LONER: LiDAR Only Neural Representations for Real-Time SLAM

Seth Isaacson*,1, Pou-Chun Kung*,1, Mani Ramanagopal[1], Ram Vasudevan[1], and Katherine A. Skinner[1]

提出了一种新的信息论损失函数

***Abstract*—This paper proposes *LONER*, the first real-time LiDAR SLAM algorithm that uses a neural implicit scene representation. Existing implicit mapping methods for LiDAR show promising results in large-scale reconstruction, but either require groundtruth poses or run slower than real-time. In contrast, LONER uses LiDAR data to train an MLP to estimate a dense map in real-time, while simultaneously estimating the trajectory of the sensor. To achieve real-time performance, this paper proposes a novel information-theoretic loss function that accounts for the fact that different regions of the map may be learned to varying degrees throughout online training. The proposed method is evaluated qualitatively and quantitatively on two open-source datasets. This evaluation illustrates that the proposed loss function converges faster and leads to more accurate geometry reconstruction than other loss functions used in depth-supervised neural implicit frameworks. Finally, this paper shows that LONER estimates trajectories competitively with state-of-the-art LiDAR SLAM methods, while also producing dense maps competitive with existing real-time implicit mapping methods that use groundtruth poses.**

***Index Terms*—SLAM, Mapping, Deep Learning Methods, Implicit Representations, NeRF**

可以查询场景中的任何点的占用情况

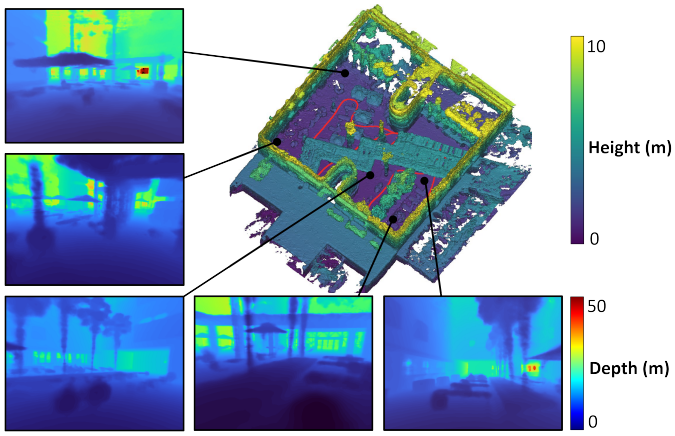

Fig. 1. LONER reconstruction on a courtyard scene [4]. The top-right is a mesh reconstruction with the estimated trajectory in red. The surrounding images are rendered depth images from novel views outside of the training trajectory, demonstrating LONER's ability to reconstruct dense novel views of an environment.

I. Introduction

NEURAL implicit scene representations, such as Neural Radiance Fields (NeRFs), offer a promising new way to represent maps for robotics applications [1]. Traditional NeRFs employ a Multi Layer Perceptron (MLP) to estimate the radiance and volume density of each point in space, enabling dense scene reconstruction and novel view synthesis. The learned scene representation has several advantages over conventional map representations, such as point clouds and occupancy grids. First, because the domain of the NeRF is continuous and does not enforce discretization, any point in the scene can be queried for occupancy. The continuity of the scene can be exploited to solve a variety of robotics problems. For example, as demonstrated in [2], a motion planner can integrate the volume density along a proposed trajectory to evaluate the likelihood of a collision. Other benefits include the ability to produce realistic renders of the scene [1]. Further, NeRFs can be used to estimate uncertainty of renders to enable view selection for active exploration [3]. This paper advances neural implicit scene representations for robotics applications. Specifically, we introduce the first real-time LiDAR-only SLAM algorithm that achieves accurate pose estimation and map reconstruction and learns a neural implicit representation of a scene.

Several recent papers have proposed real-time NeRF-based visual SLAM systems using monocular or RGB-D cameras [5, 6, 7]. These systems demonstrate impressive performance on indoor scenes. For outdoor environments, prior work has focused on using neural implicit representations for LiDAR to enable dense 3D reconstruction and novel view synthesis for large-scale scenes [8, 9, 10]. Recent methods have even shown promising results for LiDAR localization and mapping with neural implicit frameworks in large-scale outdoor scenes [11, 12]. Still, these LiDAR-supervised algorithms do not operate in real-time, which is necessary for robotics applications. The contributions of this paper are as follows:

1) We propose the first real-time neural implicit LiDAR SLAM method, which adapts to outdoor environments and provides accurate online state estimation.
2) We introduce a novel loss function that leads to faster convergence and more accurate reconstruction than existing loss functions.

We demonstrate that our proposed method, LONER, runs in real-time and estimates both trajectories and maps more accurately than baselines. Figure 1 shows the reconstruction results on the Fusion Portable dataset [4]. A project page is available at https://umautobots.github.io/loner.

The remainder of this paper is organized as follows: In Section II, we review related work. In Section III, we describe

[1]S. Isaacson, P. Kung, M. Ramanagopal, R. Vasudevan, and K. A. Skinner are with the Department of Robotics, University of Michigan, Ann Arbor, MI 48109. {sethgi, pckung, srmani, ramv, kskin}@umich.edu.

This work is supported by the Ford Motor Company via the Ford-UM Alliance under award N028603.

*These two authors contributed equally to this work.

LONER. In Section IV, we evaluate LONER, and in Section V, we conclude and discuss both limitations and future work.

II. Related Works

A. LiDAR SLAM

LiDAR SLAM has been an active research area over the past several decades [13, 14, 15, 16, 17]. The primary goal of these methods is to estimate the trajectory of the ego vehicle. Modern methods such as LeGO-LOAM estimate motion by aligning features extracted from consecutive scans, then accumulate LiDAR scans to build a map [13, 15]. These works primarily focus on accurate trajectory estimation, and thus creating dense, realistic maps is not a focus of the approach. In contrast, our method aims to achieve similar or better trajectory estimation while also estimating dense maps.

B. Real-time NeRF-based SLAM

NeRFs use images of a scene captured from known camera poses to train an MLP to predict what the scene will look like from novel poses [1]. While originally developed for offline use with known camera poses, NeRFs have recently been used to learn an implicit scene representation in RGB and RGB-D SLAM frameworks [5, 6, 7]. By representing the scene with a NeRF, these algorithms perform both tracking and mapping via gradient descent on an MLP or related feature vectors. For example, iMAP represents the scene as a single MLP [5]. Each RGB-D frame is first tracked by fixing the MLP weights and optimizing the camera poses. Then, the new information is incorporated into the map by jointly optimizing the MLP and pose estimates. iMAP shows promising results in small scenarios but does not scale to larger or outdoor scenes. NICE-SLAM replaces iMAP's simple MLP with a hierarchical feature grid combined with MLP decoders [6]. This approach demonstrates better scalability than the single MLP used in iMAP, but NICE-SLAM still only works in indoor scenarios. Additionally, NeRF-SLAM uses DROID-SLAM [18] as the tracking front-end, which allows them to use a probabilistic volumetric NeRF to perform uncertainty-aware mapping and pose refinement [7]. Recently, several more papers have introduced architectures and encodings to improve neural-implicit SLAM's memory efficiency, computation speed, and accuracy [19, 20, 21, 22]. Our method extends these recent advances to leverage implicit scene representation for real-time LiDAR-only SLAM, which allows operation in large, outdoor environments.

C. Neural Implicit Representations for LiDAR

While neural implicit representations were initially developed for visual applications, several works have introduced neural implicit representations for LiDAR to improve outdoor 3D reconstruction performance [8, 23, 9]. Urban Radiance Fields (URF) is an early example of LiDAR-integraded NeRF [8]. URF uses a novel Line-of-Sight (LOS) loss to improve LiDAR supervision. CLONeR uses LiDAR and camera data to train two decoupled MLPs, one of which learns scene structure and the other of which learns scene color [9]. CLONeR combines the decoupled NeRF with occupancy-grid enabled sampling heuristics and URF's Line-of-Sight loss to enable training with as few as two input views [9]. Both URF and CLONeR require known sensor poses and assume offline training. In contrast, our proposed method performs real-time LiDAR SLAM that both reconstructs 3D environments and estimates sensor poses for sequential input data.

In [12], a method is introduced that inputs LiDAR scans and approximate poses, then uses a novel occlusion-aware loss function to jointly optimize the poses and a NeRF. This work assumes a-priori availability of all data. Thus, it can be effectively viewed as a LiDAR-based structure-from-motion algorithm, whereas we present a full SLAM algorithm. Recently, SHINE Mapping presented a LiDAR mapping method based on neural signed distance function (SDF) and sparse feature embedding [10]. While this embedding helps scale to large scenes, in a real-time configuration, it presents a trade-off between hole-filling and overall map resolution. Our method instead uses a dense feature embedding, which enables improved performance across both hole-filling capability and map resolution. NeRF-LOAM extends this to a LiDAR SLAM system and proposes a dynamic voxel embedding generation strategy to adapt to large-scale scenarios [11]. However, it does not operate in real-time.

D. Loss for Depth-supervised NeRF

Depth-supervised NeRF frameworks, such as those that use RGB-D sensors, typically use the difference between rendered and sensed depth as a loss to learn geometry from 2D images by volumetric rendering [5, 6]. Other works use depth measurements directly in 3D space to perform depth-supervision [23, 8, 9, 12]. The Binary Cross-Entropy (BCE) loss proposed in [12] reasons about occluded objects, but does not consider measurement uncertainty. The KL divergence loss presented by DS-NeRF [23] and Line-Of-Sight (LOS) loss introduced by URF [8] approximate each LiDAR ray's termination depth as a normal distribution centered at the measured depth. The variance of the distribution is correlated with a margin parameter ϵ. The loss functions encourage the network to predict weights along a ray equal to the PDF of the normal distribution. While the KL loss leaves the variance fixed during training, [8] shows that decaying ϵ during training improves reconstruction accuracy when using the LOS loss.

While uniformly decaying a margin is successful offline, using a single margin for all rays is unsuitable for real-time SLAM, which has incremental input and limited training samples. Using a uniform margin can force the NeRF model to forget learned geometry when adding a new LiDAR scan and can cause slower convergence. Therefore, this paper proposes a novel dynamic margin loss that applies a different margin for each ray. We demonstrate the proposed loss function leads to better 3D reconstruction than previous loss functions within fewer training samples, and enables real-time performance.

III. Method

This section provides a high-level overview of our proposed system, LONER, before explaining each component in detail.

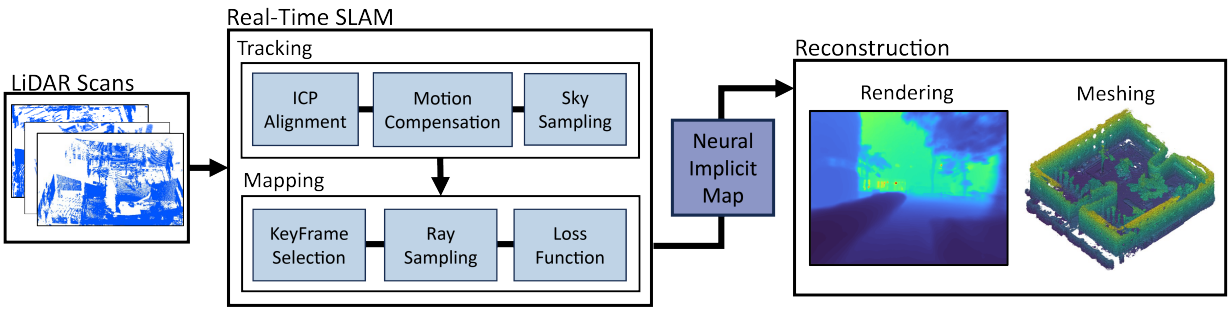

Fig. 2. LONER system overview. Incoming scans are decimated then tracked with ICP, after which the sky is segmented from the scene geometry. Selected scans are chosen as KeyFrames. Each map update includes the current KeyFrame and randomly selected past KeyFrames. Our novel loss function is used to update the poses and MLP weights. The resulting implicit map can be rendered offline to a variety of formats including depth images and meshes.

A. System Overview

An overview of LONER is shown in Fig. 2. As is common in the SLAM literature [5, 6, 24], the system comprises parallel threads for tracking and mapping. The tracking thread processes incoming scans and estimates odometry using ICP. LONER is designed for use without an IMU, so ICP uses the identity transformation as an initial guess. In parallel and at a lower rate, the mapping thread uses the current scan and selected prior scans as KeyFrames, which are used to update the training of the neural scene representation.

B. Tracking

Incoming LiDAR scans are decimated to a fixed frequency of 5Hz. The relative transform $P_{i-1,i} \in SE(3)$ from the previous scan to the current scan is estimated using Point-to-Plane ICP [25]. We adopted ICP rather than inverse-NeRF due to strong performance and to save the GPU resources for the mapping module to maintain real-time performance. The ICP estimate is later refined in our mapping optimization. The LiDAR pose $\mathbf{x}_i \in SE(3)$ is then estimated as $\hat{\mathbf{x}}_i = \hat{\mathbf{x}}_{i-1} \cdot P_{i-1,i}$. Given the previous and current pose, LiDAR scans are motion-compensated by assuming constant velocity motion between scans.

匀速运动假设

C. Implicit Map Representation

The scene is represented as an MLP with the hierarchical feature grid encoding from [26]. During online training, the parameters Θ of the MLP and the feature grid are updated to predict the volume density σ of each point in space. To train the network and estimate depths, we follow the standard volumetric rendering procedure [1]. In particular, for a LiDAR ray $\vec{r}$ with origin $\vec{o}$ and direction $\vec{d}$, we choose distances $t_i \in [t_{near}, t_{far}]$ to create N_S samples $s_i = \vec{o} + t_i\vec{d}$. LiDAR intrinsics dictate t_{near}, while t_{far} depends on the scale of the scene. The feature grid and MLP, collectively $\mathcal{F}(s_i; \Theta)$, are queried to predict the occupancy state σ_i. Then, weights transmittances T_i and weights w_i are computed according to:

$$T_i = \exp(-\sum_{j=1}^{i-1} \sigma_j \delta_j) \tag{1}$$

$$w_i = T_i \sigma_i \tag{2}$$

where $\delta_j = t_{j+1} - t_j$, and σ_i is the density at sample s_i predicted by the MLP. The weights w_i are used by the loss function and represent the probability that the ray terminates at each point. Therefore, the expected termination depth of a ray $\hat{D}(\vec{r})$ can be estimated as

$$\hat{D}(\vec{r}) = \sum_{i=1}^{N} w_i t_i. \tag{3}$$

D. Mapping

The mapping thread receives LiDAR scans from the tracking thread and determines whether to form a KeyFrame. If the scan is accepted, the map is jointly optimized with the poses.

1) KeyFrames: KeyFrames are selected temporally: if t_{KF} has passed since the previous KeyFrame, a new KeyFrame is added. Each time a KeyFrame is accepted, the optimizer is updated. N_W total KeyFrames are used in the update, including the current KeyFrame and $N_W - 1$ random selected past KeyFrames.

这里的关键帧是随机选取的

2) Optimization: Once the window of KeyFrames has been selected, the map is jointly optimized with the poses of KeyFrames in the optimization window. For a KeyFrame $\mathbf{KF}_i$ with estimated pose $\hat{\mathbf{x}}_i$ in the world frame, a twist vector $\hat{\xi}_i \in \mathbb{R}^6$ is formed to be used as the optimization variable. Specifically, $\hat{\xi}_i = (\hat{\omega}_i, \hat{v}_i)$ where $\hat{\omega}_i$ is the axis-angle representation of the rotation component of $\hat{\mathbf{x}}_i$, and $\hat{v}_i$ is the translation component. In the forward pass, this vector is converted back into a pose $\hat{\mathbf{x}}_i$ and used to compute the origin of rays. N_R rays are sampled at random from the LiDAR scan, and N_S depth samples are taken from each ray using the occupancy grid heuristic introduced by [9].

In the backward pass, gradients are computed for MLP and feature grid parameters Θ and twist vectors $\hat{\xi}_i$. At the end of the optimization, the optimized twist vectors ξ_i^* are converted into $SE(3)$ transformation matrices $\mathbf{x}_i^*$. The tracking thread is informed of this change, such that future tracking is performed relative to the optimized poses.

E. JS Dynamic Margin Loss Function

The primary loss function in our system is a novel dynamic margin loss. This is combined with terms for depth loss and sky loss as follows:

$$\mathcal{L}(\Theta) = \mathcal{L}_{JS} + \lambda_1 \mathcal{L}_{depth} + \lambda_2 \mathcal{L}_{sky}. \tag{4}$$

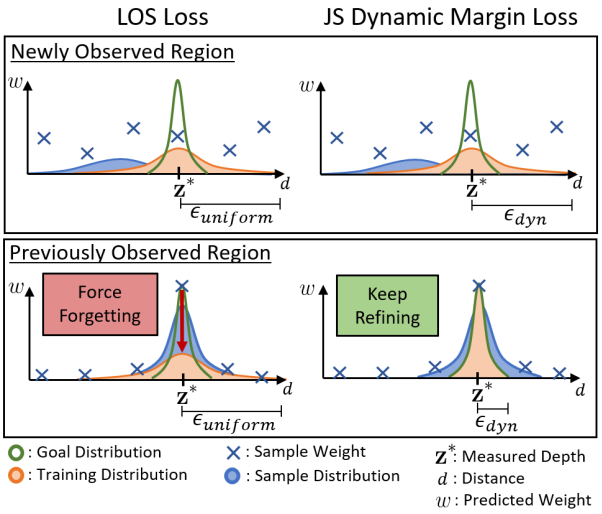

Fig. 3. Illustration of the difference between the JS loss and the LOS loss. The LOS loss sets a uniform margin ϵ for rays pointing to both learned and unobserved regions. This strategy corrupts the learned information by forcing learned regions to predict higher variances. In contrast, the proposed JS loss sets the dynamic margin ϵ for each ray depending on the similarity between goal distribution and predicted sample distribution. The JS loss sets higher margins for rays in unobserved regions to improve convergence, and sets lower margins for rays in learned regions to refine learned geometry.

Each of these terms is explained below.

1) JS Loss Formulation: The LOS loss used by [8, 9] uses a single margin for all rays; we use a similar formulation but introduce a novel strategy based on the Jensen-Shannon Divergence [27] to assign a unique margin to each ray. For a given LiDAR ray $\vec{r}$, the samples along the ray are $s_i = \vec{o}+t_i\vec{d}$, and z^* denotes the measured depth along the ray. t_i denotes the distance of individual training samples along the ray, and w_i represents a corresponding weight prediction from an MLP, as defined in Equation 2. We define a truncated Gaussian distribution $\mathcal{K}_\epsilon$ that has a bounded domain parameterized by margin ϵ, with $\mathcal{K}_\epsilon = \mathcal{N}(0, (\epsilon/3)^2)$ as the training distribution. Thus, target weights are given by $w_i^* = \mathcal{K}_\epsilon(t_i - z^*)$. The JS loss is defined as

$$\mathcal{L}_{JS}(\Theta) = \underbrace{\|w_i^* - w_i\|_1}_{\text{Primary Loss}} + \underbrace{\|1 - \sum_i w_i\|_1}_{\text{Opacity Loss}}, \tag{5}$$

where the opacity loss (explained in more detail by [9]) ensures weights along each ray sum to one and thus form a probability distribution. Note that while URF [8] uses an L2 loss to compute the LOS loss, we follow [9] and use an L1 loss. The effect of this is discussed in Section IV-D.

In [8, 9], the margin decays exponentially throughout training and, at each iteration, a single margin is shared by all of the rays. In contrast, we present a JS divergence-based dynamic margin that computes a unique margin for each ray to improve the training convergence and reconstruction accuracy.

In a SLAM application, continuous optimization, sparse sampling, and incremental input lead to different regions of the map being learned to varying degrees during online training. As shown in Fig. 3, using a uniform ϵ in the LOS loss causes forgetting in regions that have already been learned. The idea

使用较大的边距来指向地图上未知几何形状的区域，而使用较小的边距来指向熟悉的区域

of the JS dynamic margin is to use a larger margin for rays pointing toward regions of the map with unknown geometry while using a smaller margin for rays pointing toward well-learned regions. This allows the system to learn new regions while preserving and refining learned geometry. We use the JS divergence to measure the dissimilarity between the goal distribution and the sample distribution for each ray, which represents how well the map has learned along the ray. Learned regions have similar goal and sample distributions, which lead to smaller JS divergence. We define a goal distribution $G = \mathcal{N}(z^*, \sigma^*)$, where $\sigma^* = \epsilon_{min}/3$. Further, we define the sample distribution $S = \mathcal{N}(\bar{\mu}_w, \bar{\sigma}_w)$, where $\bar{\mu}_w$ and $\bar{\sigma}_w$ denote mean and standard deviation of the predicted weights along a particular ray. The dynamic margin is then defined as

$$\epsilon_{dyn} = \epsilon_{min}(1 + \alpha \mathbf{J}^*) \tag{6}$$

$$\mathbf{J}^* = \begin{cases} 0 & JS(G||S) < JS_{min} \\ JS_{max} & JS(G||S) > JS_{max} \\ JS(G||S) & \text{otherwise,} \end{cases} \tag{7}$$

where α is a constant scaling parameter. JS_{max} denotes the upper bound of the JS score, and JS_{min} denotes a threshold for scaling. Once the JS score is smaller than JS_{min}, ϵ_{dyn} is equal to ϵ_{min}.

2) Depth Loss: As in [8], we use the depth loss as an additional term in the loss function. The depth loss is the error between rendered depth and LiDAR-measured depth along each ray. The loss is defined as

$$\mathcal{L}_{depth}(\Theta) = \|\hat{D}(\vec{r}) - z^*\|_2^2 \tag{8}$$

We found the depth loss contributes to blurry reconstruction with limited training time, but still provides good hole-filling, as shown in Fig. 6. Hence, unlike [8] which weights depth loss and LOS loss equally, we down-weight the depth loss by setting $\lambda_1 = 5 \times 10^{-6}$.

3) Sky Loss: Similar to [8], we add an additional loss to force weights on rays pointing at the sky to be zero. While [8] segments the sky with camera-based semantic segmentation, we determine sky regions by observing holes in the LiDAR scans. First, each scan is converted to a depth image. This is then filtered via a single dilate and erode. Any points which remain empty reflect regions of the LiDAR scan where no return was received. If the ray corresponding to each of these points has a positive elevation angle in the global frame, it is determined to point to the sky. Thus, this heuristic works as long as the LiDAR is approximately level during initialization. For sky rays, the opacity loss is not enforced. Then, for all sky rays, the following loss is computed:

$$\mathcal{L}_{sky}(\Theta) = \|w\|_1. \tag{9}$$

F. Meshing

To form a mesh from the implicit geometry, a virtual LiDAR is placed at estimated KeyFrame poses. We compute weights along LiDAR rays, then bucket the weights into a 3D grid. When multiple weights fall within the same grid cell, the

TABLE I
Parameters for LONER.

Description	Symbol	Value
Time per KeyFrame	t_{KF}	3
KeyFrame Window Size	N_W	8
Rays per KeyFrame	N_R	512
Samples per Ray	N_S	512
Min Depth Margin	ϵ_{min}	0.5
JS Scale Hyperparameter	α	1
Min/Max JS Divergence	JS_{min}, JS_{max}	1, 10
Loss Coefficients	λ_1, λ_2	5×10^{-6}, 1

TABLE II
Pose tracking results on Fusion Portable and Newer College sequences. Reported metric is RMS APE (m). An ✗ indicates the algorithm failed.

	MCR	**Canteen**	**Garden**	**Quad**
LeGO-LOAM	0.052	0.129	0.161	**0.126**
NICE-SLAM	0.248	✗	✗	-
LONER w./ $\mathcal{L}_{\text{URF}}$	0.047	0.952	0.928	0.931
LONER w./ $\mathcal{L}_{\text{CLONeR}}$	0.034	0.071	0.073	0.306
LONER	**0.029**	**0.064**	**0.056**	0.130

maximum value is kept. Marching cubes is then used to form a mesh from the result. This process runs offline for visualization and evaluation, and is not a part of online training.

IV. Experiments

This section evaluates the trajectory estimation and mapping accuracy of LONER against state-of-the-art baselines. We further evaluate the choice of loss function and perform ablation studies over key features.

A. Implementation Details

Table I provides parameters used for evaluation of LONER, which we found to generalize well across the tested datasets. All values were tuned experimentally to maximize performance while maintaining real-time operation. For all experiments, each method and configuration was run 5 times and we report the median result, as in [24]. The complete data from our evaluations is available on the project webpage.

B. Baselines

We evaluate against NICE-SLAM [6] and LeGO-LOAM [15], which represent state-of-the-art methods in neural-implicit SLAM and LiDAR SLAM respectively. Additionally, we evaluate our SLAM pipeline with the loss functions from CLONeR [9] and URF [8]. We refer to these approaches as "LONER w./ $\mathcal{L}_{\text{CLONeR}}$" and "LONER w./ $\mathcal{L}_{\text{URF}}$" respectively. Finally, mapping performance is compared to SHINE mapping, which is run with groundtruth poses [10]. Since NeRF-LOAM [11] is a recent work and code is not yet available, it is excluded from this evaluation in favor of SHINE. Note that NeRF-LOAM does not operate in real-time.

C. Datasets

We evaluate performance on two open source datasets, Fusion Portable [4] and Newer College [28]. Collectively, the chosen sequences represent a range of scales and difficulties. From Fusion Portable, we select three scenes. The first sequence is MCR Slow 01, which is a small indoor lab scene collected on a quadruped. The others are Canteen Day and Garden Day, which are medium-scale semi-outdoor courtyard areas collected on a handheld platform. Both sequences contain few dynamic objects, as handling dynamic objects is left to future work. From Newer College, we evaluate on the Quad Easy sequence, which consists of two laps of a large outdoor college quad area. Because Newer College has monochrome fisheye cameras, it is incompatible with NICE-SLAM. Hence, NICE-SLAM is excluded from the Newer College results.

Note that the sequences used in testing do not have RGB-D sensors. Hence, we instead simulate RGB-D from stereo offline using RAFT optical flow estimation [29], and run NICE-SLAM on the result. NICE-SLAM can fail to converge in these semi-outdoor scenarios in a real-time configuration, so we increased the number of samples and iterations to improve performance. We ran NICE-SLAM for 350 iterations per KeyFrame, used 2^{14} samples per ray, and selected a KeyFrame every 5 frames. This results in offline runtime performance. To bound the computational complexity, we set the middle and fine grid sizes to 0.64m and 0.32m respectively.

D. Performance Analysis

1) Trajectory Tracking Evaluation: Trajectory estimates from each algorithm are evaluated according to the procedure described in [30]. We use an open-source package for these evaluations[1]. Trajectories are aligned, then, the root-mean-squared absolute pose error is computed and referred to as t_{APE}.

Table II compares trajectory performance to state-of-the-art methods [15, 6]. Our method offers performance competitive with or better than existing state-of-the-art LiDAR SLAM. On the evaluated scenes, we outperform LeGO-LOAM except for on Newer College Quad, which is the largest and most open sequence. Even on Quad, our estimated trajectory is within millimeters of the LeGO-LOAM result. Additionally, unlike LeGO-LOAM, our method creates dense implicit maps of the scene. On the MCR sequence, NICE-SLAM successfully estimated a trajectory four of five times. The resulting trajectories were reasonable, but not competitive with the LiDAR-based methods. On the other larger sequences, NICE-SLAM failed to track the scene. This reflects that NICE-SLAM was developed for small indoor scenarios and does not scale to more challenging scenes.

这反映了NICE-SLAM是为小型室内场景开发的，不能扩展到更具挑战性的场景

LONER with the CLONeR loss achieves trajectory accuracy similar to LONER on some sequences. However, it consistently performs worse, especially in Quad. We found LONER using the URF loss is less competitive. We also tested LONER with the KL loss proposed by DS-NeRF [23]. However, it crashes when the initial pose estimation is poor because using fixed uncertainty for the goal distribution can cause numerical instability in the SLAM context.

2) Reconstruction Evaluation: To evaluate maps, point clouds are created by first generating a mesh, then sampling

[1]https://github.com/MichaelGrupp/evo

TABLE III
Comparison of map Acccuracy (m), Completion (m), Precision, and Recall between proposed and baseline algorithms. Unlike the others, SHINE used ground truth poses. A '-' indicates invalid configurations, while '✗' indicates that the algorithm failed.

		NICE SLAM	SHINE	LONER w./ $\mathcal{L}_{\text{CLONeR}}$	LONER w./ $\mathcal{L}_{\text{URF}}$	LONER
MCR	Acc.	0.621	0.164	**0.110**	0.153	0.186
	Cmp.	0.419	0.075	0.080	0.102	**0.069**
	Prec.	0.124	0.624	**0.665**	0.449	0.473
	Rec.	0.476	0.757	**0.940**	0.884	0.932
Canteen	Acc.	✗	-	-	-	-
	Cmp.		0.116	0.220	0.190	**0.105**
	Prec.		-	-	-	-
	Rec.		0.753	0.524	0.846	**0.878**
Garden	Acc.	✗	-	-	-	-
	Cmp.		**0.130**	0.333	0.539	0.157
	Prec.		-	-	-	-
	Rec.		0.657	0.469	0.623	**0.784**
Quad	Acc.	-	**0.301**	0.663	0.552	0.380
	Cmp.	-	**0.148**	0.543	0.895	0.373
	Prec.	-	**0.453**	0.150	0.127	0.327
	Rec.	-	0.717	0.602	0.484	**0.809**

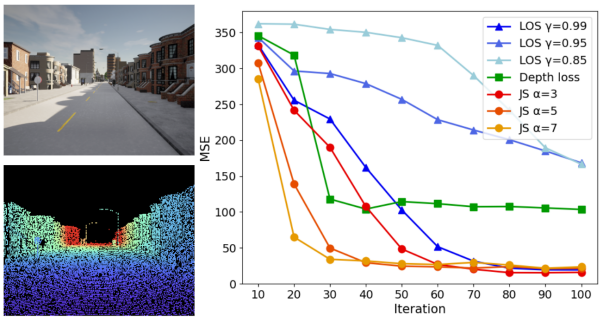

Fig. 4. Training on a single LiDAR scan from the CARLA simulator indicates that the JS loss function converges faster than alternatives. The left images show simulated camera and LiDAR data. The plot on the right compares MSE (m^2) between groundtruth depth and estimated depth throughout training.

a point cloud from the mesh. To bound the size of the generated point clouds, all maps (estimated and groundtruth) are downsampled to a voxel grid size of 5cm, except for the small MCR sequence, which uses a voxel grid size of 1cm. Finally, because groundtruth maps may extend beyond the field-of-view of the sensor used to collect each sequence, we crop each groundtruth map to the geometry observed by the sensor during data collection.

Map metrics include accuracy (mean distance from each point in the estimated map to each point in the groundtruth map) and completion (mean distance from each point in the groundtruth map to each point in the estimated map) [5, 6]. Additionally, precision and recall are computed with a 0.1m threshold. Table III shows quantitative evaluation for map reconstruction performance. LONER performs competitively with or better than the baselines in all tests. LONER and SHINE Mapping out-perform the other baselines. Qualitatively, Fig. 5 shows that SHINE and LONER estimate the most accurate maps. SHINE estimates more complete geometry, while LONER recovers finer detail and produces maps with fewer artifacts.

SHINE估计更完整的几何形状，而LONER恢复更精细的细节，
生成的地图具有更少的人工制品

E. Runtime

Runtime performance was evaluated on a computer with an AMD Ryzen 5950X CPU and an NVidia A6000 GPU, which is similar to the platform used to benchmark NICE-SLAM [6]. Each tracking step takes an average of 14ms to compute, which is faster than is needed by the 5Hz configuration. The map is updated continuously throughout operation, with 50 iterations allocated per KeyFrame and one KeyFrame added every 3 seconds. When run in parallel with the tracker, the average time to perform these 50 iterations is 2.79 seconds, or approximately 56ms per iteration. Hence, the map is updated at approximately 18Hz, and the system finishes processing a KeyFrame in under the 3 seconds allotted per KeyFrame. This ensures the system can keep up with input sensor data.

F. Loss Function Performance

We evaluate each component of the loss function in isolation. In addition to the JS dynamic margin loss, we evaluate an LOS loss with three different exponential decay rates: 0.99 (Slow), 0.95 (Medium), and 0.85 (Fast). Finally, we consider the depth loss in isolation, as is used in [6, 5]. As a qualitative comparison of mapping quality, depth images rendered from each configuration are shown in Fig. 6. The proposed JS loss shows the most complete and detailed reconstruction of the tested configurations.

Additionally, Fig. 4 demonstrates that JS loss converges faster than other losses. In this experiment, we evaluate the convergence of each function when training on a single scan in simulated data. We use the CARLA simulator, where we obtain groundtruth depth images for evaluations. We compute the mean squared error of rendered depth images throughout training to show convergence performance. The results show that our JS loss converges faster than other losses.

G. Ablation Study

This section describes the ablation studies over key components of the SLAM framework and the loss function. To compare maps in the ablations, we evaluate the L1 depth loss by comparing rendered depth to depth measured by the LiDAR. This is analogous to the L1 Depth metric commonly used in NeRF frameworks [5, 6, 7]. We compute the L1 depth across 25 randomly selected scans and report the mean value.

1) SLAM Framework: In Table IV, we compare the impact of three changes to the SLAM framework. Disabling pose optimization is confirmed to strongly impact localization and mapping. Replacing the random KeyFrame selection with either the N_W most recent or $N_W/2$ recent KeyFrames and $N_W/2$ randomly selected KeyFrames generally reduces performance. Finally, on the outdoor dataset, disabling sky segmentation has little effect on localization but degrades reconstruction accuracy.

2) Loss Function: Finally, we consider disabling features of the proposed loss function, which includes both the JS loss and the depth loss. In Table V, we evaluate using only depth loss, depth loss, and LOS loss with the fixed medium decay rate, LOS loss with dynamic margin and no depth loss, and the full system. The results demonstrate that the proposed system performs best in all metrics on all datasets.

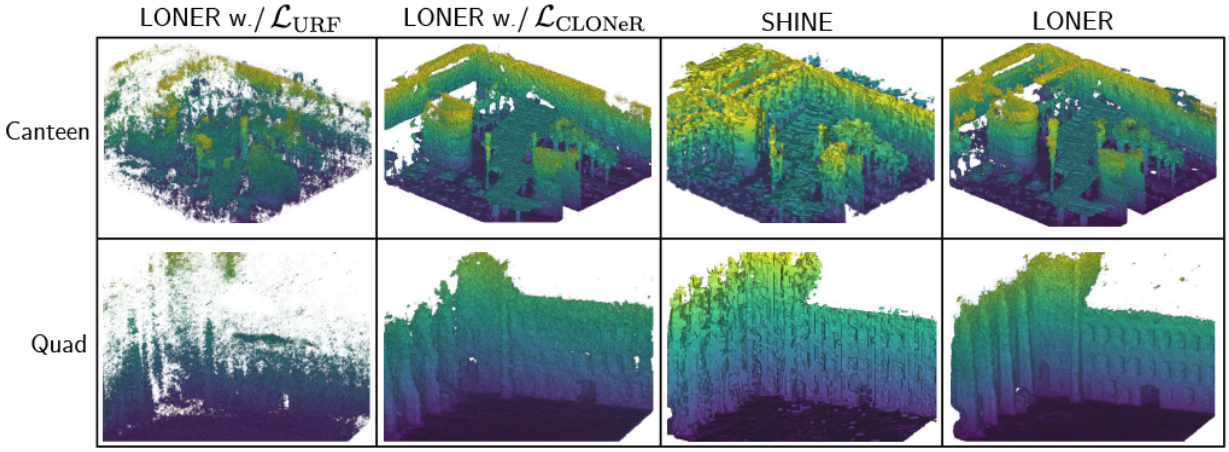

Fig. 5. Reconstruction of meshes on each sequence with the benchmarked algorithms. LONER and SHINE offer the most complete and detailed results. SHINE has slightly more complete geometry, noticeable in the top-left of the Quad images where LONER omits pillars captured by SHINE. However, LONER captures details better and has fewer artifacts.

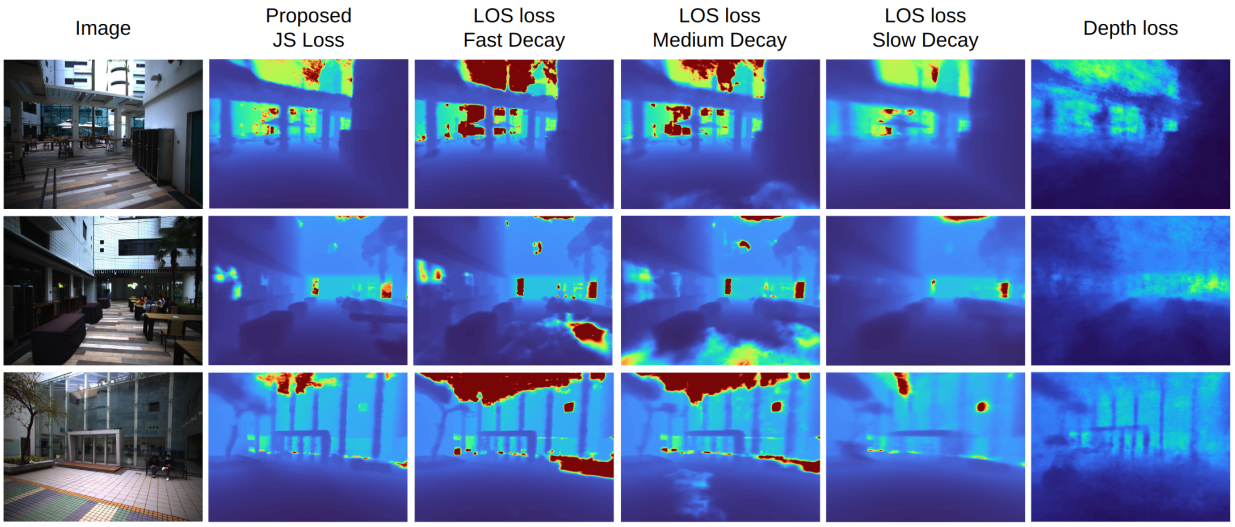

Fig. 6. The depth images rendered from the MLP trained by LONER with different loss functions. The depth loss provides blurry geometry with limited training samples. The LOS loss with a fast decay rate provides more detailed geometry but worse hole-filling. In contrast, the LOS loss with a slow decay rate estimates the untrained region better but results in blurry geometry. The proposed JS loss combines the advantages of both fast and slow decay rates, which provides good hole-filling results while preserving geometry details.

TABLE IV
We perform an ablation study over the SLAM framework by disabling key features, and show the proposed system outperforms alternatives.

Pose Optimization	**Sky Segmentation**	**Random KeyFrames**	**MCR**		**Canteen**		**Garden**		**Quad**	
			t_{APE}	L1 Depth	t_{APE}	L1 Depth	t_{APE}	L1 Depth	t_{APE}	L1 Depth
○	●	●	0.077	0.317	1.162	2.927	1.223	3.305	0.862	3.302
●	○	●	-	-	-	-	-	-	**0.128**	0.994
●	●	○	**0.028**	0.289	0.079	1.573	0.066	1.237	0.219	1.363
●	●	◐	0.029	0.285	0.065	1.298	0.057	1.261	0.130	**0.829**
●	●	●	0.029	**0.284**	**0.064**	**1.296**	**0.056**	**1.198**	0.130	0.880

TABLE V
We perform an ablation study over the loss. The first row uses only depth loss. The second uses depth loss and LOS loss with no dynamic margin. The third row uses LOS Loss with dynamic margin. The final row is the proposed system.

Depth Loss	**LOS Loss**	**Dynamic Margin**	**MCR**		**Canteen**		**Garden**		**Quad**	
			t_{APE}	L1 Depth	t_{APE}	L1 Depth	t_{APE}	L1 Depth	t_{APE}	L1 Depth
●	○	N/A	0.046	0.355	1.236	3.014	0.788	2.447	0.779	2.265
●	●	○	0.033	0.338	0.075	1.453	0.076	1.304	0.570	1.747
○	●	●	0.030	0.358	0.068	1.907	**0.056**	1.490	0.154	2.262
●	●	●	**0.029**	**0.284**	**0.064**	**1.296**	**0.056**	**1.198**	**0.130**	**0.880**

V. Conclusions and future work

This paper proposed LONER, the first real-time LiDAR SLAM algorithm with an implicit neural map representation. To achieve SLAM in real-time, we presented a novel loss function for depth-supervised training. Results demonstrated that the JS loss outperforms current loss functions in both reconstruction accuracy and hole-filling while maintaining low computational costs. By testing this method on public datasets, we demonstrated that LONER achieves state-of-the-art map and trajectory quality, while providing an implicit geometry representation to support novel view depth rendering.

There are several avenues of future work to continue improving LONER. First, adding RGB data without compromising runtime performance would aid in the realism of reconstructions. Additionally, considering alternate input feature embeddings and ray selection heuristics could improve the ability of LONER to operate in city-scale scenarios. Further, inertial data could help the system track accurately under rapid rotation and in feature-sparse scenarios, where the LiDAR data is less informative. Finally, to function in highly dynamic environments, more work is needed to handle dynamic objects in the scene.

References

[1] 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, p. 99–106, Dec 2021.

[2] M. Adamkiewicz, *et al.*, "Vision-Only Robot Navigation in a Neural Radiance World," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 4606–4613, Apr. 2022.

[3] X. Pan, Z. Lai, S. Song, and G. Huang, "Activenerf: Learning where to see with uncertainty estimation," in *2022 European Conference on Computer Vision*. Springer, 2022, pp. 230–246.

[4] J. Jiao, *et al.*, "Fusionportable: A multi-sensor campus-scene dataset for evaluation of localization and mapping accuracy on diverse platforms," *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 3851–3856, 2022.

[5] E. Sucar, S. Liu, J. Ortiz, and A. J. Davison, "imap: Implicit mapping and positioning in real-time," *2021 IEEE/CVF International Conference on Computer Vision*, pp. 6209–6218, 2021.

[6] Z. Zhu, *et al.*, "Nice-slam: Neural implicit scalable encoding for slam," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022.

[7] A. Rosinol, J. J. Leonard, and L. Carlone, "Nerf-slam: Real-time dense monocular slam with neural radiance fields," *ArXiv*, vol. abs/2210.13641, 2022.

[8] K. Rematas, *et al.*, "Urban radiance fields," *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022.

[9] A. Carlson, M. S. Ramanagopal, N. Tseng, M. Johnson-Roberson, R. Vasudevan, and K. A. Skinner, "Cloner: Camera-lidar fusion for occupancy grid-aided neural representations," *IEEE Robotics and Automation Letters*, vol. 8, no. 5, pp. 2812–2819, 2023.

[10] X. Zhong, Y. Pan, J. Behley, and C. Stachniss, "Shine-mapping: Large-scale 3d mapping using sparse hierarchical implicit neural representations," *arXiv preprint arXiv:2210.02299*, 2022.

[11] J. Deng, *et al.*, "Nerf-loam: Neural implicit representation for large-scale incremental lidar odometry and mapping," *arXiv preprint arXiv:2303.10709*, 2023.

[12] D. Yan, X. Lyu, J. Shi, and Y. Lin, "Efficient implicit neural reconstruction using lidar," *arXiv preprint arXiv:2302.14363*, 2023.

[13] J. Zhang and S. Singh, "Loam: Lidar odometry and mapping in real-time," in *Robotics: Science and Systems*, 2014.

[14] J. Behley and C. Stachniss, "Efficient surfel-based slam using 3d laser range data in urban environments." in *Robotics: Science and Systems*, 2018.

[15] T. Shan and B. Englot, "Lego-loam: Lightweight and ground-optimized lidar odometry and mapping on variable terrain," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2018, pp. 4758–4765.

[16] H. Ye, Y. Chen, and M. Liu, "Tightly coupled 3d lidar inertial odometry and mapping," in *IEEE International Conference on Robotics and Automation*, 2019, pp. 3144–3150.

[17] T. Shan, B. Englot, D. Meyers, W. Wang, C. Ratti, and D. Rus, "Lio-sam: Tightly-coupled lidar inertial odometry via smoothing and mapping," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2020, pp. 5135–5142.

[18] Z. Teed and J. Deng, "Droid-slam: Deep visual slam for monocular, stereo, and rgb-d cameras," in *34 - 35th Conference on Neural Information Processing Systems*. Neural information processing systems foundation, 2021, pp. 16 558–16 569.

[19] E. Kruzhkov, *et al.*, "Meslam: Memory efficient slam based on neural fields," in *2022 IEEE International Conference on Systems, Man, and Cybernetics*, 2022, pp. 430–435.

[20] H. Wang, J. Wang, and L. Agapito, "Co-slam: Joint coordinate and sparse parametric encodings for neural real-time slam," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 13 293–13 302.

[21] M. M. Johari, C. Carta, and F. Fleuret, "Eslam: Efficient dense slam system based on hybrid representation of signed distance fields," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 17 408–17 419.

[22] 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 *2022 IEEE International Symposium on Mixed and Augmented Reality*, 2022, pp. 499–507.

[23] K. Deng, A. Liu, J.-Y. Zhu, and D. Ramanan, "Depth-supervised nerf: Fewer views and faster training for free," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 12 882–12 891.

[24] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardós, "Orb-slam: A versatile and accurate monocular slam system," *IEEE Transactions on Robotics*, vol. 31, no. 5, pp. 1147–1163, 2015.

[25] S. Rusinkiewicz and M. Levoy, "Efficient variants of the icp algorithm," in *Proceedings Third International Conference on 3-D Digital Imaging and Modeling*, 2001, pp. 145–152.

[26] T. Müller, A. Evans, C. Schied, and A. Keller, "Instant neural graphics primitives with a multiresolution hash encoding," *ACM Trans. Graph.*, vol. 41, no. 4, pp. 102:1–102:15, July 2022.

[27] J. Lin, "Divergence measures based on the shannon entropy," *IEEE Transactions on Information Theory*, vol. 37, no. 1, pp. 145–151, 1991.

[28] 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 *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2020, pp. 4353–4360.

[29] Z. Teed and J. Deng, "Raft: Recurrent all-pairs field transforms for optical flow," in *European Conference on Computer Vision*, A. Vedaldi, H. Bischof, T. Brox, and J.-M. Frahm, Eds. Springer, 2020, pp. 402–419.

[30] Z. Zhang and D. Scaramuzza, "A tutorial on quantitative trajectory evaluation for visual(-inertial) odometry," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2018, pp. 7244–7251.