant 2024M761973 (*Corresponding author: Hesheng Wang.*)

[1]Zhong Wang and Hesheng Wang are with Department of Automation, Key Laboratory of System Control and Information Processing of Ministry of Education, State Key Laboratory of Avionics Integration and Aviation System-of-Systems Synthesis, Shanghai Jiao Tong University, Shanghai 200240, China (email: {cszhongwang, wanghesheng}@sjtu.edu.cn).

[2]Lin Zhang is with the School of Computer Science and Technology National Pilot Software Engineering School with Chinese Characteristics, Tongji University, Shanghai 201804, China (email: cslinzhang@tongji.edu.cn).

based LiDAR SLAM methods often represent the whole scene as an octree [18], [21], which can lead to two problems in large-scale scenes. First, although the octree can relatively compactly organize the scene information, as the scene expands and the octree grows, the traversal of the entire tree will gradually slow down, which impairs the efficiency and stability of the SLAM system. Second, it is well known that the front-end of the SLAM system is plagued by accumulated errors. To mitigate this impact, the usual practice is to construct global constraints through loop closure detection in the backend and perform global joint optimization. Unfortunately, representing the entire scene as a tree will cause the global optimization to be associated with a massive number of parameters in large-scale scenes, making the problem intractable.

To address the aforementioned problems, in this paper, we propose a LiDAR **SLAM** based on octree **S**ubmaps and direct **S**DF regression via Kolmogorov-Arnold Networks (**KAN**) decoding [22], termed as **S^{2}KAN-SLAM**. We represent the scene as multiple octrees (submaps), where the leaf nodes of each octree store the actually hit grids in the 3D space. The SDF values within the voxel are determined by tri-linear interpolation of the single-valued field embeddings at the eight vertices of the voxel, and are decoded by a simple two-layer KAN. This direct SDF regression approach can greatly reduce the number of learnable parameters and eliminates the need to dynamically maintain the parameter table. At the front-end, we estimate the pose of the LiDAR scan by aligning it with the currently active submap in a frame-to-model tracking manner. Based on this result, the input point cloud is first de-skewed and then jointly optimized with the submap to update the signed distance fields. After a certain number of frames accumulated in a submap, we create a new submap to keep the scale of the scene to be optimized under control. Meanwhile, to ensure favorable consistency between submaps, we maintain sufficient overlap between them. That is, when the point cloud is in the overlap region, we will jointly optimize the two associated submaps. Furthermore, to achieve global consistent mapping, we send the submaps established by the front-end to the backend, where we perform loop closure detection and construct a global pose graph with node-to-node, node-to-submap, and loop closure constraints, thereby eliminating accumulated errors and enabling convenient and flexible global map updates.

We evaluate the performance of the proposed scheme on datasets from simulated environments as well as the real world. The results show that our pipeline using single-valued SDF field representation and KAN decoder achieves higher completeness and more accurate details in mapping performance relative to SOTA's neural representation; our submap-based management strategy enables strong flexibility of S^{2}KAN-SLAM to eliminate accumulated errors and adjust submap poses in a timely manner to achieve global consistent localization and dense mapping for large-scale scenes. In addition, this strategy also enables the system to show advantages in terms of memory utilization and learning efficiency.

To summarize, our contributions are threefold:

1) We present a novel approach that leverages Kolmogorov-Arnold Networks to achieve lightweight continuous SDF regression. This method involves direct interpolation and decoding of single-valued embeddings from octree vertices, eliminating the necessity for high-dimensional features. Consequently, it leads to a substantial reduction in the number of parameters to be trained and simplifies the reconstruction of intricate high-frequency details.
2) We develop an elastic dense neural LiDAR SLAM system by organizing the whole scene into multiple single-valued SDF submaps. The system comprises a pose graph at the backend incorporating node-to-node, node-to-submap, as well as loop closure constraints. Such designs make our S^{2}KAN-SLAM achieve globally consistent dense reconstruction for expansive outdoor environments.
3) We conduct extensive experiments on both simulated and real-world datasets, which demonstrate the effectiveness of our proposed method. To benefit the community, we also release all the relevant source codes and data to enable interested audiences to easily reproduce our results, which will be made publicly available upon the acceptance of this paper.

The remainder of this article is organized as follows. Sec. II introduces related studies. Details of the proposed S^{2}KAN-SLAM are presented in Sec. III. Experimental results are reported in Sec. IV. Finally, Sec. V concludes the paper.

II. Related Work

In this part, we first review the LiDAR odometry/SLAM systems, including traditional ones and learning-based ones, which are closely relevant to our work.

A. Traditional LiDAR SLAM

Traditional LiDAR SLAM methods can be broadly categorized based on their map representation manners: sparse and dense. The former class is typically more focused on localization, where mapping primarily serves the purpose of pose estimation.

One of the most direct sparse approaches is to estimate the pose between frames or from a frame to the point cloud map using the seminal Iterative Closest Point (ICP) [23] point cloud registration method. To address the challenge of point cloud deskewing, Dellenbach *et al.* [24] propose estimating both the starting and ending poses during the ICP iterative optimization process. Vizzo *et al.* [25] further conduct in-depth analysis of the ICP submodules and parameter settings, providing useful guidelines for accurate and robust ICP registration.

Compared to traditional direct sparse methods, an alternative approach involves the initial extraction of features followed by inter-frame registration based on geometric feature associations as visual SLAM systems usually do [26]. For instance, LOAM [1] is a prominent example that extracts edge and planar points as features. It iteratively determines the relative pose between frames by minimizing geometric point-to-line and point-to-plane distances at a high frequency. Additionally, it addresses cumulative errors through frame-to-map registration at a lower frequency. Expanding upon this

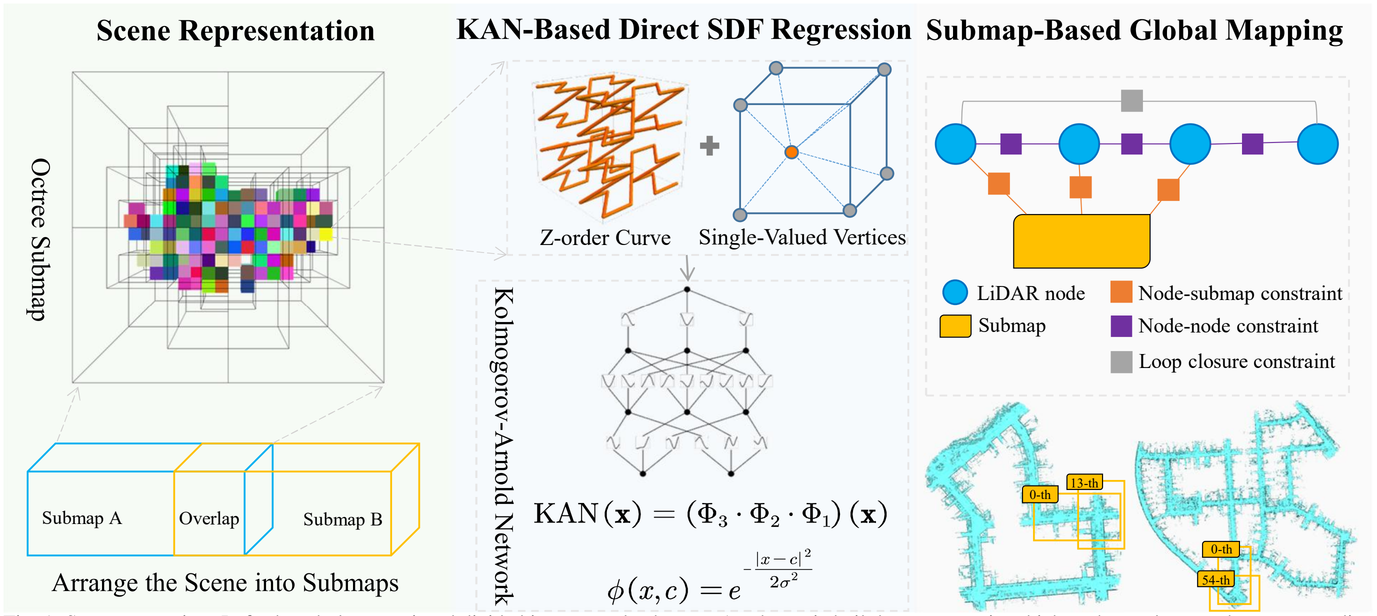
Scene Representation
Octree Submap
Submap A
Overlap
Submap B
Arrange the Scene into Submaps
KAN-Based Direct SDF Regression
Z-order Curve
Single-Valued Vertices
Kolmogorov-Arnold Network
$\mathrm{KAN}(\mathbf{x}) = (\Phi_3 \cdot \Phi_2 \cdot \Phi_1)(\mathbf{x})$
$\phi(x,c) = e^{-\frac{|x-c|^2}{2\sigma^2}}$
Submap-Based Global Mapping
LiDAR node
Submap
Node-submap constraint
Node-node constraint
Loop closure constraint
0-th
13-th
0-th
54-th

Fig. 1: System overview. Left: the whole scene is subdivided into several submaps. A submap is built by an octree in which each voxel stores the corresponding neural embeddings of that spatial location. Middle: when rendering a sampled point, its feature is first interpolated from the features of neighboring vertices and subsequently fed into a Kolmogorov-Arnold network to decode the resulting SDF value. Right: the finished submap is sent to the backend, in which loop closures are detected and a pose graph is established to achieve elastic global consistent dense mapping.

foundation, a range of feature-based LiDAR SLAM methodologies, such as LeGO-LOAM [2], LOAM-livox [27], and F-LOAM [3], enhance pose estimation accuracy and efficiency through improved feature extraction techniques and optimized registration strategies.

Unlike the sparse approaches that prioritize localization, dense mapping methods aim to generate detailed and complete representations of the environment, which can be leveraged for a wider range of tasks beyond just pose estimation. In a semi-dense manner, SUMA [28] represents the map as 3D surfels and estimates the pose through frame-to-surfel map registration. Vizzo *et al.* [29] construct dense mesh maps from point clouds and register by minimizing the geometric distance from points to the mesh. There are also some schemes that represent the map as a probabilistic occupancy grid map and estimate the pose by maximizing the hit probability [4], [30]. However, these are mostly limited to the 2D LiDAR SLAM scenario.

B. *Learning-based LiDAR SLAM*

With the rapid development of deep learning, learning-based LiDAR SLAM has also made great progress in recent years. Early learning-based schemes focus on replacing traditional LiDAR SLAM modules with data-driven approaches, such as feature extraction and registration [31], [32], loop closure detection [33]–[35]. To fully exploit the potential of these methods, some schemes integrate feature encoding, dynamic object removal, and pose regression modules into a single network and achieve comparable results to traditional geometry-based methods through end-to-end training [36], [37].

In recent years, the great success of implicit neural fields in novel view synthesis has spawned a series of SLAM schemes based on neural scene representations. Typical examples include NICE-SLAM [12] based on implicit NeRF [6] representation, Vox-Fusion [15] based on explicit SDF representation, and Mono-GS [38] and SplaTAM [39] based on explicit 3DGS [40] representation. Inspired by these neural representation-based RGB/RGBD schemes, some methods also leverage neural representations for map construction using LiDAR data, such as SHINE-Mapping [21] and NF-Atlas [41]. However, there is still limited research on simultaneous localization and mapping using such representations.

The closest to our approach is NeRF-LOAM [18] proposed by Deng *et al.* It represents the entire scene as an octree, with the tree nodes storing the corresponding high-dimensional neural features, and learns the scene representation by decoding these features. In contrast, our approach uses direct regression based on Kolmogorov-Arnold Network (KAN) [22] to reduce the scene learning parameters, and the neural submap representation makes the map more flexible and facilitates the construction of globally consistent maps. By combining the strengths of learning-based and neural representation-based methods, our work advances the state-of-the-art in dense LiDAR SLAM for large-scale outdoor environments.

III. Methodology

A. *System Overview*

As illustrated in Fig. 1, our approach for achieving elastic mapping involves the partitioning of the scene into multiple submaps. Each submap encompasses learnable single-valued SDF features that are associated with the vertices of the submap, organized within a sparse voxel octree structure. The feature of a specific point within the spatial domain is obtained through interpolation from the features of adjacent vertices in the octree submap. During the tracking process, sampling is performed along LiDAR rays from the sparse voxel octree. Subsequently, the SDF features of the sampled points are queried, and the SDF values of the corresponding points are decoded utilizing a Kolmogorov-Arnold network. The LiDAR pose is iteratively optimized by minimizing losses based on sampling distances, SDF values, and range values. During the mapping phase, optimization is conducted jointly on the point cloud pose and SDF field. The optimized submap (SDF field) is then forwarded to the backend for loop closure detection and constraint formulation. Ultimately, a globally consistent dense map is generated from the SDF fields by adjusting the submaps with optimized poses derived from the pose graph.

B. Scene Representation

1) Octree Submaps: Existing LiDAR SLAM schemes for neural field representation usually represent the entire scene in a single map. In indoor small-scale scenarios, this choice is beneficial to reduce the system state drift. But in outdoor scenarios, it will bring two problems. One is that as the range of the map grows, the resources occupied by the system increase and the computational efficiency decreases, which is detrimental to the stability of the system. The second is that this representation is not conducive to the pose adjustment and globally consistent map construction when loop closure detected.

In order to solve the aforementioned problems, we divide the scene and represent it with a series of submaps. For a specific submap, we use it to store the scene features in the current local environment. To obtain faster learning efficiency, unlike the NeRF-like scheme that implicitly represents the scene as an MLP, we divide the scene into a voxel grid at a certain resolution. However, it is impractical to densely store such a large number of voxels in an outdoor scene. Therefore, we borrow sparse voxel octree to organize them. We dynamically grow nodes in the octree only when the voxel is hit by a LiDAR point.

Moreover, although the sparse octree compactly represents the scene, further compression is still required for the storage of 3D voxels. To achieve this, we organize all voxel coordinates within a submap with the help of a Z-order curve (also known as Morton code) [42], thus compressing a 3-dimensional vector into 1-dimensional. When the coordinates of a specific node are queried from the octree, they are decoded from the corresponding code without observable delay.

2) Single-valued Neural SDF Field: Each node in the octree is characterized by the 8 vertex features attached to it. It is worth noting that the vertex features are shared among neighboring voxels in order to avoid artifacts at the edges as well as to reduce the learning parameters. Typically, in neural scene representations such as VoxFusion [15], TensoRF [19], etc., voxels are usually characterized by high-dimensional feature vectors, supplemented by an MLP decoder to obtain features at the corresponding points, and ultimately ray rendering to obtain target values such as color, opacity, etc. However, since the SDF field characterizes the distance of a point in space from its nearest surface, we believe that this should be a single-valued mapping process without the need for high-dimensional characterization as well as a complex decoding process and integral rendering. Therefore, our SDF field consists of only trainable single-valued features. Specifically, when a point falls inside the voxel, its corresponding feature is obtained by:

$$f_{\boldsymbol{p}} = \text{TriInt}(\boldsymbol{p}, f_0, f_1, \ldots, f_7), \tag{1}$$

where $\text{TriInt}(\cdot)$ denotes tri-linear interpolation of the eight vertices' single-valued features $(f_0, f_1, \ldots, f_7)$ of its nearest neighbors.

3) KAN-Based Direct SDF Regression: After obtaining the SDF feature of a point $\boldsymbol{p}$, we need to further decode its actual SDF value from it. Since the SDF field is continuous, we expect to construct a decoder that accepts a single-valued input and produces a continuous output. Although MLPs are widely used in neural field decoding, Sitzmann *et al.* showed that the discontinuous activation function in MLPs makes it difficult to capture high-frequency information [20]. Thereby additional positional coding is often required in practice. To overcome this limitation, we are committed to finding a more convenient and applicable decoder for our case.

Recently Liu *et al.* [22] constructed a learnable KAN based on Kolmogorov-Arnold theory, which is a promising decoder of continuous functions since it learns the combination coefficients of continuous functions instead of the weights of the node values without the need of discontinuous activation functions. Therefore, we resort to KAN to decode SDF values from the single-valued SDF field.

A KAN is defined by a vector of integers:

$$[n_0, \ldots, n_l, \ldots, n_L], \tag{2}$$

where n_l denotes the number of neurons at the l-th layer. The activation value of the j-th neuron at layer $l+1$ is the sum of post activations from the l-th layer:

$$x_{l+1,j} = \sum_{i=1}^{n_l} \phi_{l,j,i}(x_{l,i}),\ j = 1, \ldots, n_{l+1}, \tag{3}$$

where $\phi_{l,j,i}(\cdot)$ denotes the activation function that connects the i-th neuron at the l-th layer and the j-th neuron at the $l+1$-th layer. It can be seen that one of the main differences between KAN and MLP is that its edges are learnable functions rather than weights.

The above KAN layer can also be expressed in matrix form:

$$\begin{aligned}\boldsymbol{x}_{l+1} &= \begin{bmatrix} \phi_{l,1,1}(\cdot) & \phi_{l,1,2}(\cdot) & \cdots & \phi_{l,1,n_l}(\cdot) \\ \phi_{l,2,1}(\cdot) & \phi_{l,2,2}(\cdot) & \cdots & \phi_{l,2,n_l}(\cdot) \\ \vdots & \vdots & \vdots & \\ \phi_{l,n_{l+1},1}(\cdot) & \phi_{l,n_{l+1},2}(\cdot) & \cdots & \phi_{l,n_{l+1},n_l}(\cdot) \end{bmatrix} \boldsymbol{x}_l \\ &= \boldsymbol{\Phi}(\boldsymbol{x}_l),\end{aligned} \tag{4}$$

where $\boldsymbol{\Phi}(\cdot)$ is the function matrix of the l-th KAN layer. Simply stacking L layers produces a KAN network:

$$\text{KAN}(\boldsymbol{x}) = (\boldsymbol{\Phi}_{L-1} \cdot \boldsymbol{\Phi}_{L-2} \Phi_1 \cdot \boldsymbol{\Phi}_0)\boldsymbol{x}, \tag{5}$$

where $\boldsymbol{x} \in \mathbb{R}^{n_0}$ and $\text{KAN}(\boldsymbol{x}) \in \mathbb{R}^{n_{L-1}}$.

In the original KAN, the edges of the network (i.e., the learnable functions) consist of a series of B-splines. Instead, in this paper, we empirically use Gaussian radial basis functions:

$$\phi\,(x, c) = e^{-\frac{|x-c|^2}{2\sigma^2}}, \tag{6}$$

where x is the neuron activation; c and σ are the parameters of the function to be learned. We will demonstrate in the experiment that such a function is more suitable for SDF decoding than B-splines.

Since our submap is represented as a single-valued SDF field, and also, since the SDF values of the points to be decoded are scalar, the input and output dimensions of our KAN are both 1. After experiments, we set the network with two intermediate hidden KAN layers containing 64 neurons at each layer to seek a balance between efficiency and accuracy.

C. Mapping within Submap

Given the current submap $\mathcal{S}$ and a point cloud $\mathcal{P} = \{\mathcal{D}, \mathcal{R}\}$ with measuring directions $\mathcal{D}$, ranges $\mathcal{R}$, and its initial estimated pose $\boldsymbol{T}$, the mapping problem is to solve the following maximized likelihood probability problem:

$$\{\mathcal{S}, \boldsymbol{T}\}^* = \arg\max_{\mathcal{S},\boldsymbol{T}} p(\mathcal{S}, \boldsymbol{T}|\mathcal{P}) \tag{7}$$

$$= \arg\max_{\mathcal{S},\boldsymbol{T}} \frac{p(\mathcal{S}, \boldsymbol{T})p(\mathcal{P}|\mathcal{S}, \boldsymbol{T})}{p(\mathcal{P})} \tag{8}$$

$$= \arg\max_{\mathcal{S},\boldsymbol{T}} p(\mathcal{S})p(\boldsymbol{T})p(\mathcal{P}|\mathcal{S}, \boldsymbol{T}). \tag{9}$$

In our case, solving this problem is equivalent to seeking to minimize the difference in distance between the sampled points along rays and the SDF field, i.e:

$$\mathcal{L}_s = \sum_{k=0}^{K-1} \|r(\boldsymbol{p}_k) + \frac{\text{KAN}(\mathcal{S}(\boldsymbol{p}_k))}{\xi} - \mathcal{R}(\mathcal{D}(\boldsymbol{p}_k))\|_2, \tag{10}$$

$$\xi = \begin{cases} \sin(\theta), & \text{if } \mathcal{D}(\boldsymbol{p}_k)) \in \mathcal{D}^G \\ 1 & \text{otherwise} \end{cases}, \tag{11}$$

where $r(\cdot)$ means the range along the ray to the sampled point $\boldsymbol{p}_k$, $\mathcal{D}(\cdot)$ returns the direction of the given point, and $\mathcal{R}(\cdot)$ returns the measuring range of the given direction; $\mathcal{D}^G$ represents the rays whose measured ending points belong to the ground; θ is the acute angle between the ray and the ground plane; and $\mathcal{S}(\boldsymbol{p}_k)) = f_{\boldsymbol{p}_k}$. The reason for treating ground and non-ground points differently is that in outdoor environments, when the tilt angle is very small, phenomena such as occlusion can produce ambiguity in the SDF observations at different locations [18].

In outdoor environments, the presence of dynamic objects not only affects the pose tracking but also degrades the mapping quality, so we take the loss term of free space into account:

$$\mathcal{L}_f = \sum_{k=0}^{K-1} \|\frac{\text{KAN}(\mathcal{S}(\boldsymbol{p}_k))}{\xi} - \tau\|_2, \tag{12}$$

where τ is the truncation of the SDF field. This loss term aims to constrain the space between the center of LiDAR and the sampled point to be free. Note that only the points which fall outside the truncation region are considered.

In addition, since the SDF field inscribes the distance from a point in space to its nearest face element, we would like to have the direction of change coincide with the fastest descent of the field for points located within the truncated region, hence the introduction of Eikonal loss:

$$\mathcal{L}_e = \sum_{k=0}^{K-1} \|\frac{\partial \text{KAN}(\mathcal{S}(\boldsymbol{p}_k))}{\partial \boldsymbol{p}_k} - 1\|_2. \tag{13}$$

Finally, we define the loss function as:

$$\mathcal{L} = \lambda_s \mathcal{L}_s + \lambda_f \mathcal{L}_f + \lambda_e \mathcal{L}_e, \tag{14}$$

where λ_s, λ_f, and λ_e are the weights of the corresponding losses.

D. Frame-to-Model Tracking

A reasonable initial pose is necessary for pose tracking. Due to the lack of additional sensors, we use a constant motion model to predict the current pose. Assuming the poses at time $t-2$ and $t-1$ are $\boldsymbol{T}_{t-2}$ and $\boldsymbol{T}_{t-1}$ respectively, the initial pose value at the current time can be obtained by the following equation:

$$\tilde{\boldsymbol{T}}_t = \boldsymbol{T}_{t-1}\Delta\boldsymbol{T} = \boldsymbol{T}_{t-1} \cdot \boldsymbol{T}_{t-2}^{-1} \cdot \boldsymbol{T}_{t-1} \ . \tag{15}$$

For LiDAR in motion, there is distortion in the observed points, and using it directly for pose estimation and map updates will accelerate the drift of the system. However, distortion correction and pose optimization have become a chicken egg problem. In addition, the computational burden of correcting point clouds in each iteration is difficult to bear. Therefore, we adopt a two-stage distortion reduction strategy, which first distorts the point cloud using the calculated initial pose, and then distorts the point cloud again based on the optimized pose after the pose estimation is completed. Finally, the undistorted point cloud will be used for map updates.

Specifically, assuming that the sampling interval of a frame of the point cloud is δt and the relative pose is $\Delta\boldsymbol{T} = [\Delta\boldsymbol{R}|\Delta\boldsymbol{t}]$, the average angular velocity and the average linear velocity during the sampling interval are:

$$\boldsymbol{v} = \frac{\Delta\boldsymbol{t}}{\delta t},\ \boldsymbol{\omega} = \frac{\text{Log}(\Delta\boldsymbol{R})}{\delta t}, \tag{16}$$

where $\text{Log}(\cdot)$ denotes the logarithmic map from $SO(3)$ to $so(3)$. Assuming that point $\boldsymbol{p}_i$ is sampled with a timestamp s_i relative to the current frame's start moment, the sampled pose corresponding to $\boldsymbol{p}_i$ can be obtained as:

$$\boldsymbol{p}_i^* = \text{Exp}(s_i\boldsymbol{\omega})\boldsymbol{p}_i + s_i\boldsymbol{v}, \tag{17}$$

where $\text{Exp}(\cdot)$ remaps an element in $so(3)$ back to $SO(3)$.

When only pose estimation is conducted, we fix the current submap, and according to Eq. 10, the optimization problem simplifies to:

$$\hat{\boldsymbol{T}} = \arg\max_{\boldsymbol{T}} p(\mathcal{P}|\mathcal{S}, \boldsymbol{T}). \tag{18}$$

At this time, we choose to minimize $\mathcal{L}_s$ and iteratively optimize the pose.

E. Global Consistent Mapping

Within the submap range, the combination of joint mapping and frame-to-model tracking facilitates the creation of highly precise maps. However, to address the issue of cumulative error that may arise over prolonged operation within expansive scenes, a pose graph is constructed in the backend. This serves the purpose of mitigating accumulated errors and facilitating elastic global consistent mapping.

1) Loop Constraint Construction: The formulation of loop closure constraints encompasses two primary components: loop closure detection and geometric registration. In recent years, substantial advancements have been made in loop closure detection, with the prevalent utilization of methodologies such as Scan-Context [43] to identify potential loop closures. To cater to diverse scenarios and ensure a high recall rate, a relatively lenient similarity threshold is employed for loop closure screening, followed by a rigorous geometric validation process to confirm genuine loop closures. While geometric validation traditionally relies on Iterative Closest Point (ICP) algorithms, challenges can arise when significant pose disparities exist between the loop closure frames. To address this issue, an initial estimate of the 3 degrees of freedom is determined by aligning the bird's-eye views of the point clouds, subsequently refined through ICP alignment to establish the final loop closure constraints.

2) Pose Graph Optimization: In order to facilitate the construction of a globally consistent map, our pose graph not only needs to contain node-to-node and loop closure constraints, but also needs to consider the constraints between frame nodes and submaps. In this way, when the optimization is completed, we can easily achieve the construction of the global map by adjusting the submap's poses. To this end, the variables to be optimized for the whole system are defined as:

$$\mathcal{T}_f = \{\boldsymbol{T}_{f0}, \boldsymbol{T}_{f1}, \ldots, \boldsymbol{T}_{fn}\}, \tag{19}$$

$$\mathcal{T}_s = \{\boldsymbol{T}_{s0}, \boldsymbol{T}_{s1}, \ldots, \boldsymbol{T}_{sm}\}, \tag{20}$$

where $\mathcal{T}_f$ contains the poses of all frames and $\mathcal{T}_s$ envolves all the submap poses.

A node-to-node or loop closure constraint is a measurement of one frame $\boldsymbol{T}_{fj}$ from another frame $\boldsymbol{T}_{fi}$. Similarly, a node-to-submap constraint is the measurement of $\boldsymbol{T}_{fi}$ from its corresponding submap $\boldsymbol{T}_{sk}$. The measured poses by odometry estimation, loop scan registration, and scan-to-submap registration, are denoted by $\bar{\boldsymbol{z}}_{fi}^{f,i+1}$, $\bar{\boldsymbol{z}}_{fj}^{fi}$, and $\bar{\boldsymbol{z}}_{fi}^{sk}$, with measuring covariances $\boldsymbol{Q}_o$, $\boldsymbol{Q}_l$, and $\boldsymbol{Q}_s$, respectively. For any actual poses of $\boldsymbol{T}_{fi}$ and $\boldsymbol{T}_{fj}$, their relative offset can be obtained via:

$$\boldsymbol{T}_{fj}^{fi} = \boldsymbol{T}_{fi}^{-1}\boldsymbol{T}_{fj} = \begin{bmatrix} \boldsymbol{R}_{fj}^{fi} & \boldsymbol{t}_{fj}^{fi} \\ \boldsymbol{0} & 1 \end{bmatrix}, \tag{21}$$

$$h(\boldsymbol{T}_{fi}, \boldsymbol{T}_{fj}) = \begin{cases} \boldsymbol{t}_{fj}^{fi} \\ \mathrm{Log}(\boldsymbol{R}_{fj}^{fi}). \end{cases} \tag{22}$$

Likewise, we can get the relative offset between a frame and a submap, $h(\boldsymbol{T}_{sk}, \boldsymbol{T}_{fi})$. The errors of the system are thus obtained by:

$$\boldsymbol{e}_o = \bar{\boldsymbol{z}}_{f,i+1}^{fi} - h(\boldsymbol{T}_{fi}, \boldsymbol{T}_{f,i+1}), \tag{23}$$

$$\boldsymbol{e}_l = \bar{\boldsymbol{z}}_{fj}^{fi} - h(\boldsymbol{T}_{fi}, \boldsymbol{T}_{fj}), \tag{24}$$

$$\boldsymbol{e}_s = \bar{\boldsymbol{z}}_{fi}^{sk} - h(\boldsymbol{T}_{sk}, \boldsymbol{T}_{fi}), \tag{25}$$

$$\mathcal{E} = \sum_{ijk} \boldsymbol{e}_o^T\boldsymbol{Q}_o^{-1}\boldsymbol{e}_o + \boldsymbol{e}_l^T\boldsymbol{Q}_l^{-1}\boldsymbol{e}_l + \boldsymbol{e}_s^T\boldsymbol{Q}_s^{-1}\boldsymbol{e}_s. \tag{26}$$

The initial values of $\boldsymbol{T}_f$ and $\boldsymbol{T}_s$ are obtained while performing odometry estimation and the aforementioned loop constraint construction. When conducting joint optimization, although a series of relevant optimization techniques emerged recently to improve the optimization robustness [44]–[48], via experiments validation, we find the classic Levenberg–Marquardt algorithm [49] works well in our case to iteratively find the optimal solution ($\mathcal{T}_f^*$ and $\mathcal{T}_s^*$) that minimizes $\mathcal{E}$.

Algorithm 1 Pipeline of S^{2}KAN-SLAM

```
Input: LiDAR frames ($\mathcal{F} = \{\mathcal{F}_0, \ldots, \mathcal{F}_n\}$).
Output: Optimized poses ($\mathcal{T}_f^*$ and $\mathcal{T}_s^*$) and submaps ($\mathcal{S}^* = \{\mathcal{S}_0^*, \ldots, \mathcal{S}_m^*\}$).
1:  for $\mathcal{F}_i \in \mathcal{F}$ do
2:      Sampling points along rays of $\mathcal{F}_i$, resulting in $\mathcal{P}$.
3:      if not initialized then
4:          Allocate voxels for $\mathcal{S}_0$ using $\mathcal{P}$.
5:          Generate single-valued embeddings $\mathcal{V}_0$ attached at the voxel vertices of $\mathcal{S}_0$.
6:          Optimize $\mathcal{V}_0$ and the KAN decoder (Eq. 5) by minimizing $\mathcal{L}_s$ (Eq. 10) of $\mathcal{P}$.
7:      else
8:          Predict the pose of the current frame ($\tilde{\boldsymbol{T}}_{fi}$) according to Eq. 15.
9:          Deskew $\mathcal{P}$ as Eq. 17, resulting in $\mathcal{P}^*$.
10:         Optimize $\tilde{\boldsymbol{T}}_{fi}$ by minimizing $\mathcal{L}_s$ (Eq. 10) of $\mathcal{P}^*$, producing $\hat{\boldsymbol{T}}_{fi}$.
11:         Insert $\mathcal{P}^*$ into the submap $\mathcal{S}_k$ using $\hat{\boldsymbol{T}}_{fi}$.
12:         Create embeddings for newly allocated voxels.
13:         Optimize the embeddings $\mathcal{V}_k$ of $\mathcal{S}_k$ and the decoder by minimizing $\mathcal{L}_s$ (Eq. 10) of $\mathcal{P}^*$.
14:     end if
15:     Add frame-to-frame constraints.
16:     Add frame-to-submap constraints.
17:     if $\mathcal{S}_k$'s last frame received then
18:         Loop closure detection using $\mathcal{F}_i$.
19:         if loop closure found then
20:             Add loop closure constraint to the pose graph.
21:             Jointly optimize the pose graph under Eq. 26, resulting in $\{\boldsymbol{T}_{f0}^*, \ldots, \boldsymbol{T}_{fi}^*\}$ and $\{\boldsymbol{T}_{s0}^*, \ldots, \boldsymbol{T}_{sk}^*\}$.
22:             Select key frames $\mathcal{K}_k$ in $\mathcal{S}_k$.
23:             Jointly optimize the embeddings of $\mathcal{S}_k$ using $\mathcal{K}_k$ under Eq. 10, obtaining $\mathcal{S}_k^*$.
24:         end if
25:     end if
26: end for
```

3) Keyframe Refinement and Mesh Generation: When the optimization of the pose graph is completed, we use the optimized poses to pick the keyframes within the submap to bundle adjusting the submap SDF field. The optimization objective is shown in Eq. 15. Different from mapping when odometry, only the SDF field is optimized at this time without optimizing the keyframe poses. Moreover, thanks to the aforementioned global pose adjustment containing frame-to-submap constraints, we can easily construct a global map from the optimized poses and submaps. When generating mesh maps, we produce dense maps with the help of the marching cube algorithm [50].

At the last, to formally describe the tracking and mapping procedure of S^{2}KAN-SLAM, we provide the pseudocode of the whole pipeline in Algorithm 1.

IV. Experiment

A. Setup

1) Datasets: In order to comprehensively assess the mapping and localization performance of the proposed S^{2}KAN-SLAM, experiments were conducted utilizing three popular datasets: the KITTI Odometry dataset [51], the Newer College dataset [52], and the MaiCity dataset [29]. The KITTI Odometry dataset comprises data captured by an in-vehicle platform in a large-scale urban setting, offering pose ground truth data derived from RTK-GPS and high-precision IMU measurements. The Newer College dataset, captured by a handheld device, provides mapping ground truth reconstructed using a high-precision Leica BLK360 scanner within a small campus courtyard environment. The MaiCity dataset is a simulated dataset featuring small street environments generated from a CAD model, facilitating mapping evaluation.

2) Implementation: We tested the relevant algorithms on a workstation equipped with an RTX-8000 GPU, a 32-core Intel(R) Xeon W-3365 CPU, and 256GB of memory. The mapping and tracking modules of S^{2}KAN-SLAM are implemented based on PyTorch. The pose graph construction and optimization are implemented using GTSAM.

3) Metrics: Average Translation Error (*ATE*). For the trajectory estimated by the algorithm, we first align it with the ground truth trajectory based on the Umeyama algorithm [53]. Then we compute the root mean square error of the aligned trajectory to assess the accuracy of pose estimation.

F-score. Let the reconstructed map be $\mathcal{M}_r$ and the ground truth map as $\mathcal{M}_g$, the mapping *precision* (*PR*) and *recall* (*RC*) can be obtained via:

$$e_{\boldsymbol{r}\to\mathcal{M}_g} = \min_{\boldsymbol{g}\in\mathcal{M}_g} \|\boldsymbol{r}-\boldsymbol{g}\|, \tag{27}$$

$$e_{\boldsymbol{g}\to\mathcal{M}_r} = \min_{\boldsymbol{r}\in\mathcal{M}_r} \|\boldsymbol{g}-\boldsymbol{r}\|, \tag{28}$$

$$PR = \frac{1}{|\mathcal{M}_r|}\sum_{\boldsymbol{r}\in\mathcal{M}_r}(e_{\boldsymbol{r}\to\mathcal{M}_g} < d), \tag{29}$$

$$RC = \frac{1}{|\mathcal{M}_g|}\sum_{\boldsymbol{g}\in\mathcal{M}_g}(e_{\boldsymbol{g}\to\mathcal{M}_r} < d), \tag{30}$$

where d is the acceptance distance threshold. More compactly, precision and recall can be expressed as a single metric [54]:

$$\textit{F-score} = \frac{2 * PR * RC}{PR + RC}. \tag{31}$$

Accuracy. Mapping accuracy refers to the average error between reconstructed points and their corresponding points on the ground truth map:

$$\textit{Accuracy} = \frac{1}{|\mathcal{M}_r|}\sum_{\boldsymbol{r}\in\mathcal{M}_r} e_{\boldsymbol{r}\to\mathcal{M}_g}. \tag{32}$$

Completeness. Mapping completeness refers to the average error between ground truth map points and their corresponding reconstructed points:

$$\textit{Completeness} = \frac{1}{|\mathcal{M}_g|}\sum_{\boldsymbol{g}\in\mathcal{M}_g} e_{\boldsymbol{g}\to\mathcal{M}_r}. \tag{33}$$

Chamfer distance. As typical mapping schemes do [18], [21], we also evaluate the $l1$ Chamfer distance (C_{l1}):

$$C_{l1} = 0.5 * (\textit{Accuracy} + \textit{Completeness}). \tag{34}$$

B. Incremental Odometry and Mapping

1) Incremental Mapping Results: The mapping performance of various algorithms was assessed using the MaiCity and Newer College datasets, which offer ground truth maps for evaluation. In conducting the reconstruction analysis, we compared the outcomes of our proposed S^{2}KAN-SLAM with traditional geometry-based VDBFusion [55], PUMA [29], and SLAMesh [58], as well as learning-based approaches such as SHINE-Mapping [21], 4dNDF [56], NKSR [57], and NeRF-LOAM [18]. Notably, SHINE-Mapping, VDBFusion, 4dNDF, and NKSR solely focus on mapping tasks and do not independently estimate poses, thus we utilized poses from the KISS-ICP odometry scheme as inputs for their mapping procedures. To ensure fair comparisons, both NeRF-LOAM and our S^{2}KAN-SLAM were provided with KISS-ICP poses for standalone mapping evaluations. Additionally, PUMA, SLAMesh, NeRF-LOAM, and our S^{2}KAN-SLAM utilized their own estimated poses for mapping tasks. Detailed results of these evaluations are reported in Table I.

It can be seen that our approach consistently attains a minimum of the top two performances across all sequences, with the best results being achieved in the majority of cases. Specifically, employing KISS-ICP poses, our S^{2}KAN-SLAM demonstrates comparable or even better performance compared with state-of-the-art approaches at Newer College. While it falls short of achieving the optimal outcome at MaiCity, the discrepancy compared to the leading solution NeRF-LOAM is deemed insignificant. Notably, our methodology excels when estimating poses concurrently, underscoring the efficacy of simultaneously estimating poses and maps in enhancing accuracy compared to solely focusing on mapping.

To enhance the visual representation of the mapping outcomes, we reconstruct mesh maps using the learned SDF and decoder and present them in Fig. 2. The first and second rows of the figure showcase the constructed maps generated by various algorithms in both real-world setting Newer College

TABLE I: Mapping statistics on MaiCity and Newer College. The results on MaiCity are obtained with an acceptance distance threshold of 10cm, while with 20cm for that on Newer College. The methods of first six rows use KISS-ICP poses and only conduct mapping, while the ones of the last four rows perform simultaneously odometry estimation and mapping. "-" indicates the results are not available.

Method	Pose	MaiCity @ d=10cm				Newer College @ d=20cm			
		Accuracy ↓	Completeness ↓	C_{l1} ↓	F-score ↑	Accuracy ↓	Completeness ↓	C_{l1} ↓	F-score ↑
SHINE-Mapping [21]	KISS-ICP	5.75	38.45	22.10	67.00	14.87	20.02	17.45	68.85
VDBFusion [55]		4.95	46.79	25.87	68.15	14.03	25.46	19.75	69.50
4dNDF [56]		5.65	36.13	20.89	71.03	15.84	$\underline{19.57}$	$\underline{16.70}$	69.99
NKSR [57]		-	-	-	-	15.67	36.87	26.27	58.57
NeRF-LOAM [18]		**4.16**	37.20	20.67	73.31	14.31	24.39	19.35	68.70
S^{2}KAN-SLAM (ours)		4.81	37.83	21.32	72.52	14.19	20.25	17.22	71.72
PUMA [29]	Odometry	7.89	$\underline{9.14}$	8.51	68.04	15.30	71.91	43.60	57.27
SLAMesh [58]		5.66	13.01	9.33	75.42	19.21	48.83	34.02	45.24
NeRF-LOAM [18]		5.69	11.23	$\underline{8.46}$	$\underline{77.26}$	**12.89**	22.21	17.55	**74.37**
S^{2}KAN-SLAM (ours)		$\underline{4.28}$	**7.47**	**5.88**	**86.75**	$\underline{13.32}$	**18.80**	**16.06**	$\underline{72.03}$

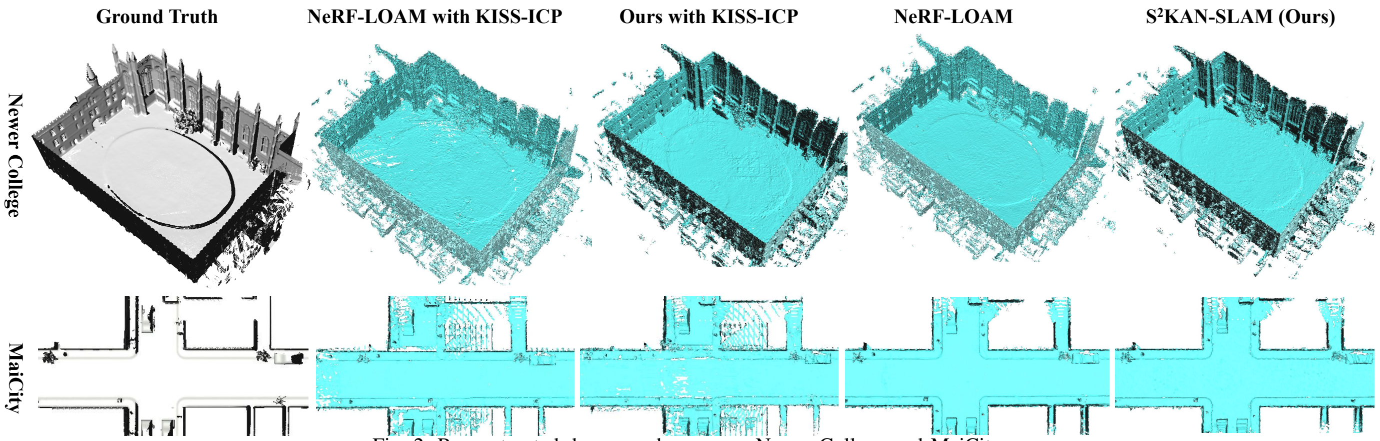

Fig. 2: Reconstructed dense mesh maps on Newer College and MaiCity.

and simulated environment MaiCity. The subfigures within each row, arranged from left to right, depict the ground truth maps, the outcomes of NeRF-LOAM with KISS-ICP, the results of S^{2}KAN-SLAM with KISS-ICP, the results of NeRF-LOAM, and the outputs of our S^{2}KAN-SLAM, respectively.

As can be seen, when utilizing the KISS-ICP poses, both our S^{2}KAN-SLAM and NeRF-LOAM exhibit voids in the mapping outputs, with ours appearing in the middle of the road while NeRF-LOAM's gaps are situated at the base of the walls. Conversely, when employing KISS-ICP poses at Newer College, our S^{2}KAN-SLAM outperforms NeRF-LOAM by presenting a more comprehensive reconstructed map characterized by a smoother terrain surface and enhanced details such as columns and wall textures. Furthermore, simultaneous map construction and pose estimation yield further enhancements in detail for both NeRF-LOAM and our scheme, with our approach demonstrating more pronounced improvements, particularly in the intricate ground features at Newer College and the delineation of trees and wall edges in MaiCity. These findings align with the quantitative data provided in Table I.

TABLE II: Tracking *ATE*s (m) on the KITTI Odometry dataset. Seq00 is the abbreviation of Sequence-00, the same for Seq05, Seq07, and Seq09.

	@100		@200		@300	
	NeRF-LOAM	Ours	NeRF-LOAM	Ours	NeRF-LOAM	Ours
Seq00	**0.21**	0.27	0.30	**0.24**	0.44	**0.27**
Seq05	**0.15**	0.19	**0.32**	0.34	**0.37**	0.41
Seq07	0.15	**0.14**	0.31	**0.29**	**0.38**	**0.38**
Seq09	0.46	**0.32**	1.10	**0.70**	1.51	**0.99**

2) Odometry Results: The tracking performance assessment was conducted on the KITTI Odometry dataset utilizing ground truth poses. Given the submap partitioning strategy employed in S^{2}KAN-SLAM, it is anticipated that the system's front-end would demonstrate sufficient accuracy within each submap. In order to validate this capability, we compared the odometry results of S^{2}KAN-SLAM with the neural field-based NeRF-LOAM for tracking sequences of 100, 200, and 300 frames on KITTI Odometry. We leave the analysis of long-term consistent localization performance in the subsequent section. The results presented in Table II indicate that our S^{2}KAN-SLAM achieves comparable or superior tracking performance relative to NeRF-LOAM in short-term tracking scenarios. This robust performance in tracking at the front-end enables focus on submap pose optimization at the backend, ultimately facilitating improved globally consistent localization and mapping outcomes.

C. Globally Consistent Localization and Mapping

We evaluate the system's global consistent localization and mapping performance on the outdoor autonomous driving dataset KITTI Odometry. We provide the results of four typical sequences with loop closures, Sequence-00, Sequence-05, Sequence-07, and Sequence-09.

1) Global Mapping: As shown in Fig. 3, we draw a panoramic top view of the finalized maps, where A∼H are the locations where loop closures occur along the vehicle's moving trajectory. The corresponding local zoom-in maps

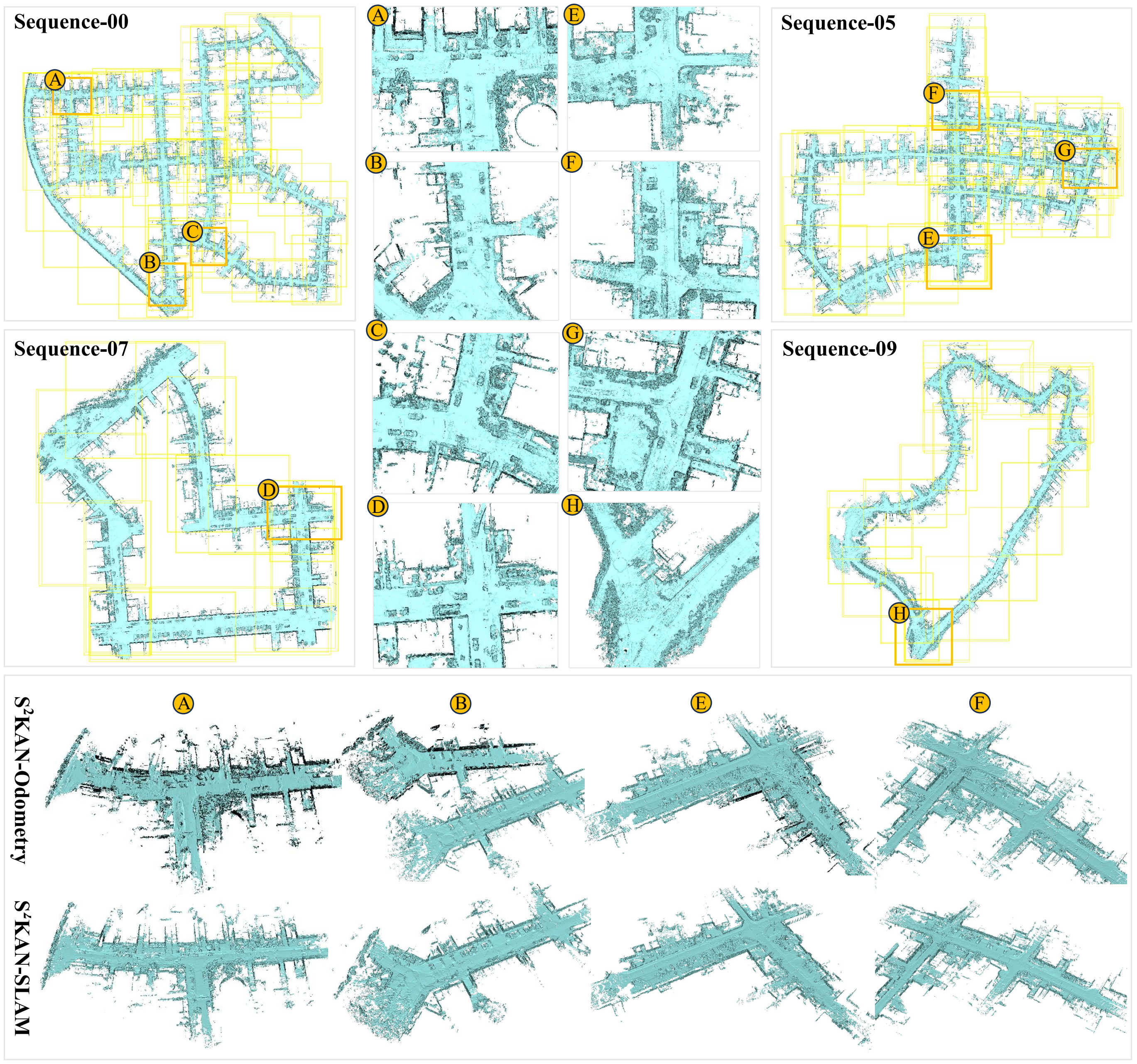

Fig. 3: Global consistent dense mapping results on KITTI Odometry. The yellow boxes indicate the bounding areas of submaps. The orange boxes represent the locations where loop closure exists, and their corresponding magnified images are drawn in the middle two columns of the subfigures. The two bottom rows exhibit the perspective views of four typical loop locations, where the maps built by S^{2}KAN-SLAM have much better global consistency over the ones produced by S^{2}KAN-Odometry.

are presented in the middle two columns of the subfigures. The perspective views of built maps at several loop closure locations by S^{2}KAN-Odometry and S^{2}KAN-SLAM are also presented in the two bottom rows, respectively.

As can be seen, the maps constructed at each loop closure location exhibit consistent alignment, underscoring the efficacy of S^{2}KAN-SLAM in mitigating cumulative errors and accurately aligning disparate submaps at identical locations. This successful alignment is attributed to the integration of loop closure detection at the backend and pose graph optimization incorporating node-node, loop closure, and node-submap constraints.

2) Global Localization: In the context of achieving global consistency in localization, we aligned the estimated trajectories generated by NeRF-LOAM, our odometry sub-system, and S^{2}KAN-SLAM with the respective ground truth trajectories, and visualized them in Fig. 4. The results indicate that S^{2}KAN-SLAM successfully attains globally consistent localization across various sequences, regardless of whether they feature a singular loop closure (Sequence-07 and Sequence-09) or multiple loop closures (Sequence-00 and Sequence-05). This highlights the system's capability to adapt to expansive environments. It is noteworthy that our odometry sub-system exhibits a higher cumulative error in long-term tracking compared to NeRF-LOAM, a consequence of the multi-submap

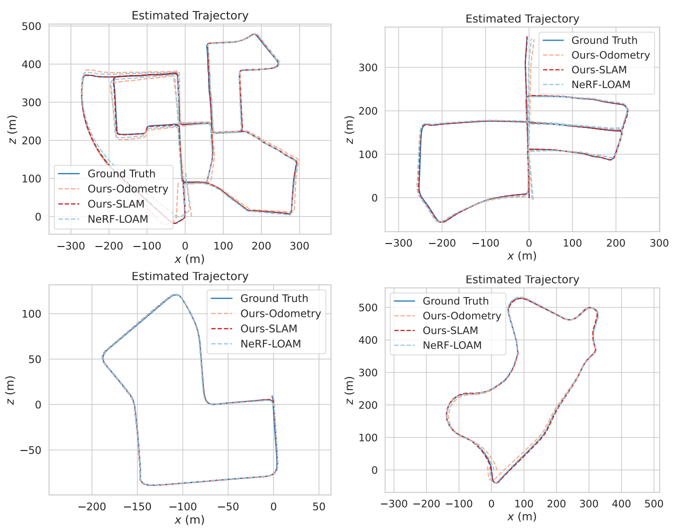

Fig. 4: Global consistent localization on KITTI Odometry. "Ours-SLAM" is the full S^{2}KAN-SLAM system with both the front-end and the backend, while "Ours-Odometry" refers to only the front-end.

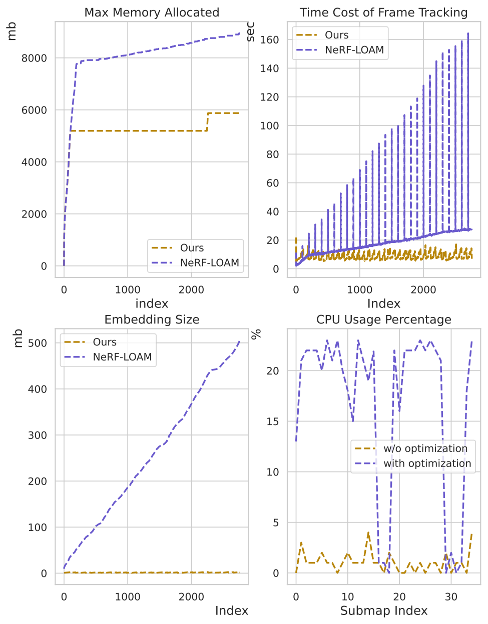

Fig. 5: The change curves of peak GPU memory usage, time cost of frame tracking, embedding size, and CPU usage percentage as the system runs.

representation as opposed to the global unique map representation. Despite this inherent drawback, we contend that the former choice offers greater flexibility and adaptability for long-term global mapping through the combination of high-accuracy short-term tracking and globally consistent backend optimization, in contrast to the latter one.

D. Run Time

Our submap representation allows the system's computational size and efficiency to be stabilized within a certain range. To corroborate this, we analyze the performance of the system in terms of memory usage, time cost, embedding size, and computation load.

1) Memory Usage: As shown in the top-left of Fig. 5, the peak GPU memory usage of NeRF-LOAM undergoes a rapid rise phase within the first 200 frames, and then still climbs slowly as the mapping range grows. In contrast, our scheme stabilizes the peak memory within a specific value of 6G after the cumulative number of processed point clouds reaches the maximum allowed by the submap (in all experiments, we set the submap size to 100 frames of point clouds). This feature allows our scheme to cope with the long-term stable mapping of outdoor large-scale scenes.

2) Time Cost: In terms of computational efficiency, we conducted a comparison of the time required to track a single frame of point cloud data. The analysis, depicted in the top-right of Fig. 5, reveals that NeRF-LOAM's tracking time demonstrates a growth trend similar to its peak GPU memory usage. An interesting observation is the pronounced spikes in tracking time at specific intervals. In contrast, our approach showcases a consistent periodic pattern in tracking time across different submap ranges. Notably, as the submap size expands, our computational time gradually increases, before decreasing when tracking transitions to a new submap. This results in a smooth overall trend without significant spikes in time cost.

3) Embedding Size: As previously highlighted, our utilization of single-valued embeddings for scene representation results in a significant reduction in the number of parameters that need to be learned. Furthermore, our submap representation serves to confine the parameter space of the scene within a specific range. The comparison depicted in the left-bottom subfigure of Fig. 5 illustrates that NeRF-LOAM experiences a substantial increase in parameter size as the mapping range expands, whereas our approach consistently maintains parameter size at a lower level over an extended duration (within 3Mb). This representation not only effectively minimizes GPU memory usage but also, and perhaps more crucially, a reduced number of parameters to be learned enhances learning efficiency.

4) Computation Load.: In terms of the front-end's computational load, as Table III shows, the flops of the MLP-based counterpart are approximately 17 times higher than that of our KAN-based decoder. Furthermore, with an increase in batch size, the computational complexity advantage of our KAN-based decoder becomes even more pronounced. It can also be seen that the parameters of the MLP-based decoder are approximately 17.5 times that of ours. In terms of the backend's computational load, as the right-bottom subfigure of Fig. 5 exhibits, when no loop closure exists, the computational load on the backend does not significantly increase with the growth in the number of submaps (expansion of mapping area). Upon detecting loop closures and establishing loop constraints, intermittent increases in CPU utilization are observed during the optimization of the pose graph by the backend. However, the peak CPU utilization does not exceed 25%, thus mitigating any substantial computational burden.

E. Ablation Study

We perform ablation studies on the decoder settings on the MaiCity dataset. The relevant results are reported in Table IV.

TABLE III: The flops and params of the KAN decoder of our S^{2}KAN-SLAM and those of the MLP decoder used in the counterpart NeRF-LOAM.

Decoder	Batch Size=1		Batch Size=2048	
	Flops (1e-3Mb)	Params (1e-3Mb)	Flops (Mb)	Params (1e-3Mb)
KAN	4.224	4.353	8.650	4.353
MLP	69.888	70.401	143.130	70.401

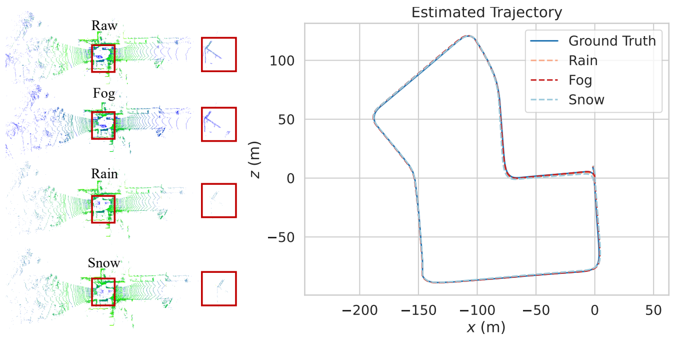

Fig. 6: Results of S^{2}KAN-SLAM under adverse conditions. Left: examples of the raw point cloud and the ones perturbed by fog, rain, and snow. Right: trajectories estimated by S^{2}KAN-SLAM.

1) Basis Function: We first analyze the effect of different basis functions on the mapping results. As shown in Table IV, compared with the B-Spline basis function used in the original KAN, the Gaussian Radial Basis used in this paper can obtain better results with the same number of parameters. This suggests that the Gaussian Radial Basis function is more suitable for modeling single-valued SDF fields.

2) Layers: The adequacy of the model's complexity must be tailored to the specific problem at hand to mitigate the risk of overfitting. As evidenced in Table IV, when using a single hidden layer, the KAN model obtains competent results with 64 functions. When increasing the number of layers, superior decoding results are achieved with a double hidden layer and 64 hidden functions.

3) Single-valued Embeddings: A comparison is conducted between single-valued SDF fields and high-dimensional embedding fields. The findings presented in the last and the third from the last rows of Table IV indicate a substantial enhancement in dense mapping with the utilization of single-valued fields. It is posited that in the context of SDF, the adoption of high-dimensional embedding may lead to excessive parameterization, consequently diminishing the efficacy of the learning process. Furthermore, as expounded upon in preceding sections, single-valued scene representations not only enhance learning efficiency but also contribute to a reduction in memory usage.

F. Results under Adverse Conditions

To demonstrate the robustness of the system, we utilized the raw point cloud data from the KITTI Odometry's 07 sequence as a basis to simulate point clouds under adverse weather conditions (such as fog, rain, and snow). These simulated point clouds were then used to test our S^{2}KAN-SLAM. The images in Fig. 6 illustrate the differences between the noisy point clouds and the original one, along with the corresponding trajectories of S^{2}KAN-SLAM. Despite the presence of pertubation, resulting in more noise points near and far from the LiDAR center, S^{2}KAN-SLAM achieved consistent localization under these conditions compared to the ground truth, confirming its robustness to noises.

TABLE IV: Ablation study on the decoder settings.

Basis Function	Layers	Accuracy	Completeness	C_{l1}	F-score
B-Spline	[1,32,1]	7.63	86.42	47.03	36.39
	[1,64,1]	7.03	23.31	15.17	60.10
	[1,128,1]	7.64	96.58	52.11	32.98
	[1,32,32,1]	7.81	30.17	18.99	49.70
	[1,64,64,1]	4.83	**7.40**	6.11	83.20
Gaussian Radial Basis	[1,32,1]	7.57	14.43	11.00	55.34
	[1,64,1]	5.22	8.42	6.82	80.99
	[1,128,1]	8.70	95.41	52.06	24.01
	[1,32,32,1]	5.59	12.42	9.01	76.48
	[1,64,64,1]	**4.28**	7.47	**5.88**	**86.75**
	[1,64,64,64,1]	4.66	7.50	6.08	86.47
	[16,64,64,1]	7.54	78.60	43.07	43.80

V. Conclusion

In this paper, to realize elastic globally consistent localization and dense mapping of an outdoor large-scale scene, we propose a SLAM scheme based on the scene representation with a single-valued SDF along with a KAN decoder and the scene organization with multiple SDF submaps. In the front-end, the system tracks the LiDAR pose and constructs local neural maps by minimizing the difference between the ranging value and the sum of sampling distance and sampled SDF value in a frame-to-model manner; in the backend, the system constructs a global pose graph containing node-to-node, node-to-submap, and loop closure constraints, eliminating the cumulative error through joint optimization to achieve globally consistent localization and dense map construction. Extensive experiments on virtual and real large-scale scene datasets have validated the effectiveness of our approach.

While S^{2}KAN-SLAM achieves global consistency in mapping through submap representation and single-value field, ray tracing based on the SDF field remains an efficiency bottleneck during tracking and mapping. On the one hand, designing new ray casting algorithms and utilizing CUDA acceleration can alleviate this issue to some extent. On the other hand, adopting the space-time trade-off concept, such as representing scenes as regular grids and utilizing tensor decomposition for compression, can also enable faster SDF queries without significantly increasing storage consumption. In future research, we will focus on these aspects to further enhance the efficiency of front-end tracking and mapping.

References

[1] J. Zhang and S. Singh, "LOAM: Lidar odometry and mapping in real-time," in *Proc. Robot. Sci. Syst. Conf.*, 2014, pp. 1–9.
[2] T. Shan and B. Englot, "LeGO-LOAM: Lightweight and ground-optimized lidar odometry and mapping on variable terrain," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2018.
[3] H. Wang, C. Wang, C.-L. Chen, and L. Xie, "F-LOAM: Fast lidar odometry and mapping," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2021, pp. 4390–4396.
[4] W. Hess, D. Kohler, H. Rapp, and D. Andor, "Real-time loop closure in 2D LiDAR SLAM," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2016, pp. 1271–1278.

[5] Z. Wang, L. Zhang, S. Zhao, and Y. Zhou, “Ct-LVI: A framework toward continuous-time laser-visual-inertial odometry and mapping,” *IEEE Trans. Circuits Syst. Video Tech.*, vol. 34, no. 6, pp. 4378–4391, 2024.

[6] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng, “NeRF: representing scenes as neural radiance fields for view synthesis,” *Commun. ACM*, vol. 65, no. 1, pp. 99–106, 2021.

[7] S. Guo, Q. Wang, Y. Gao, R. Xie, L. Li, F. Zhu, and L. Song, “Depth-guided robust point cloud fusion nerf for sparse input views,” *IEEE Trans. Circuits Syst. Video Tech.*, Early Access, 2024.

[8] L. Lin, J. Zhu, and Y. Zhang, “Multiview textured mesh recovery by differentiable rendering,” *IEEE Trans. Circuits Syst. Video Tech.*, vol. 33, no. 4, pp. 1684–1696, 2023.

[9] S. Fang, “Exploring the capabilities of NeRF in generating 3D models,” *EAI Endorsed Trans. AI Robot.*, vol. 3, pp. 1–12, 2024.

[10] S. Fang, X. Feng, and Y. Lv, “Methods and strategies for 3D content creation based on 3D native methods,” *EAI Endorsed Trans. AI Robot.*, vol. 3, pp. 1–12, 2024.

[11] C.-H. Lin, W.-C. Ma, A. Torralba, and S. Lucey, “BARF: Bundle-adjusting neural radiance fields,” in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2021, pp. 5721–5731.

[12] Z. Zhu, S. Peng, V. Larsson, W. Xu, H. Bao, Z. Cui, M. R. Oswald, and M. Pollefeys, “NICE-SLAM: Neural implicit scalable encoding for SLAM,” in *Proc. IEEE Conf. Comput. Vis. Patt. Recog.*, 2022, pp. 12 776–12 786.

[13] W. Bian, Z. Wang, K. Li, J. Bian, and V. A. Prisacariu, “NoPe-NeRF: Optimising neural radiance field with no pose prior,” in *Proc. IEEE Conf. Comput. Vis. Patt. Recog.*, 2023, pp. 4160–4169.

[14] T. Zhang, L. Zhang, F. Zhang, S. Zhao, and Y. Zhou, “I-DACS: Always maintaining consistency between poses and the field for radiance field construction without pose prior,” *IEEE Trans. Circuits Syst. Video Tech.*, Early Access, 2024.

[15] X. Yang, H. Li, H. Zhai, Y. Ming, Y. Liu, and G. Zhang, “Vox-Fusion: dense tracking and mapping with voxel-based neural implicit representation,” in *IEEE Int. Symp. Mixed Augmented Reality*, 2022.

[16] H. Wang, J. Wang, and L. Agapito, “Co-SLAM: Joint coordinate and sparse parametric encodings for neural real-time SLAM,” in *Proc. IEEE Conf. Comput. Vis. Patt. Recog.*, 2023.

[17] S. Isaacson, P.-C. Kung, M. Ramanagopal, R. Vasudevan, and K. A. Skinner, “LONER: Lidar only neural representations for real-time SLAM,” *IEEE Robotics and Automation Letters*, vol. 8, no. 12, pp. 8042–8049, 2023.

[18] J. Deng, Q. Wu, X. Chen, S. Xia, Z. Sun, G. Liu, W. Yu, and L. Pei, “NeRF-LOAM: Neural implicit representation for large-scale incremental lidar odometry and mapping,” in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2023, pp. 8184–8193.

[19] A. Chen, Z. Xu, A. Geiger, J. Yu, and H. Su, “TensoRF: Tensorial radiance fields,” in *Proc. Euro. Conf. Comput. Vis.*, 2022, pp. 333–350.

[20] V. Sitzmann, J. N. Martel, A. W. Bergman, D. B. Lindell, and G. Wetzstein, “Implicit neural representations with periodic activation functions,” in *Neural Info. Process. Syst.*, 2020, pp. 1–12.

[21] X. Zhong, Y. Pan, J. Behley, and C. Stachniss, “SHINE-Mapping: Large-scale 3D mapping using sparse hierarchical implicit neural representations,” in *Proc. IEEE Int. Conf. Robot. Autom.*, 2023, pp. 8371–8377.

[22] Z. Liu, Y. Wang, S. Vaidya, F. Ruehle, J. Halverson, M. Soljačić, T. Y. Hou, and T. Max, “KAN: Kolmogorov-Arnold networks,” 2024, arXiv preprint arXiv:2404.19756v4.

[23] P. Besl and N. D. McKay, “A method for registration of 3-D shapes,” *IEEE Trans. Patt. Anal. Mach. Intell.*, vol. 14, no. 2, pp. 239–256, 1992.

[24] P. Dellenbach, J.-E. Deschaud, B. Jacquet, and F. Goulette, “CT-ICP: real-time elastic lidar odometry with loop closure,” in *Proc. IEEE Int. Conf. Robot. Autom.*, 2022, pp. 5580–5586.

[25] 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 Robot. Autom. Letters*, vol. 8, no. 2, pp. 1029–1036, 2023.

[26] X. Hu, Y. Wu, M. Zhao, L. Yang, X. Zhang, and X. Ji, “PAS-SLAM: A visual SLAM system for planar-ambiguous scenes,” *IEEE Trans. Circuits Syst. Video Tech.*, Early Access, 2024.

[27] J. Lin and F. Zhang, “Loam livox: A fast, robust, high-precision lidar odometry and mapping package for lidars of small FoV,” in *Proc. IEEE Int. Conf. Robot. Autom.*, 2020, pp. 3126–3131.

[28] J. Behley and C. Stachniss, “Efficient surfel-based slam using 3D laser range data in urban environments,” in *Proc. Robot. Sci. Syst. Conf.*, 2018, pp. 1–10.

[29] I. Vizzo, X. Chen, N. Chebrolu, J. Behley, and C. Stachniss, “Poisson surface reconstruction for lidar odometry and mapping,” in *Proc. IEEE Int. Conf. Robot. Autom.*, 2021, pp. 5624–5630.

[30] S. Kohlbrecher, O. Von Stryk, J. Meyer, and U. Klingauf, “A flexible and scalable SLAM system with full 3D motion estimation,” in *Proc. IEEE Int. Symp. Safet. Secur. Rescu. Robot.*, 2011, pp. 1–6.

[31] Z. Zhang, J. Sun, Y. Dai, B. Fan, and M. He, “VRNet: Learning the rectified virtual corresponding points for 3D point cloud registration,” *IEEE Trans. Circuits Syst. Video Tech.*, vol. 32, no. 8, pp. 4997–5010, 2022.

[32] Y. Aoki, H. Goforth, R. A. Srivatsan, and S. Lucey, “PointNetLK: Robust & efficient point cloud registration using PointNet,” in *Proc. IEEE Conf. Comput. Vis. Patt. Recog.*, 2019, pp. 7156–7165.

[33] M. A. Uy and G. H. Lee, “PointnetVLAD: Deep point cloud based retrieval for large-scale place recognition,” in *Proc. IEEE Conf. Comput. Vis. Patt. Recog.*, 2018, pp. 4470–4479.

[34] Y. Wang, Y. Qiu, P. Cheng, and J. Zhang, “Hybrid CNN-Transformer features for visual place recognition,” *IEEE Trans. Circuits Syst. Video Tech.*, vol. 33, no. 3, pp. 1109–1122, 2023.

[35] W. Tang, “Review of image classification algorithms based on graph convolutional networks,” *EAI Endorsed Trans. AI Robot.*, vol. 2, no. 1, pp. 1–10, 2023.

[36] Q. Li, S. Chen, C. Wang, X. Li, C. Wen, M. Cheng, and J. Li, “LO-Net: Deep real-time lidar odometry,” in *Proc. IEEE Conf. Comput. Vis. Patt. Recog.*, 2019, pp. 8465–8474.

[37] G. Wang, X. Wu, Z. Liu, and H. Wang, “PWCLO-Net: Deep lidar odometry in 3D point clouds using hierarchical embedding mask optimization,” in *Proc. IEEE Conf. Comput. Vis. Patt. Recog.*, 2021, pp. 15 905–15 914.

[38] H. Matsuki, R. Murai, P. H. Kelly, and A. J. Davison, “Gaussian splatting SLAM,” in *Proc. IEEE Conf. Comput. Vis. Patt. Recog.*, 2024, pp. 18 039–18 048.

[39] 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. IEEE Conf. Comput. Vis. Patt. Recog.*, 2024, pp. 21 357–21 366.

[40] B. Kerbl, G. Kopanas, T. Leimkuehler, and G. Drettakis, “3D Gaussian splatting for real-time radiance field rendering,” *ACM Trans. Graph.*, vol. 42, no. 4, pp. 1–14, 2023.

[41] X. Yu, Y. Liu, S. Mao, S. Zhou, R. Xiong, Y. Liao, and Y. Wang, “NF-Atlas: Multi-volume neural feature fields for large scale lidar mapping,” *IEEE Robot. Autom. Letters*, vol. 8, no. 9, pp. 5870–5877, 2023.

[42] E. Vespa, N. Nikolov, M. Grimm, L. Nardi, P. H. J. Kelly, and S. Leutenegger, “Efficient octree-based volumetric SLAM supporting signed-distance and occupancy mapping,” *IEEE Robot. Autom. Letters*, vol. 3, no. 2, pp. 1144–1151, 2018.

[43] G. Kim and A. Kim, “Scan Context: Egocentric spatial descriptor for place recognition within 3D point cloud map,” in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2018, pp. 4802–4809.

[44] Z. Li, S. Li, O. O. Bamasag, A. Alhothali, and X. Luo, “Diversified regularization enhanced training for effective manipulator calibration,” *IEEE Trans. Neural Networks Learn. Syst.*, vol. 34, no. 11, pp. 8778–8790, 2023.

[45] Z. Li, S. Li, A. Francis, and X. Luo, “A novel calibration system for robot arm via an open dataset and a learning perspective,” *IEEE Trans. Circuits Syst. II: Express Briefs*, vol. 69, no. 12, pp. 5169–5173, 2022.

[46] Z. Li, S. Li, and X. Luo, “An overview of calibration technology of industrial robots,” *IEEE/CAA J. Automatica Sinica*, vol. 8, no. 1, pp. 23–36, 2021.

[47] A. H. Khan, X. Cao, B. Xu, and S. Li, “A model-free approach for online optimization of nonlinear systems,” *IEEE Trans. Circuits Syst. II: Express Briefs*, vol. 69, no. 1, pp. 109–113, 2022.

[48] A. H. Khan, X. Cao, S. Li, V. N. Katsikis, and L. Liao, “BAS-ADAM: an ADAM based approach to improve the performance of beetle antennae search optimizer,” *IEEE/CAA J. Auto. Sinica*, vol. 7, no. 2, pp. 461–471, 2020.

[49] K. Levenberg, “A method for the solution of certain problems in least square,” *Quarterly Applied Mathematics*, vol. 2, no. 2, pp. 164–168, 1944.

[50] W. E. Lorensen and H. E. Cline, “Marching cubes: A high resolution 3d surface construction algorithm,” New York, NY, USA, 1987, p. 163–169.

[51] A. Geiger, P. Lenz, and R. Urtasun, “Are we ready for autonomous driving? the KITTI vision benchmark suite,” in *Proc. IEEE Conf. Comput. Vis. Patt. Recog.*, 2012, pp. 3354–3361.

[52] M. Ramezani, Y. Wang, M. Camurri, D. Wisth, M. Mattamala, and M. Fallon, “The Newer College dataset: Handheld lidar, inertial and

vision with ground truth," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2020, pp. 4353–4360.

[53] S. Umeyama, "Least-squares estimation of transformation parameters between two point patterns," *IEEE Trans. Patt. Anal. Mach. Intell.*, vol. 13, no. 4, pp. 376–380, 1991.

[54] A. Knapitsch, J. Park, Q.-Y. Zhou, and V. Koltun, "Tanks and temples: benchmarking large-scale scene reconstruction," *ACM Trans. Graph.*, vol. 36, no. 4, pp. 1–13, 2017.

[55] I. Vizzo, T. Guadagnino, J. Behley, and C. Stachniss, "VDBFusion: flexible and efficient TSDF integration of range sensor data," *Sensors*, vol. 22, no. 3, pp. 1296–1310, 2022.

[56] X. Zhong, Y. Pan, C. Stachniss, and B. Jens, "3D lidar mapping in dynamic environments using a 4D implicit neural representation," in *Proc. IEEE Conf. Comput. Vis. Patt. Recog.*, 2024, pp. 15 417–15 427.

[57] J. Huang, Z. Gojcic, M. Atzmon, O. Litany, S. Fidler, and F. Williams, "Neural kernel surface reconstruction," in *Proc. IEEE Conf. Comput. Vis. Patt. Recog.*, 2023, pp. 4369–4379.

[58] J. Ruan, B. Li, Y. Wang, and Y. Sun, "SLAMesh: Real-time LiDAR simultaneous localization and meshing," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2023, pp. 3546–3552.

Zhong Wang received the B.S. and M.S. degrees from the School of Surveying and Geo-Informatics, Tongji University, Shanghai, China, in 2016 and 2019, respectively. He received the Ph.D. degree from the School of Software Engineering, Tongji University, Shanghai, China in 2023. Starting from Dec. 2023, he worked as a postdoctoral at the Department of Automation, Shanghai Jiao Tong University, China. His research interests include SLAM, 3D reconstruction, and robotic vision.

Lin Zhang (Senior Member, IEEE) received the B.Sc. and M.Sc. degrees from the Department of Computer Science and Engineering, Shanghai Jiao Tong University, Shanghai, China, in 2003 and 2006, respectively. He received the Ph.D. degree from the Department of Computing, The Hong Kong Polytechnic University, Hong Kong, in 2011. From March 2011 to August 2011, he was a Research Associate with the Department of Computing, The Hong Kong Polytechnic University. In Aug. 2011, he joined the School of Software Engineering, Tongji University, Shanghai, China, where he is currently a Full Professor. His current research interests include environment perception of intelligent vehicle, pattern recognition, computer vision, and perceptual image/video quality assessment. He serves as an Associate Editor for IEEE Robotics and Automation Letters, and Journal of Visual Communication and Image Representation. He was awarded as a Young Scholar of Changjiang Scholars Program, Ministry of Education, China.

Hesheng Wang (Senior Member, IEEE) received the B.Eng. degree in electrical engineering from the Harbin Institute of Technology, Harbin, China, in 2002, and the M.Phil. and Ph.D. degrees in automation and computer-aided engineering from The Chinese University of Hong Kong, Hong Kong, in 2004 and 2007, respectively. He is currently a Distinguished Professor with the Department of Automation, Shanghai Jiao Tong University, Shanghai, China. His current research interests include visual servoing, intelligent robotics, computer vision, and autonomous driving. Dr. Wang is an Associate Editor of Robotic Intelligence and Automation and the International Journal of Humanoid Robotics, a Senior Editor of the IEEE/ASME Transactions on Mechatronics, an Editor-in-chief of Robot Learning. He served as an Associate Editor of the IEEE Transactions on Robotics from 2015 to 2019, an IEEE Transactions on Automation Science and Engineering from 2021 to 2023, a Technical Editor of the IEEE/ASME Transactions on Mechatronics from 2020 to 2023, an Editor of Conference Editorial Board (CEB) of IEEE Robotics and Automation Society from 2022 to 2024. He was the General Chair of IEEE ROBIO 2022 and IEEE RCAR 2016, and the Program Chair of the IEEE ROBIO 2014 and IEEE/ASME AIM 2019. He will be the General Chair of IEEE/RSJ IROS 2025.