Improving Indoor Localization Accuracy by Using an Efficient Implicit Neural Map Representation

Haofei Kuang Yue Pan Xingguang Zhong Louis Wiesmann Jens Behley Cyrill Stachniss

***Abstract*— Globally localizing a mobile robot in a known map is often a foundation for enabling robots to navigate and operate autonomously. In indoor environments, traditional Monte Carlo localization based on occupancy grid maps is considered the gold standard, but its accuracy is limited by the representation capabilities of the occupancy grid map. In this paper, we address the problem of building an effective map representation that allows to accurately perform probabilistic global localization. To this end, we propose an implicit neural map representation that is able to capture positional and directional geometric features from 2D LiDAR scans to efficiently represent the environment and learn a neural network that is able to predict both, the non-projective signed distance and a direction-aware projective distance for an arbitrary point in the mapped environment. This combination of neural map representation with a light-weight neural network allows us to design an efficient observation model within a conventional Monte Carlo localization framework for pose estimation of a robot in real time. We evaluated our approach to indoor localization on a publicly available dataset for global localization and the experimental results indicate that our approach is able to more accurately localize a mobile robot than other localization approaches employing occupancy or existing neural map representations. In contrast to other approaches employing an implicit neural map representation for 2D LiDAR localization, our approach allows to perform real-time pose tracking after convergence and near real-time global localization. The code of our approach is available at: `https://github.com/PRBonn/enm-mcl`.**

I. Introduction

Estimating the state of a robot in terms of its position and orientation in an environment is crucial to enable robots to operate autonomously, but it is also a key requirement for realizing autonomous navigation and planning. Localization in a pre-built map is a common way to realize state estimation for robotic systems, where probabilistic methods using various map representations are often employed. In indoor environments, localization via external sources of global positioning information, such as GNSS data, is typically not available, therefore, global localization must rely on onboard sensors, like wheel odometry and 2D LiDAR sensors. A gold standard approach for that is Monte Carlo localization [5].

In this paper, we investigate the problem of building an effective map for achieving accurate and efficient global localization and ego-pose tracking in indoor environments.

All authors are with the Center for Robotics, University of Bonn, Germany. Cyrill Stachniss is additionally with the Department of Engineering Science at the University of Oxford, UK, and with the Lamarr Institute for Machine Learning and Artificial Intelligence, Germany.

This work has partially been funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy, EXC-2070 – 390732324 – PhenoRob, and by the German Federal Ministry of Education and Research (BMBF) in the project "Robotics Institute Germany" under grant No. 16ME0999.

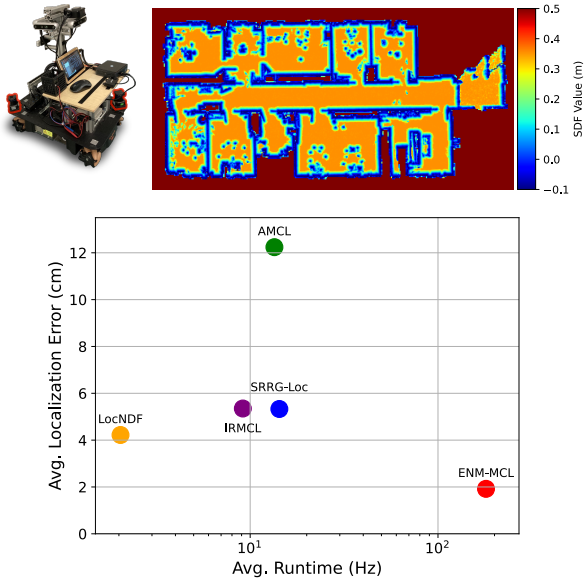

Fig. 1: We learn a neural network to represent the surfaces of environments. Our method achieves both efficiency and accuracy for indoor localization by integrating our proposed efficient neural map representation into a Monte Carlo localization system. We show the average runtime for each method in sequence 1 of the in-house dataset.

Specifically, we aim to build a map representation that can be used for implementing an accurate and computationally efficient probabilistic localization approach, where we employ a conventional Monte Carlo localization (MCL) and use our map representation for an efficient observation model.

A classical approach to build an effective map representation is occupancy grid mapping [23] often used to realize MCL [7], [8], [31] in indoor environments. Recently, several approaches [11], [24] investigated the usage of novel implicit neural map representations to replace the commonly employed occupancy grid maps due to promising prospects regarding representing scene details continuously with a comparably small memory footprint and scene completion capabilities [28]. While these approaches demonstrate the advantages of an implicit map representation over conventional occupancy maps, they typically require significant computational resources for training and deployment in global localization, especially in large-scale environments.

To address these key limitations of current implicit neural map representation-based global localization methods [11], [24], we propose a novel approach that improves the efficiency of using the map representation in an MCL-based localization, while simultaneously increasing the localization accuracy at the same time, as we show in Fig. 1.

The main contribution of this paper is a novel implicit map representation, which we call efficient neural map (ENM), that allows to learn the surface information of a mapped environment through a dense feature grid combined with a light-weight neural network enabling efficient global localization in large indoor environments with a large number of particles. The light-weight neural network learns an approximate regular (non-projective) signed distance field (SDF), but is also able by incorporating information of the ray direction to accurately predict a direction-aware projective SDF (PSDF) capturing both, geometric and directional features of the environment. We integrate our ENM representation into the existing MCL framework by replacing the occupancy grid map used to compute a beam end point-based observation model [23], resulting in the approach we call ENM-MCL. Our experimental evaluation on multiple publicly available sequences for indoor localization indicates that our ENM-MCL is able to more accurately localize a mobile robot than other probabilistic localization approaches employing occupancy or so far used neural representations. Furthermore, the results show that ENM-MCL is able to perform real-time pose tracking after convergence, while allowing near real-time global localization.

In sum, we make three key claims: (i) Our map representation, ENM, used by ENM-MCL achieves more accurate global localization compared to other localization approaches employing conventional occupancy grid maps or neural map representations; (ii) Our ENM-MCL system operates at real time for pose tracking but also enables efficient global localization even with a large number of particles; (iii) Our ENM-MCL converges quickly to the correct pose estimate enabling fast and reliable localization in indoor environments. We will publish the source code of our ENM-MCL.

II. Related Work

Pose tracking and global localization for mobile robots are key research areas in robotics as determining the location of a robot in a given map is crucial for many downstream tasks. Regarding global localization for mobile robots, Monte Carlo localization is a gold standard method, which exploits a particle filter to estimate the robot's state in a probabilistic manner and has been implemented with various sensors, such as 2D LiDARs [5], [8], [22] and cameras [2], [30]. To improve the efficiency of the MCL algorithm, Fox et al. [8] propose an adaptive sampling strategy, which adapt the size of the sample set on the fly. Traditionally, MCL methods for LiDAR-based localization often rely on occupancy grid maps [4], [8], [31] for 2D LiDARs, but also for 3D LiDARs [3], [13].

The representational capacity of occupancy maps is constrained by their pre-defined grid resolution, where simply increasing the resolution can substantially increase memory consumption. To overcome this limitation of a fixed resolution, some approaches for LiDAR-based localization have been proposed to construct a multi-resolution map [6] or learn continuous implicit map representations via Gaussian processes [16], [26], reproducing kernel Hilbert maps [20]. Nonetheless, these methods are quite time-consuming and challenging to directly apply to real-world scenarios.

Recently, implicit neural map representations have gained popularity due to their compactness and continuous representational capacity. These representations have been used to model complex geometric information, such as learning a radiance field [1], [15], [21] from images to representing the 3D world [17], [19]. Consequently, NeRFs have been employed for visual localization tasks, such as global localization [14] or re-localization [12]. Other methods [17], [19] learn a signed distance field from LiDAR or RGB-D camera as a more accurate geometric representation of the environment. Moreover, some works [25], [27] use the neural network not only to encode the geometry or appearance of the scene, but also the semantics of the environment. Due to their high representational capacity of the employed neural network, several works [11], [24] use a single MLP to represent the entire scene and integrate the implicit neural map representation into an observation model of MCL to achieve global localization. Although these methods exhibit good map representation capabilities, the reconstruction process is very time-consuming and the time needed to querying the representation does not meet the required efficiency for real-time localization.

To address the challenge with respect to efficiency, recent works [18], [28], [29] have introduced feature fields combined with shallow MLPs as a representation of the environment. This approach not only accelerates convergence of training process but also enhances the quality and capacity of the map representation. However, these methods have primarily been applied to 3D reconstruction tasks and have not yet been efficiently deployed in 2D indoor localization scenarios.

In this work, we leverage the high capacity of neural networks to learn both the non-projective SDF and a direction-aware projective SDF by encoding the ray direction of 2D LiDAR into the neural map representation. Additionally, we propose a novel implicit representation based on a dense feature grid combined with a light-weight neural network, which substantially reduces computational cost, especially for querying the neural map representation. Our novel implicit neural map representation enables more detailed modeling of the environment while improving the accuracy and efficiency of global localization using 2D LiDAR scans.

III. Our Approach for Global Localization

The goal of our approach is to accurately localize a mobile robot in a given map by learning an efficient neural map representation of the environment. More specifically, we propose to represent the environment via a feature grid-based representation, which is learned from 2D LiDAR scans recorded in a mapping session and allows to predict the signed distance to the surface at an arbitrary position via a shallow multi-layer perceptron (MLP), see Sec. III-A. For learning the implicit neural map representation, we supervise learnable features of the feature grid and the weights of the employed MLP with real measurements from a 2D LiDAR as

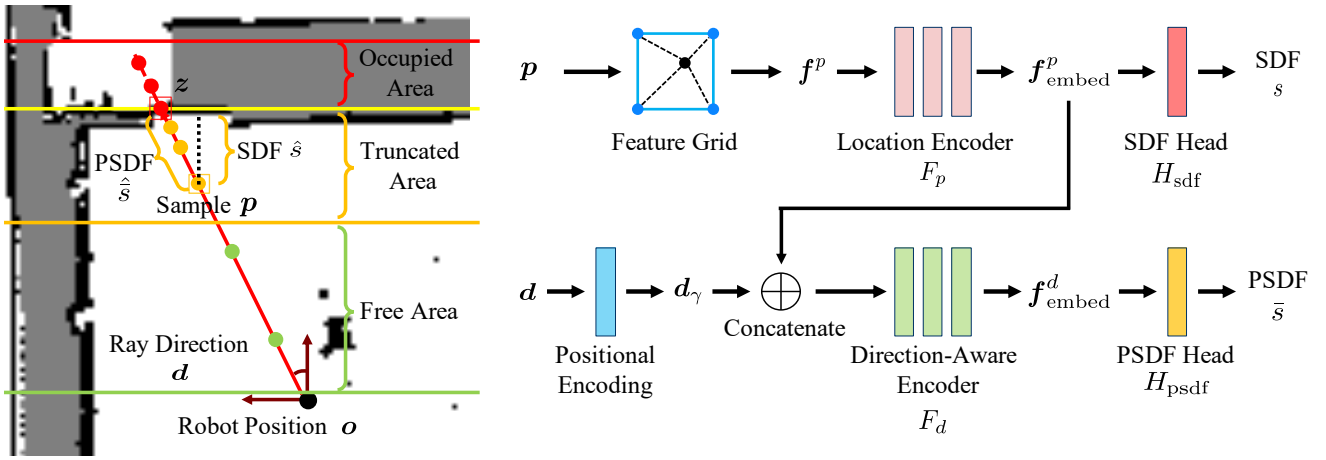

Fig. 2: Overview of our approach for jointly predicting the non-projective SDF and direction-aware projective SDF values using our proposed efficient neural map representation. We sample several positions along a LiDAR ray and input the 2D position $\boldsymbol{p} = (x, y)^\top$ as well as the corresponding ray direction $\boldsymbol{d} = (d_x, d_y)^\top$ into our ENM model to estimate the SDF and PSDF. The predictions of the SDF and PSDF values by the neural network are supervised with the ground truth SDF/PSDF values from 2D LiDAR measurements.

presented in Sec. III-B. After that, we use the efficient neural map representation in an MCL-based localization approach, where we use the estimated SDF for an observation model to update the weights of the particles, as described in more detail in Sec. III-C.

A. Efficient Neural Map Representation

For our map representation, we want to leverage a neural representation that allows for estimating the signed distance to the surface in the environment. More specifically, we propose an implicit neural surface representation, which is a function $F_{\Theta,\mathbf{G}}$ that takes the 2D location vector $\boldsymbol{p} = (x, y)^\top$ and 2D Cartesian unit vector $\boldsymbol{d} = (d_x, d_y)^\top$ as inputs, and predicts the corresponding non-projective signed distance field (SDF) value s and direction-aware projective signed distance field (PSDF) value $\bar{s}$. The SDF value represents the shortest distance from the point $\boldsymbol{p}$ to the nearest surface in the environment and the PSDF is the distance from a point $\boldsymbol{p}$ to the surface along the specific ray direction $\boldsymbol{d}$. Formally, we have the following:

$$s, \bar{s} = F_{\Theta,\mathbf{G}}(\boldsymbol{p}, \boldsymbol{d}). \tag{1}$$

We represent $F_{\Theta,\mathbf{G}}$ by a MLP, where Θ represents the weights of the MLP and $\mathbf{G}$ corresponds to a dense grid of learnable feature vectors. By combining a light-weight network with a dense feature grid, we can substantially reduce the computation-cost meanwhile maintaining the quality of the map model, which supports to use it with a particle filter for localization in real time.

Our dense feature grid $\mathbf{G}$ has a given grid resolution $\Delta_{\mathbf{G}}$. Each grid corner at the location coordinate (i, j) stores a D-dimensional feature vector $\boldsymbol{c}^g_{i,j} \in \mathbb{R}^D$. During operation, the input 2D location $\boldsymbol{p}$ is first encoded into a location feature $\boldsymbol{f}^p \in \mathbb{R}^D$ by performing bilinear interpolation on the dense feature grid $\mathbf{G}$. We use the neural network to estimate the SDF s and PSDF $\bar{s}$ of input 2D location $\boldsymbol{p}$ and ray direction $\boldsymbol{d}$, which consist of two branches, as shown in Fig. 2.

In the SDF branch, the location feature $\boldsymbol{f}^p$ is first processed by an encoder F_p, a 3-layer MLP, to extract the positional embedding $\boldsymbol{f}^p_{\text{embed}}$. Then, $\boldsymbol{f}^p_{\text{embed}}$ is decoded into the s value of the input location by the SDF head H_{sdf}, which is a 1-layer MLP. Formally, we have the following:

$$\boldsymbol{f}^p_{\text{embed}} = F_p(\boldsymbol{f}^p), \tag{2}$$

$$s = H_{\text{sdf}}(\boldsymbol{f}^p_{\text{embed}}). \tag{3}$$

In the PSDF branch, the extracted positional embedding $\boldsymbol{f}^p_{\text{embed}}$ is concatenated with the positional encoding of the directional vector $\boldsymbol{d}_\gamma$ and fed into another 3-layer MLPs F_d, to extract the direction-aware embedding $\boldsymbol{f}^d_{\text{embed}}$, and then predict the $\bar{s}$ value of the input location and direction by a PSDF head H_{psdf}, which is also a 1-layer MLP, as follows:

$$\boldsymbol{f}^d_{\text{embed}} = F_d(\boldsymbol{f}^p_{\text{embed}} \oplus \boldsymbol{d}_\gamma), \tag{4}$$

$$\bar{s} = H_{\text{psdf}}(\boldsymbol{f}^d_{\text{embed}}), \tag{5}$$

where $\oplus$ is the concatenation of two vectors. To enable the model to capture high-frequency geometric features of the ray direction, we encode the directional vector $\boldsymbol{d}$ via a positional encoding γ, where we apply γ to each component [15], i.e., $\boldsymbol{d}_\gamma = (\gamma(d_x), \gamma(d_y))^\top$. The positional encoding function $\gamma : \mathbb{R} \mapsto \mathbb{R}^{2L+1}$ is defined as:

$$\gamma(d) = \left(d, \sin(2^0 d), \cos(2^0 d), \ldots, \sin(2^{L-1} d), \cos(2^{L-1} d)\right), \tag{6}$$

where L is the number of frequency bands used.

The networks F_p and F_d are shallow MLPs. Each hidden layer has D neurons and $D + 2(2L + 1)$ neurons for F_p and F_d, respectively. D is the dimension of the feature vector from $\mathbf{G}$ and L is the bandwidth of γ. In our model, we set $D = 4$ and $L = 4$, and each layer is followed by a ReLU activation. These shallow MLPs enable our model to run in real-time, even with a large number of particles.

B. Learning the ENM from 2D LiDAR Data

We learn our ENM representation using 2D LiDAR data from scans recorded in a mapping run. Given a posed 2D LiDAR scan with a set of rays $\mathcal{B} = \{(r_j, \boldsymbol{d}_j)\}$, where $\boldsymbol{d}_j$ are

ray directions and r_j are range readings, we sample positions on these rays for a training set $\mathcal{S}$ for each scan.

We sample training data by selecting points along each LiDAR ray within three regions: the truncated space, occupied space, and free space, see Fig. 2 left. The truncated space is the area in front of the surface, and the occupied space is the area behind the surface. For each ray, we randomly sample M_t samples in the truncated space, denoted as $\mathcal{S}_t$, and fewer M_o samples in the occupied space, denoted as $\mathcal{S}_o$, to encourage the model to learn the surface features. Furthermore, we sample a small number of M_s samples in the free space, denoted as $\mathcal{S}_f$, as these areas contribute less to learn the fine geometric details. The set of sampled points along each ray of a single LiDAR scan is then given by $\mathcal{S} = \mathcal{S}_t \cup \mathcal{S}_o \cup \mathcal{S}_f$. Fig. 2 shows the sampling process visually.

To optimize the weights and learnable feature vectors of $F_{\Theta,\mathbf{G}}$, we supervise the predicted s and $\bar{s}$ values using the generated ground truth from the training samples. Regarding the PSDF loss $\mathcal{L}_{\text{psdf}}$, we can directly generate the ground-truth projective SDF values from the range readings of each LiDAR ray. Let $\boldsymbol{p}$ be a sample on the LiDAR ray, then the ground-truth PSDF value $\hat{\bar{s}}$ is given by the distance between real measured distance r and the distance between $\boldsymbol{p}$ and the LiDAR origin $\boldsymbol{o}$:

$$\hat{\bar{s}} = r - \|\boldsymbol{p} - \boldsymbol{o}\|_2. \tag{7}$$

The loss $\mathcal{L}_{\text{psdf}}$ for the projected SDF is then computed using only the sampled points near the surface, $\mathcal{S}_t \cup \mathcal{S}_o$, as:

$$\mathcal{L}_{\text{psdf}} = \frac{1}{|\mathcal{S}_t| + |\mathcal{S}_o|} \sum_{(\boldsymbol{p},\boldsymbol{d}) \in \mathcal{S}_t \cup \mathcal{S}_o} |\bar{s} - \hat{\bar{s}}|, \tag{8}$$

where $\boldsymbol{d}$ is the corresponding ray direction of sample $\boldsymbol{p}$ and $\bar{s}$ is the predicted PSDF from $F_{\Theta,\mathbf{G}}(\boldsymbol{p}, \boldsymbol{d})$.

Regarding the SDF loss $\mathcal{L}_{\text{sdf}}$, we activate the predicted SDF values using a sigmoid function σ, and minimize a binary cross-entropy loss as the SDF objective $\mathcal{L}_{\text{sdf}}$, formulated as:

$$\mathcal{L}_{\text{sdf}} = -\frac{1}{|\mathcal{S}|} \sum_{(\boldsymbol{p},\boldsymbol{d}) \in \mathcal{S}} \Big(\sigma(\hat{s}) \log \sigma(s) + (1 - \sigma(\hat{s})) \log(1 - \sigma(s)) \Big), \tag{9}$$

where $s = F_{\Theta,\mathbf{G}}(\boldsymbol{p}, \boldsymbol{d})$ is the predicted signed distance value for the input sample, and $\sigma(\hat{s}) \in [0, 1]$ is the ground-truth value indicating the probability of the point being inside or outside the surface. Since we cannot directly obtain the non-projective SDF value from LiDAR range readings, we supervise the SDF prediction in an approximate manner, i.e., $\sigma(\hat{s}) \approx \sigma(\hat{\bar{s}})$, inspired by prior work [28].

Additionally, we incorporate an Eikonal loss $\mathcal{L}_{\text{eikonal}}$ to regularize the SDF [18], [28]. The Eikonal loss ensures that the gradient of the predicted SDF ∇s satisfies the Eikonal equation $\|\nabla s\| = 1$, which helps to enforce SDF smoothness and prevents the SDF from unrealistic deformations, leading to a more accurate neural map representation. The Eikonal loss $\mathcal{L}_{\text{eikonal}}$ is defined as:

$$\mathcal{L}_{\text{eikonal}} = \frac{1}{|\mathcal{S}_t| + |\mathcal{S}_o|} \sum_{(\boldsymbol{p},\boldsymbol{d}) \in \mathcal{S}_t \cup \mathcal{S}_o} (\|\nabla s_i\| - 1)^2. \tag{10}$$

By minimizing this loss, we ensure that the predicted SDF adheres to the properties of a signed distance function, improving both stability and accuracy in the learned map representation. To sum up, our final loss objective is:

$$\mathcal{L}_{\text{final}} = \mathcal{L}_{\text{sdf}} + \mathcal{L}_{\text{psdf}} + \beta \mathcal{L}_{\text{eikonal}}. \tag{11}$$

We optimize the loss function of the ENM model using the Adam optimizer [10] with a learning rate of 0.001, and we use $\beta = 0.1$ for the Eikonal loss. $\mathcal{S}$. We sample $M_t = 6$ positions in the truncated space, $M_o = 4$ positions in the occupied space, and $M_f = 5$ positions in the free space. The training process runs for 5,000 iterations, at each iteration we randomly sample batch of rays from all training LiDAR scans with a batch size of 2,048.

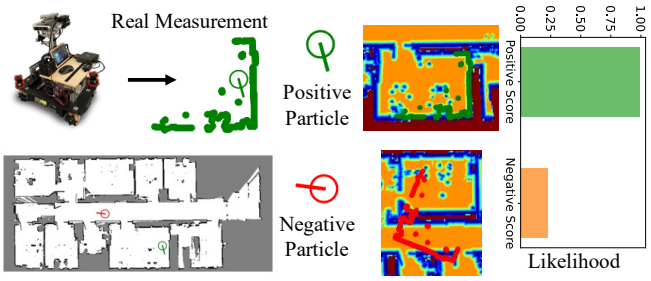

Fig. 3: The observation model based on the ENM representation. We estimate the SDF and PSDF values for all beam end-points of the particles using the ENM model, and update the particles' weights by computing the likelihood based on the SDF and PSDF values of the beam end-points. The positive particle has higher likelihood which scans are aligned with the zero level of distance field.

C. Efficient Neural Map-based MCL

Given our new map representation, we need to integrate it into MCL as our localization approach, ENM-MCL. We want to estimate the SE(2) state $\boldsymbol{x}_t = (x, y, \theta)_t^\top$ of the robot at time t defined by the 2D position $(x, y)^\top$ and the orientation $\theta \in [-\pi, \pi]$. To this end, we estimate the posterior $p(\boldsymbol{x}_t | \boldsymbol{z}_t, m)$ of the robot state $\boldsymbol{x}_t$ at time t with observations $\boldsymbol{z}_t$ and map m using a recursive Bayes filter [23], which is realized in MCL [5] via a particle filter. The particle filter approximates $p(\boldsymbol{x}_t | \boldsymbol{z}_t, m)$ by a set of particles $\mathcal{N} = \{(\boldsymbol{x}_t^n, w_t^n)\}$, $|\mathcal{N}| = N$, where each particle is a hypothesis of the robot's state $\boldsymbol{x}_t^n = (x_t^n, y_t^n, \theta_t^n)^\top$ with its corresponding weight w_t^n. The weights w_t^n are updated based on the likelihood $p(\boldsymbol{z}_t | \boldsymbol{x}_t^n, m)$ of the observation $\boldsymbol{z}_t$, commonly called observation model.

We use as observation model the conventional beam end-point model of MCL, but adapt it to employ our map representation. Here, we want to achieve that the observation model results in a higher likelihood if a particle hypothesis is closer to the real robot state, which means that the beam end points have the small SDF and PSDF values with our map representation. Specifically, we transfer the measured LiDAR

scan to a particle's state $\boldsymbol{x}_t^n$ at time t and then check the SDF and PSDF value of end-point $\boldsymbol{z}_t^i$ for each ray $(r_t^i, \boldsymbol{d}_t^i) \in \mathcal{B}^t$, formally as:

$$\boldsymbol{z}_t^i = \mathbf{R}_t^n (r_t^i \boldsymbol{d}_t^i) + t_t^n, \tag{12}$$

where $\mathbf{R}_t^n \in \mathbb{R}^{2\times 2}$ is the 2D rotation matrix for angle θ_t^n, and $t_t^n = (x_t^n, y_t^n)^\top$. Then, we exploit our ENM model $F_{\Theta,\mathbf{G}}$ to estimate the SDF value s_t^i of $\boldsymbol{z}_t^i$, and the PSDF value $\bar{s}_t^i$ of $(\boldsymbol{z}_t^i, \mathbf{R}_t^n \boldsymbol{d}_t^i)$ as:

$$s_t^i, \bar{s}_t^i = F_{\Theta,\mathbf{G}}(\boldsymbol{z}_t^i, \mathbf{R}_t^n \boldsymbol{d}_t^i), \tag{13}$$

where $\mathbf{R}_t^n \boldsymbol{d}_t^i$ is i-th the ray direction in the reference frame of the particle. Then, the likelihood of $\boldsymbol{z}_t^i$ is given by:

$$p(\boldsymbol{z}_t \mid \boldsymbol{x}_t^n, F_{\Theta,\mathbf{G}}) \propto \exp\left(-\lambda \frac{1}{|\mathcal{B}^t|} \sum_{i=1}^{|\mathcal{B}^t|} \frac{|s_t^i| + |\bar{s}_t^i|}{2}\right). \tag{14}$$

Similar to other localization systems [9], we compute an average alignment for each scan to the map. Furthermore, we average the predictions from the ENM model since it reduces the impact of noise or inaccuracies in either s_t^i or $\bar{s}_t^i$, resulting in a more robust and consistent likelihood estimation.

IV. Experimental Evaluation

The focus of this work is an efficient MCL system using an efficient and accurate neural map representation. We present our experiments to show the capabilities of our method. The results of our experiments support our key claims, which are: (i) the method achieves high accuracy in localization by representing the environment with the ENM model; (ii) our ENM-MCL system operates in real-time at pose tracking but also enables efficient global localization even with large numbers of particles; (iii) our algorithm converges quickly to the correct pose estimation to support rapid and reliable localization.

A. Experimental Setup

We evaluate our global localization results using the in-house dataset, previously used in prior work [11], [24]. It was collected with a KuKa YouBot equipped with a UTM-30LX 2D LiDAR and an upward-facing camera (not used for localization). The ground-truth robot poses were generated by localizing a large number of AprilTags on the ceiling detected by the upward-facing camera, as described in our prior work [11], providing a reference trajectory as near ground truth with a global position error around 1 cm, often even better. The dataset covers different indoor scenes such as offices, a kitchen, and a long corridor. It contains a 31,608-frame mapping sequence for training ENM and five test sequences averaging 1,419 frames for localization evaluation.

To demonstrate the localization accuracy of our method, we compare with four existing LiDAR-based localization algorithms: AMCL [8], SRRG-Loc [9], IRMCL [11], and LocNDF [24]. AMCL and SRRG-Loc are both occupancy grid map-based methods. The AMCL is a widely used global localization method from the standard ROS1 implementation. The SRRG-Loc [9] is developed by the Sapienza Robust Robotics Group (SRRG), which is a well-designed MCL implementation by Giorgio Grisetti and is based on occupancy grid maps. In contrast, IRMCL and LocNDF are two state-of-the-art methods based on implicit neural representations.

Seq	Method	Location RMSE (cm) ↓	Yaw RMSE (degree) ↓	Success Rate (%)
1	AMCL	12.24 ± 0.33	2.08 ± 0.08	40.0
	SRRG-Loc	5.33 ± 0.02	0.75 ± 0.00	100.0
	IRMCL	5.35 ± 0.07	1.06 ± 0.01	100.0
	LocNDF	4.06 ± 0.06	0.60 ± 0.00	60.0
	ENM-MCL	**1.96 ± 0.07**	**0.41 ± 0.00**	**100.0**
2	AMCL	10.28 ± 0.00	0.86 ± 0.00	100.0
	SRRG-Loc	6.59 ± 0.02	1.09 ± 0.00	100.0
	IRMCL	5.53 ± 0.06	0.82 ± 0.01	60.0
	LocNDF	**4.06 ± 0.93**	0.62 ± 0.03	80.0
	ENM-MCL	4.18 ± 0.06	**0.60 ± 0.01**	**100.0**
3	AMCL	-	-	0.0
	SRRG-Loc	-	-	0.0
	IRMCL	4.70 ± 0.13	0.77 ± 0.04	100.0
	LocNDF	3.85 ± 0.13	**0.72 ± 0.01**	100.0
	ENM-MCL	**3.05 ± 0.03**	0.74 ± 0.01	**100.0**
4	AMCL	-	-	0.0
	SRRG-Loc	-	-	0.0
	IRMCL	11.72 ± 0.31	1.57 ± 0.12	100.0
	LocNDF	10.89 ± 0.55	1.55 ± 0.16	60.0
	ENM-MCL	**5.82 ± 0.12**	**0.97 ± 0.08**	**100.0**
5	AMCL	-	-	0.0
	SRRG-Loc	6.29 ± 0.06	1.03 ± 0.05	100.0
	IRMCL	6.12 ± 0.03	1.26 ± 0.01	100.0
	LocNDF	-	-	0.0
	ENM-MCL	**2.40 ± 0.02**	**0.54 ± 0.01**	**100.0**

TABLE I: Quantitative results of global localization on the in-house dataset. We report the ATE for both location and orientation RMSE, along with the success rate of each method over five runs. The ATE is only reported if at least one run was successful; otherwise, '-' indicates failure.

B. Global Localization Performance

The first experiment evaluates the performance of our approach and its outcomes support the claim that our method achieves the state-of-the-art accuracy for global localization.

Regarding the parameters of our MCL system, we use a large particle number N = 80,000 during the initialization phase, where particles are uniformly distributed across the map with random orientations for global localization, and reduce it to N = 1,000 for efficient pose tracking after convergence. This adaptive adjustment for the particle number balances the trade-off between reliability and computational cost for the MCL system. To evaluate the localization results, we compute the absolute trajectory error (ATE) between the predicted and ground-truth trajectories. It includes the RMSE of translation (in centimeters) and orientation (in degrees). To reduce the sensitivity to randomness in the particle filter, we run five times with different random seeds, and take the average as the final ATE for each sequence.

The comparison results are shown in Tab. I. Our method demonstrates better accuracy compared to the baseline methods across all sequences. Even in challenging scenarios, such as sequence 3 where the robot starts in a corridor, our method substantially outperforms other approaches, leading to a 35.7% improvement over IRMCL. Meanwhile occupancy grid map-based methods failed entirely without parameter

Fig. 4: Qualitative global localization results of ENM-MCL on all five sequences of the in-house dataset. We show the predicted trajectories after convergence. The predicted trajectories are mostly aligned with the ground truth, indicating the high accuracy and reliability of our method.

tuning in this sequence. Furthermore, the results highlight that our method is more robust than the baselines, achieving a 100% success rate on all five sequences. This indicates that our model offers a better geometric representation to support more reliable localization in complex indoor environments. The predicted trajectories of our method are shown in Fig. 4.

C. Runtime Analysis

The second experiment evaluates the runtime for both training and localization, illustrating the efficiency of our approach. Specifically, we measure the runtime during performing MCL in initialization and pose tracking phase, and also propose the average frame rate on sequence 1 of the in-house dataset. We compare our method with IRMCL [11] and LocNDF [24], two other implicit representation-based MCL methods. Since all these methods are based on neural networks to build implicit representations, our results support the claim that the ENM architecture can reduce computation costs for real-time applications. We test all approaches on a desktop computer with a 3.7 GHz CPU and 64 GB memory, and a NVIDIA Quadro RTX 5000 GPU with 16 GB memory.

As shown in Tab. II, the runtime performance of the tested methods demonstrates the efficiency of our approach. We keep the default settings of the number of particle to each method for fair comparison. Specifically, our ENM-MCL outperforms the other methods in both initialization and pose tracking phases on the dataset. During pose tracking, ENM-MCL achieves 180.2 fps using 1,000 particles, making it suitable for real-time operation. Although our method requires more particles during the initialization phase (80,000 particles), it still reaches a speed of 4 Hz, showcasing its efficiency even with larger particle sets.

D. Convergence Analysis

Finally, we analyze our method with respect to its ability for rapid and reliable global localization. For fair comparison, we compare the convergence speed of localization with baseline methods in sequence 1 of the dataset, as it is a relatively simple scenario where all baselines succeed, and the trajectory of the sequence is shown in Fig. 4.

The location RMSE error curves of all methods are shown in Fig. 5. The occupancy grid map-based methods converge slowly in the sequence because of many similar offices in the environment and the occupancy grid map does not include enough geometric details to quickly distinguish similar rooms. In contrast, the implicit representation-based methods quickly converge and our ENM-MCL maintains a fast convergence time of only 3.6 seconds.

Method	Localization Speed [FPS]		
	Initialization (**#Particles**)	Pose Tracking (**#Particles**)	Average Speed on Sequence 1
IRMCL	1.2 Hz (100,000)	27.0 Hz (5,000)	9.1 Hz
LocNDF	0.5 Hz (100,000)	2.8 Hz (10,000)	2.1 Hz
ENM-MCL	**4.0 Hz (80,000)**	**250.0 Hz (1,000)**	**180.2 Hz**

TABLE II: Runtime comparison of different methods. We report the frame rate for the initialization and pose tracking phase of MCL under the default parameters of baselines. We also report average frame rate of both phases over the complete sequence.

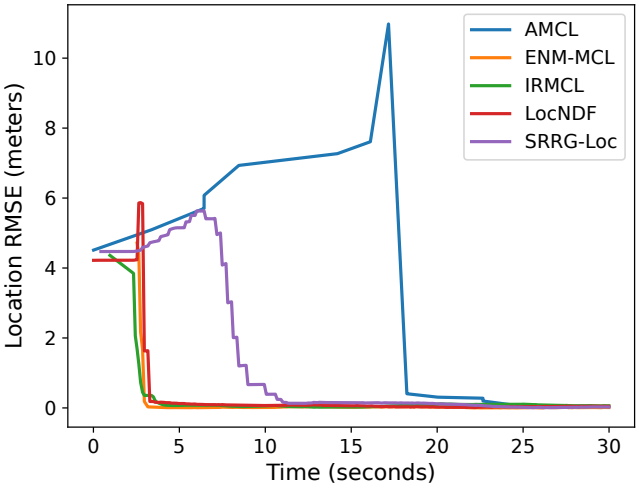

Fig. 5: The error curves of location RMSE on the Sequence 5 of the in-house dataset.

In summary, our evaluation suggests that our method is highly efficient and robust, which is suitable for efficient global localization. At the same time, our method maintains high accuracy and reliability. Thus, we supported all our claims with this experimental evaluation.

V. Conclusion

In this paper, we presented a novel approach to robot localization using an efficient and effective implicit neural representation. Our map representation is capable of learning both, the non-projective signed distance fields and direction-aware projective distance fields from 2D LiDAR data, which stores both positional and directional geometric features of the environment. Our method exploits a learnable dense feature grid combined with a light-weight neural network as the map representation model. This allows us to successfully integrate the implicit representation into a Monte Carlo localization framework to improve the accuracy and efficiency for implicit representation-based MCL. We evaluated our approach on a public dataset and provided comparisons to other existing methods and supported all claims made in this paper. The experiments suggest that by leveraging neural representations, we can not only improve the quality of map, but also reduce the computation costs of MCL to perform real-time pose tracking and enable efficient global localization with a large number of particles.

REFERENCES

[1] J. Barron, B. Mildenhall, D. Verbin, P.P. Srinivasan, and P. Hedman. Mip-nerf 360: Unbounded anti-aliased neural radiance fields. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[2] M. Bennewitz, C. Stachniss, W. Burgard, and S. Behnke. Metric Localization with Scale-Invariant Visual Features using a Single Perspective Camera. In *Proc. of the European Robotics Symposium*, 2006.

[3] X. Chen, T. Läbe, L. Nardi, J. Behley, and C. Stachniss. Learning an Overlap-based Observation Model for 3D LiDAR Localization. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2020.

[4] X. Chen, I. Vizzo, T. Läbe, J. Behley, and C. Stachniss. Range Image-based LiDAR Localization for Autonomous Vehicles. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2021.

[5] F. Dellaert, D. Fox, W. Burgard, and S. Thrun. Monte carlo localization for mobile robots. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 1999.

[6] D. Droeschel and S. Behnke. Efficient Continuous-Time SLAM for 3D Lidar-Based Online Mapping. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2018.

[7] D. Fox, W. Burgard, F. Dellaert, and S. Thrun. Monte Carlo Localization: Efficient Position Estimation for Mobile Robots. In *Proc. of the National Conf. on Artificial Intelligence (AAAI)*, 1999.

[8] D. Fox. KLD-sampling: Adaptive particle filters. In *Proc. of the Conf. Neural Information Processing Systems (NIPS)*, 2001.

[9] G. Grisetti. srrg-localizer2d (1.6.0). `https://gitlab.com/srrg-software/srrg_localizer2d`, 2018.

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

[11] H. Kuang, X. Chen, T. Guadagnino, N. Zimmerman, J. Behley, and C. Stachniss. IR-MCL: Implicit Representation-Based Online Global Localization. *IEEE Robotics and Automation Letters (RA-L)*, 8(3):1627–1634, 2023.

[12] J. Liu, Q. Nie, Y. Liu, and C. Wang. Nerf-loc: Visual localization with conditional neural radiance field. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2023.

[13] W. Lu, Y. Zhou, G. Wan, S. Hou, and S. Song. L3-Net: Towards Learning Based LiDAR Localization for Autonomous Driving. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[14] D. Maggio, M. Abate, J. Shi, C. Mario, and L. Carlone. Loc-nerf: Monte carlo localization using neural radiance fields. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2023.

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

[16] S. O'Callaghan and F. Ramos. Gaussian Process Occupancy Maps. *Intl. Journal of Robotics Research (IJRR)*, 31(1):42–62, 2012.

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

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

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

[20] F. Ramos and L. Ott. Hilbert maps: Scalable continuous occupancy mapping with stochastic gradient descent. *Intl. Journal of Robotics Research (IJRR)*, 35(14):1717–1730, 2016.

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

[22] C. Stachniss and W. Burgard. Mobile Robot Mapping and Localization in Non-Static Environments. In *Proc. of the National Conf. on Artificial Intelligence (AAAI)*, 2005.

[23] S. Thrun, W. Burgard, and D. Fox. *Probabilistic Robotics*. MIT Press, 2005.

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

[25] Y. Yuan and A. Nüchter. Uni-fusion: Universal continuous mapping. *IEEE Trans. on Robotics (TRO)*, 40:1373–1392, 2024.

[26] Y. Yuan, H. Kuang, and S. Schwertfeger. Fast gaussian process occupancy maps. In *Proc. of the Intl. Conf. on Control, Automation, Robotics and Vision (ICARCV)*, 2018.

[27] S. Zhi, T. Laidlow, S. Leutenegger, and A. Davison. In-place scene labelling and understanding with implicit scene representation. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2021.

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

[29] 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. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[30] N. Zimmerman, T. Guadagnino, X. Chen, J. Behley, and C. Stachniss. Long-Term Localization using Semantic Cues in Floor Plan Maps. *IEEE Robotics and Automation Letters (RA-L)*, 8(1):176–183, 2023.

[31] N. Zimmerman, L. Wiesmann, T. Guadagnino, T. Läbe, J. Behley, and C. Stachniss. Robust Onboard Localization in Changing Environments Exploiting Text Spotting. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2022.

CERTIFICATE OF REPRODUCIBILITY

The authors of this publication declare that:

1) The software related to this publication is distributed in the hope that it will be useful, support open research, and simplify the reproducability of the results but it comes without any warranty and without even the implied warranty of merchantability or fitness for a particular purpose.

2) *Haofei Kuang* primarily developed the implementation related to this paper. This was done on Ubuntu 22.04.

3) *Yue Pan* verified that the code can be executed on a machine that follows the software specification given in the Git repository available at:

```
https://github.com/PRBonn/enm-mcl
```

4) *Yue Pan* verified that the experimental results presented in this publication can be reproduced using the implementation used at submission, which is labeled with a tag in the Git repository and can be retrieved using the command:

```
git checkout icra25-release
```