LocNDF: Neural Distance Field Mapping for Robot Localization

Louis Wiesmann Tiziano Guadagnino Ignacio Vizzo Nicky Zimmerman
Yue Pan Haofei Kuang Jens Behley Cyrill Stachniss

***Abstract*—Mapping an environment is essential for several robotic tasks, particularly for localization. In this paper, we address the problem of mapping the environment using LiDAR point clouds with the goal to obtain a map representation that is well suited for robot localization. To this end, we utilize a neural network to learn a discretization-free distance field of a given scene for localization. In contrast to prior approaches, we directly work on the sensor data and do not assume a perfect model of the environment or rely on normals. Inspired by the recently proposed NeRF representations, we supervise the network by points sampled along the measured beams, and our loss is designed to learn a valid distance field. Additionally, we show how to perform scan registration and global localization directly within the neural distance field. We illustrate the capabilities to globally localize within an indoor environment utilizing a particle filter as well as to perform scan registration by tracking the pose of a car based on matching LiDAR scans to the neural distance field.**

***Index Terms*—Localization, Mapping, Deep Learning Methods**

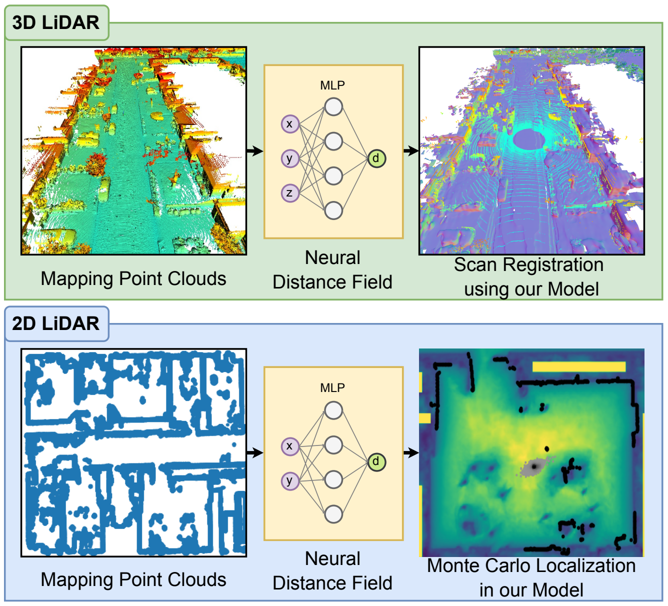


Fig. 1: In this paper, we will show how to learn a neural distance field from point cloud data. Using this neural distance field as a map representation enables scan registration and localization.

I. Introduction

MAPPING the environment is a crucial building block for many robotic applications, each with different requirements and needs to the map. Some tasks like reconstruction try to replicate the scene as accurate as possible. For other tasks like localization, pose tracking, or path planning the map is just a means to an end: The map is only as good, as it is useful to solve the actual task. At the same time, simultaneous localization and mapping (SLAM) reqiures to update the map incrementally and has to operate in real-time. Here, for many on board or on-demand applications the memory footprint can be a limiting factor.

We address the problem of representing the environment using point clouds from sensor data to improve the localization performance of a robot in the proposed map representation. Common map representations in mobile robotics are occupancy maps [18], surfels [2], [35], distance fields [14], [30], NDTs [34], or raw point clouds [40], [42]. These representations are often combined with data structures for efficient management: octrees, kD-trees, grid maps, or hash maps just to name a few. In this work, we will take a closer look at representing the scene with a distance field. The advantage of this representation for mobile robotics is that the distance it provides can be used for common tasks like scan registration, Monte Carlo localization (MCL), path planning, and even for reconstruction.

Recent works in computer vision train neural networks to learn the distance field for a given scene. This so-called neural distance field (NDF) is usually modeled by a simple multi-layer perceptron, which returns for each point in space the distance to the closest surface. The advantage over the common grid-based distance fields is the continuity of the network, therefore, it is not limited by a grid resolution. Additionally, a network has the ability to represent low-dimensional information efficiently, whereas grid-based representations spend a lot of cells to store low-dimensional objects like walls or a ground plane. In computer vision, the training of the NDFs is usually done by either the usage of high-resolution ground truth meshes, normal information, or is directly supervised by a given distance field. So far, it has been rarely investigated how the NDF can be trained from raw sensor data, such as

Manuscript received: March 2, 2023; Revised: May 25, 2023; Accepted: June 14, 2023. This paper was recommended for publication by Editor Javier Civera upon evaluation of the Associate Editor and Reviewers' comments.

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 European Union's Horizon 2020 research and innovation programme under grant agreement No 101017008 (Harmony). All authors are with the 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.

Digital Object Identifier (DOI): see top of this page.

raw LiDAR data. Additionally, it is not clear how well those NDFs are suited to tackle mobile robotic tasks.

In this work, we will learn neural distance fields (NDFs) based on point cloud data as commonly obtained from real LiDAR sensors. Additionally, we will show how to localize in a given NDF, as visualized in Fig. 1. In the domain of autonomous driving, we look at pose tracking for odometry estimation in a given map. For indoor environments, we will look at global localization using MCL. In sum, we make three key claims: (1) Our proposed loss function enables us to learn a neural distance field, which (2) is well suited for scan registration and (3) allows for localization within MCL. We plan to release the code, as well as the pre-trained models at https://github.com/PRBonn/LocNDF

II. Related Work

1) Map Representations: Many different map representations have been used in the robotic context. The point cloud representation is quite common since it can represent directly the sensor observations. Acceleration structures like octrees [3], [24], hash maps [27], [40], or kD-trees [4] can speed up accessing certain points and can help to reduce redundancy in the data. Different ways to explicitly represent the surface are to use meshes [38], or surfels [2], [35]. Occupancy fields have the notion of free and occupied space and can also be extended to represent unknown space in a probabilistic fashion [18]. These occupancy fields are often stored in image-like grids for 2D [15] and classically in octrees for 3D [18]. neural radiance fields (NeRF) [25] use most commonly an MLP to learn the occupancy of a scene. Instead of storing the occupancy of a scene, another common representation is to store for each position the distance to the closest surface [9]. A Euclidean distance transform can be used to transform an occupancy grid [30] into a distance field. Traditionally, the distance values are stored in a grid. Nowadays, more learning-based approaches emerge, which use neural networks to learn the so-called neural distance fields. Such NDFs are usually supervised either by density fields [37], normals [33], [43], or directly the distance field [8], [28]. Learning them directly from sensor observations is just rarely exploited [1], [45]. Instead of learning the whole scene by one MLP, some approaches learn local MLPs or embedding vectors [29], [45]. Our goal is to learn an NDF directly from sensor observations, where we use a simple compact MLP as a representation.

2) Scan Registration: Scan registration is a common problem in robotics and the most common method is the iterative closest point (ICP) algorithm [5]. It usually, consists of two steps: first finding correspondences between the two point clouds, which can then be used to estimate the pose by optimizing an error metric. For finding correspondences, one can search for the closest point [5], use projective data associations [2], [26], or do feature-based matching [17], [41]. Point-to-point [5], point-to-plane [7], or even plane-to-plane [31], [32] metrics are minimized for computing an alignment. Plane-based metrics are often based on point normals or triangle meshes. Robust kernels and correspondence thresholds are used to reduce the impact of outliers, low overlap, and dynamic objects [13], [19]. We are directly registering the point clouds in the NDF, which leads to a similar optimization as for point-to-plane ICP.

3) Monte Carlo localization: MCL is a method to localize a mobile robot in a given map using a particle filter [10], [11]. Its key idea is to represent the posterior belief about the robot's pose by a set of weighted samples, so-called particles. Each particle weight represents the likelihood of the corresponding pose hypothesis, and we can compute it through sensor observations. The basic idea is to compare the current sensor reading at the particle location with the map, using the so-called observation model [36].

In the context of 2D global localization, the map is often represented as an occupancy grid [10] or a floor plan map [6], [46]. Although effective, the usage of such maps limits the accuracy of the localization to the grid resolution. Kuang et al. [20] propose a neural occupancy field as a map for MCL to overcome this limitation and improve localization accuracy. However, their approach requires computationally demanding ray casting-based rendering to evaluate the observation model and thus has to reduce the usable number of beams per scan for online localization. Conversely, using our NDF, we predict the distances to the closest surface directly and use it in the observation model without relying on compute-heavy rendering. In the image domain, NeRFs have been used to localize [23].

III. Learning Neural Distance Fields for Robot Localization

In this paper, we aim at learning a neural distance field from point cloud data, acquired by sensors, like LiDAR sensors, or RGB-D cameras as a representation to explicitly support localization. We do not rely on point cloud normals, since normal estimation heavily relies on the type of sensor and is prone to errors. The above-mentioned range sensors have in common that they measure the distance from the sensor origin to the surface. The only assumption we make is that the space in between is free space.

In the following, we will first explain how we train the NDF from the sensor data. Second, we show two common localization methods for point cloud data within the NDF, namely scan registration using ICP and global localization using a particle filter.

A. Learning NDF from Sensor Data

Our goal is to be able to query the neural distance field D at an arbitrary point in space $\boldsymbol{p} \in \mathbb{R}^{\mathcal{D}}$ to obtain the distance d to the closest surface. In this work, we focus on applications in outdoor robotics with $\mathcal{D} = 3$, which covers localization and registration using 3D point clouds produced by commonly employed automotive LiDAR sensors or terrestrial laser scanners, but also indoor environments, where we often use $\mathcal{D} = 2$, for commonly equipped 2D LiDARs for localization and navigation.

The representation of our map is a multi-layer perceptron $D : \mathbb{R}^{\mathcal{D}} \mapsto \mathbb{R}$ which maps the input coordinates to

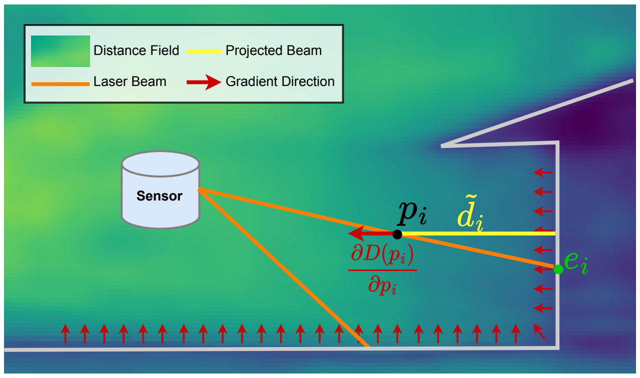


Fig. 2: Instead of learning the distance between the point $\boldsymbol{p}_i$ and the measured point on the surface $\boldsymbol{e}_i$ (red), e.g., as for TSDF, we project the beam along the direction of the gradient (grey arrow) to supervise by the approximated distance $\tilde{d}_j$ to the closest surface. The color of the background corresponds to the distance field; the brighter the color, the higher the distance.

the Euclidean distance space. We use a positional encoding $\pi : \mathbb{R}^{\mathcal{D}} \mapsto \mathbb{R}^{2I_\omega}$ with periodic activation functions,

$$\pi(\boldsymbol{p}) = (\boldsymbol{p}, \sin(\omega_1 \boldsymbol{p}), \cos(\omega_1 \boldsymbol{p}), \ldots, \sin(\omega_{I_\omega} \boldsymbol{p}), \cos(\omega_{I_\omega} \boldsymbol{p})), \quad (1)$$

which is applied separately to each coordinate of the point $\boldsymbol{p}$ with the aim to retain high-frequency information in the distance field [25].

In contrast to previous approaches which supervised the training process either by ground truth distances [8], [28], occupancy fields [37], or given normals [33], [43], we exploit the measurement process of the LiDAR sensors similar to truncated signed distance field (TSDF) fusion pipelines [26], [39]. However, we do not directly supervise by the TSDF values like Zhong et al. [45], but rather by an approximated distance as explained as follows.

Laser sensors measure the distance from the sensor origin to the surface, which we will call the ray distance d_p. Inspired by NeRFs, we sample points $\{\boldsymbol{p}_i \in \mathbb{R}^D \mid i = 1, \ldots, N_i\}$ along the LiDAR beam, i.e., $\boldsymbol{p}_i = (1 - \lambda_i)\boldsymbol{o}_i + \lambda_i \boldsymbol{e}_i$, between the sensor origin $\boldsymbol{o}_i \in \mathbb{R}^{\mathcal{D}}$ and the end of the beam $\boldsymbol{e}_i \in \mathbb{R}^{\mathcal{D}}$. We sample more points near the surface by sampling log-linearly along the ray, i.e.,

$$\lambda_i = \frac{1 - 10^{\frac{i}{N-1} - 1}}{0.9}. \quad (2)$$

Consequently, the ray distance d_i for each sampled point to the surface is given by $d_i = \|\boldsymbol{e}_i - \boldsymbol{p}_i\|$. Note that the ray distance d_i does not necessarily correspond to the distance to the closest surface. Instead of searching for each sampled point the closest point on a surface, which is computationally expensive for large-scale maps and requires determining first the surface by reconstruction, we can approximate the direction $\boldsymbol{n}_i$ to the closest surface analytically using the NDF D by using the gradient:

$$\boldsymbol{n}_i = -\frac{\partial D(\boldsymbol{p}_i)}{\partial \boldsymbol{p}_i}. \quad (3)$$

A visualization of this process is depicted in Fig. 2. The gradient provides us with the direction of the steepest increase of the distance field, therefore the negative gradient points toward the closest surface. In practice, we can use automatic differentiation to compute $\boldsymbol{n}_i$. We project the distance d_i along the direction $\boldsymbol{n}_i$ to approximate the distance to the closest surface

$$\tilde{d}_i = \frac{(\boldsymbol{e}_i - \boldsymbol{p}_i)^\top \boldsymbol{n}_i}{\|\boldsymbol{n}_i\|}. \quad (4)$$

We use the approximated distance $\tilde{d}_i$ to supervise the training. Note that this is a circular problem: The better the approximated distance $\tilde{d}_i$, the better we can supervise the NDF. At the same time, the better the NDF, the better we can estimate the direction to the surface, which finally should result in a better-approximated distance. This circular dependency might raise the question if the training is stable, especially when we initialize the NDF D with random weights. Practically, we did not notice any instabilities due to this approximation in the training. We reason that this might be due to the fact that the approximation error, $\epsilon_i = |\tilde{d}_i - d_i|$, is smaller, the closer the query point $\boldsymbol{p}_i$ is to the surface. Thus, for surface points or points close to the surface, i.e., $d_i \approx 0$, we approach $\epsilon_i \approx 0$ regardless of the gradient $\boldsymbol{n}_i$. Therefore, the correct distance propagates from the surface to the free space while training the NDF.

Similar to TSDF fusion pipelines [39], we prioritize measurements $\boldsymbol{e}_i$ with lower distance d_i. For this, we introduce a weight w_i given by

$$w_i = (d_{max} - d_i)^\gamma, \quad (5)$$

where d_{max} is the largest distance in a batch and γ is a hyperparameter which regulates the impact of measurements from higher distances, i.e., the higher γ, the lower the impact of far points. We supervise the NDF by minimizing the weighted L1 loss of our approximated distances

$$\mathcal{L}_{\text{dist}} = \sum_i \frac{w_i |D(\pi(\boldsymbol{p}_i)) - \tilde{d}_i|}{\sum_j w_j}. \quad (6)$$

Additionally, we have an additional loss to enforce that the endpoints lie on the surface

$$\mathcal{L}_{\text{end}} = \sum_i |D(\boldsymbol{e}_i)|. \quad (7)$$

Similar to Zhong et al. [45], we add a regularization loss to enforce the Eikonal equation $\|\nabla N\| = 1$, which needs to hold for being a valid distance field

$$\mathcal{L}_{\text{Eik}} = \sum_i |\|\boldsymbol{n}_i\|_2 - 1|, \quad (8)$$

as well as a loss to enforce that neighboring points have similar normals

$$\mathcal{L}_{\text{n}} = \sum_i |\angle(\boldsymbol{n}_i, \hat{\boldsymbol{n}}_i)|, \quad (9)$$

where $\angle(\cdot)$ is the cosine distance and $\hat{\boldsymbol{n}}_j$ is the gradient of the neighbor of $\boldsymbol{p}_j$. The neighbors $\boldsymbol{p}_j$ are sampled within a distance τ of $\boldsymbol{p}_i$, i.e., we randomly select from all radius neighbors $\{\boldsymbol{p} \mid ||\boldsymbol{p}_i - \boldsymbol{p}|| < \tau\}$ an arbitrary point $\boldsymbol{p}_j$.

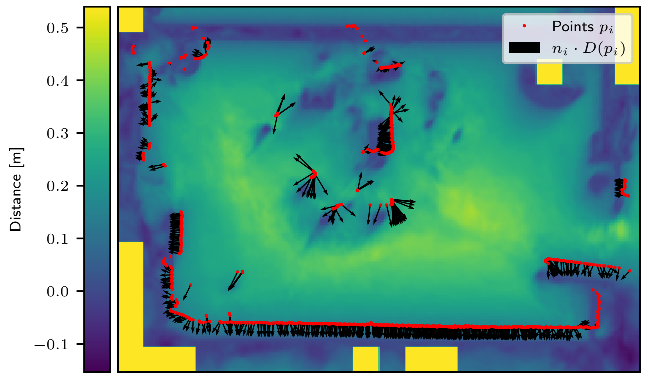


Fig. 3: Principle of scan registration in an NDF. Querying the NDF at the positions of the input scan provides us with the distance and by differentiation also with the direction we have to go, to align the scan to the NDF. This procedure can be solved iteratively in an ICP fashion. The color of the background corresponds to the distance field; the brighter the color, the higher the distance.

The final loss is a linear combination of the aforementioned losses

$$\mathcal{L} = \mathcal{L}_{\text{dist}} + \alpha_{\text{end}}\mathcal{L}_{\text{end}} + \alpha_{\text{Eik}}\mathcal{L}_{\text{Eik}} + \alpha_{\text{n}}\mathcal{L}_{\text{n}}. \tag{10}$$

B. Scan Registration using a NDF

In this section, we show how to leverage the learned NDF to register point clouds to the map using ICP effectively. The objective is to find the rotation $\boldsymbol{R}$ and transformation $\boldsymbol{t}$ that aligns the point cloud P to the NDF map D, i.e., that reduces the distance between the point cloud and the surface

$$\boldsymbol{R}^*, \boldsymbol{t}^* = \underset{\boldsymbol{R},\boldsymbol{t}}{\operatorname{argmin}} \sum_{\boldsymbol{p}_i \in P} D(\boldsymbol{R}\boldsymbol{p}_i + \boldsymbol{t})^2. \tag{11}$$

We solve the problem using non-linear least squares optimization, where the Jacobians for i$^{\text{th}}$ point are

$$\boldsymbol{J}_i = \left[\frac{\partial D(\boldsymbol{R}\boldsymbol{p}_i + \boldsymbol{t})}{\partial \boldsymbol{t}}, \frac{\partial D(\boldsymbol{R}\boldsymbol{p}_i + \boldsymbol{t})}{\partial \boldsymbol{\Theta}}\right] = \left[\boldsymbol{n}_i, \boldsymbol{p}_i \times \boldsymbol{n}_i\right], \tag{12}$$

where $\boldsymbol{\Theta}$ is the axis-angle parameterization of $\boldsymbol{R}$.

As we can see in Eq. (11) and Eq. (12), we do not rely on corresponding points. This has the advantage of not needing to search for correspondences in contrast to classical ICP-based methods. We can solely solve the problem by knowing in which direction ($\boldsymbol{n}$) and how far (given by $D(\boldsymbol{p})$) we have to go. This information is directly encoded in the weights of the network and learned in the generation of the map. A visualization of this for the 2D case can be seen in Fig. 3.

We know from theory [16] that the distance field needs to fulfill the Eikonal equation $\|\nabla D\| = 1$, or in simpler words: if we move one meter away from the surface, the distance needs to increase by one meter. Since we only approximate the NDF by a neural network, this does not necessarily hold. If the norm of the gradient is either larger or smaller, we would over or underestimate the distance accordingly. To counter this phenomenon, we normalize the distance by the norm of the gradient $D(\boldsymbol{p}_i)/\|\boldsymbol{n}_i\|$. If for example the gradient would be $\|\nabla D\| > 1$, one would overestimate the distance and overshoot, ending up behind and not at the surface. Therefore, by shortening the step to only move $D(\boldsymbol{p}_i)/\|\boldsymbol{n}_i\|$ towards the surface, one would end up on, or at least closer to the surface, which enables a more reliable registration.

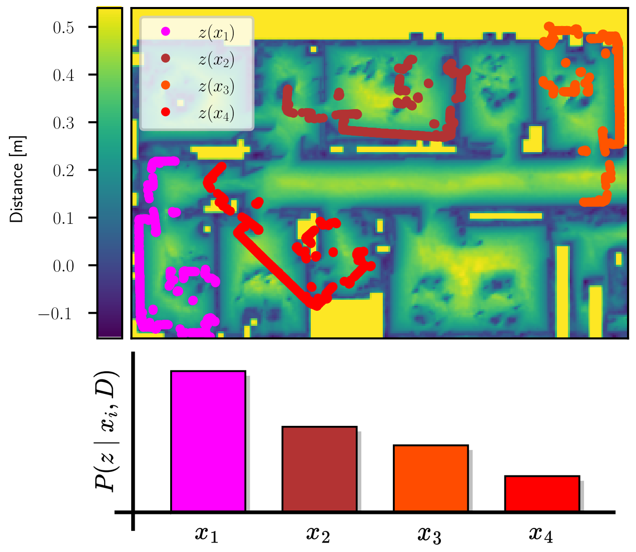


Fig. 4: The neural distance field can be used in an particle filter to evaluate the likelihood of a measurement for each particle in the given NDF. The better the scans are aligned with the zero level of the NDF, the higher the likelihood.

C. MCL-based Localization using a NDF

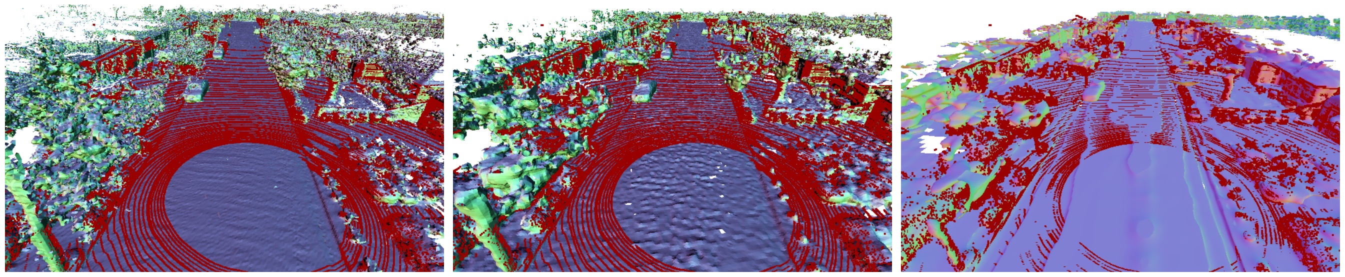
In this section, we explain how to globally localize within an NDF using Monte Carlo localization. The belief $bel(\boldsymbol{x}_t)$ about the robots position $\boldsymbol{x}_t$, at time t is represented by as set of particles $\{(\boldsymbol{x}_t^i, w_t^i) \mid i = 1...I\}$ each with a corresponding weight w_t^i. A motion model $p(\boldsymbol{x}_t) \sim p(\boldsymbol{x}_t \mid \boldsymbol{x}_{t-1}, \boldsymbol{u}_t)$ updates the particles based on their previous position $\boldsymbol{x}_{t-1}$ and the control commands $\boldsymbol{u}_t$. The weight of the particles is updated by the observation model $w_i^t \propto p(\boldsymbol{z}_t \mid \boldsymbol{x}_t, D)$ which depends on the observations $\boldsymbol{z}_t$, the pose $\boldsymbol{x}_t$, and the NDF D. Assuming we observe the local surrounding with a LiDAR sensor, we can evaluate the distance field at the observed point cloud $\{{}^j\boldsymbol{z}_t^i \mid j = 1, ..., J\}$ around the particle position ${}^j\boldsymbol{x}_t^i$ with the classical beam-end model

$${}^j d_t^i = D\big(\boldsymbol{R}_t^i\, {}^j\boldsymbol{z}_t^i + \boldsymbol{t}_t^i\big). \tag{13}$$

$\boldsymbol{R}_t^i$ and $\boldsymbol{t}_t^i$ are the rotation matrix and translation of the particle's position $\boldsymbol{x}_t^i$ respectively. Assuming a Gaussian noise model, we can compute the weight for a particle by

$$p(\boldsymbol{z}_t \mid \boldsymbol{x}_t^i, D) \propto w_t^i = \exp\left(-\frac{\beta}{J}\sum_{j=1}^{J} {}^j d_t^i\right) + \omega. \tag{14}$$

The parameters β and ω are commonly used for robustness against outliers. In other words, the lower the average distance between the observations and the surface, the higher the weight (see Fig. 4). The particles are resampled after each observation step based on their weight w_t^i to focus on the most likely positions.



(a) VDBFusion [39] (b) SHINE mapping [45] (c) LocNDF (Ours)

Fig. 5: Qualitative results of registering a scan (red points) to the maps generated by different distance field-based methods. The meshes are generated using the marching cubes algorithm. We use the mesh only for visualization purposes, the registration is done directly on the distance field.

IV. Experimental Evaluation

In this section, we evaluate the localization performance to validate the three key claims that (1) the proposed training strategy can provide neural distance fields which (2) are well suited for scan registration as well as (3) global localization using MCL. To evaluate the scan registration performance (2), we will track the pose of a car in the given maps using ICP. For the global localization (3), we evaluate in a 2D indoor dataset of our office environment. Eventually, we provide our ablation studies to validate our poposed training methodology (1).

A. Training Setup

In this section, we provide the hyperparameters used to conduct the experiments, which, unless stated differently, are used throughout all experiments. We transform the coordinates of the point clouds to be in the range of [0,1] before passing them into the positional encoding. For the positional encoding, we sample $I_\omega = 30$ different frequencies. The default network uses a SIREN [33] backbone with a hidden feature dimension of size 128. For the training, we sample $N_i = 40$ points between the sensor origin and the endpoint to supervise the distance field (Eq. (5)) and additionally 20 pairs of points with a distance up to 10 cm, which are distributed randomly in space on which we are computing the regularization on the normals (Eq. (8) and Eq. (9)). The coefficients between the different loss terms are $\alpha_{end} = 10^{-1}$, $\alpha_{Eik} = 10^{-4}$, and $\alpha_n = 10^{-3}$, as well as $\gamma = 3$. We optimize using AdamW with a start learning rate of 10^{-4}, which gets decreased with a cosine annealing scheduler to 10^{-7} over around 25,000 steps. The experiments have been evaluated on a desktop PC with i7 @ 3.5 GHz$\times$8 CPU and an Nvidia RTX A5000.

B. 3D Pose Tracking in Outdoor Scenes

For the 3D localization, we want to estimate the current vehicle pose by aligning local LiDAR point clouds with the map. We assume a rough initial location to be given, which is usually provided by a low-cost GPS sensor and track the vehicle's position using scan registration. The initial guess for ICP of the first timestamp is provided by a rough GPS position, whereas, for the following scans, we use a constant velocity model as the initial guess for ICP. We evaluate the registration performance on the Apollo-Southbay [22] dataset, which has multiple runs through the same areas recorded at different points in time. Since the environments changed substantially between the different points in time, points cannot always be matched, and therefore a robust kernel, here a Geman McLure kernel with the parameter $k = 0.3$ m, is used.

TABLE I: Scan Registration Results

Approach	MAE($\boldsymbol{t}$) [m]	Memory [MB]	Runtime [s]
VDBFusion (KDTree)	0.072	170.8	0.87
VDBFusion (Projective)	0.072	170.8	1.52
LOAM (KDTree)	0.0714	6.3	0.23
SHINE (KDTree)	0.070	154.1	0.83
SuMa (Projective)	0.085	240.7	0.03
Ours	**0.059**	5.1	0.42

For the map generation, we use the first 800 scans of the ColumbiaPark-3 mapping run and the provided poses, which got obtained using a combination of GPS, IMU, and a SLAM system. Instead of training one big network for the whole scene, we found it beneficial to follow the key pose paradigm. For this, we use bounding boxes of 50 m size and 20 % overlap along the trajectory. We assign to each bounding box an NDF and train them incrementally based on the weights of the previous key pose. We fine-tune the maps for 10 epochs with the standard parameters. The evaluation will be done on 700 scans in the same area (starting at scan 5280) from the test run. Hyperparameters are tuned on 800 scans of the training set (starting at scan 6880). We compare against VDBFusion [39] as a highly effective, traditional TSDF fusion pipleline, as well as against the recent learning-based SHINE mapping system [45], that utilizes also neural distance fields for mapping. For these baselines, we do the registration based on point-to-plane ICP using the same robust kernel using their meshes. Additionaly, we compare against the surfel-based method SuMa [2] and the grid based method LOAM [44], both in localization mode, i.e., first constructing the map on the mapping run, and use the second run from a different point in time solely for registration without updating the map. For the evaluation, we use the mean average translation error MAE($\boldsymbol{t}$), the memory consumption of the maps, as well as the average runtime for aliging a scan. The baselines either use a kD-tree or projective data associations to find correspondences.

The results are presented in Tab. I. Our approach is able to outperform the baselines in respect of mean average translation error MAE($\boldsymbol{t}$) while requiring also the least memory. All the approaches have a mean average rotation error lower than 0.1°. In Fig. 5, the registration of a scan w.r.t. the map representations are depicted. Note that we use the mesh obtained using marching cubes [21] only for visualization and not for the registration. We can see that the points are clipping inside our triangle mesh (Fig. 5c), showing that the point cloud is well aligned. The meshes from the other approaches are more detailed, but the aligned point clouds seem like floating in the scene. This is due to the acquisition process of the TSDF, where for each voxel a weighted average over the ray distances is stored. This leads to an overestimation of the distance to the closest surface, resulting in slightly smaller objects and the aligned point cloud seems to be always in front of the surface. This effect is mitigated for our approach, due to projecting the ray distance along the surface normal and having a special loss term given in Eq. (7) to enforce that the measured surface actually is the zero level of the distance field. SHINE mapping [45] is also supervising using the ray distance leading to similar effects as for TSDF.

C. 2D Monte Carlo Localization

In this experiment, we evaluate the global localization performance using the MCL in our office environment. The robot is equipped with a 2D laser scanner (Hokuyo UTM-30LX) and wheel odometry. We use directly the measurements $\boldsymbol{u}_t = (\Delta x, \Delta y, \Delta\theta)$ of the wheel odometry as a motion model with a Gaussian noise model of

$$\boldsymbol{\Sigma}_{\boldsymbol{u}_t} = \begin{bmatrix} 5 \cdot 10^{-1} & 2.5 \cdot 10^{-2} & 1 \cdot 10^{-2} \\ 2.5 \cdot 10^{-2} & 5 \cdot 10^{-3} & 5 \cdot 10^{-3} \\ 5 \cdot 10^{-2} & 5 \cdot 10^{-4} & 2.5 \end{bmatrix}. \tag{15}$$

We use the same sequences for mapping and evaluation used by Kuang et al. [20] as well as the same setting using 100,000 particles for initialization for a fair comparison. We assume the particle filter to be converged when the standard deviation of the particle's position is below 30 cm to switch into the pose tracking mode with 10,000 particles. We reweight and resample the particles if the robot moved by at least 5 cm or 0.1 rad. The hyperparameters for the observation model are set to $\beta = 100$ and $\omega = 10^{-8}$.

For constructing the map, we train for 15 epochs on the 31,608 training scans with the default parameters for our network. The NDF has no notion about unknown space by itself, i.e., areas that did not get supervised. Therefore, we also store a low-resolution bitmap of size $[100 \times 100]$ voxels to know roughly, which spaces are not supervised. Points that lie in an unknown area have the maximum distance assigned, rather than querying the network. We use the standard root mean squared error metric (RMSE) between the ground truth and estimated positions. The RMSE is evaluated for converged positions at certain thresholds (5 cm, 10 cm, 20 cm). Additionally, we provide the percentage of poses below the given thresholds. The results are averaged over 5 runs where the RMSE is only reported if all runs at least converged once, otherwise denoted as "-". We compare against the standard ROS1 localizer AMCL [12], the MCL system by Grisetti [14], as well as the recent, learning-based IR-MCL [20] approach.

TABLE II: MCL Results

	Approach	RMSE(t) @ 5cm	RMSE(t) @ 10cm	RMSE(t) @ 20cm
Seq 1	AMCL	- (0.0%)	- (0.0%)	- (0.0%)
	SRRG	0.034 (57.1%)	0.047 (88.6%)	0.049 (90.3%)
	IR-MCL	0.033 (60.3%)	0.047 (92.5%)	0.052 (95.7%)
	Ours	**0.031** (**76.6%**)	**0.041** (**96.2%**)	**0.047** (**99.2%**)
Seq 2	AMCL	0.037 (26.5%)	0.061 (56.2%)	0.089 (80.6%)
	SRRG	0.034 (41.4%)	0.059 (**87.4%**)	0.063 (92.6%)
	IR-MCL	**0.029** (60.1%)	0.048 (**87.4%**)	0.054 (**93.8%**)
	Ours	**0.028** (**78.0%**)	**0.032** (83.0%)	**0.042** (86.7%)
Seq 3	AMCL	0.038 (20.0%)	0.066 (58.7%)	0.099 (81.3%)
	SRRG	0.033 (36.5%)	0.050 (59.0%)	0.075 (71.0%)
	IR-MCL	0.033 (**68.2%**)	0.043 (**84.4%**)	0.054 (**89.8%**)
	Ours	**0.028** (54.4%)	**0.034** (62.0%)	**0.053** (70.0%)
Seq 4	AMCL	0.034 (**63.1%**)	0.048 (**88.7%**)	0.059 (**98.8%**)
	SRRG	0.035 (54.1%)	0.052 (83.7%)	**0.058** (88.2%)
	IR-MCL	0.033 (33.7%)	0.054 (59.9%)	0.091 (85.2%)
	Ours	**0.031** (42.4%)	**0.046** (64.5%)	0.069 (73.7%)
Seq 5	AMCL	- (0.0%)	- (0.0%)	- (0.0%)
	SRRG	0.035 (**48.8%**)	0.051 (**86.8%**)	**0.055** (89.0%)
	IR-MCL	0.032 (41.4%)	0.057 (81.3%)	0.064 (**89.8%**)
	Ours	**0.027** (41.5%)	**0.032** (45.0%)	0.071 (55.6%)

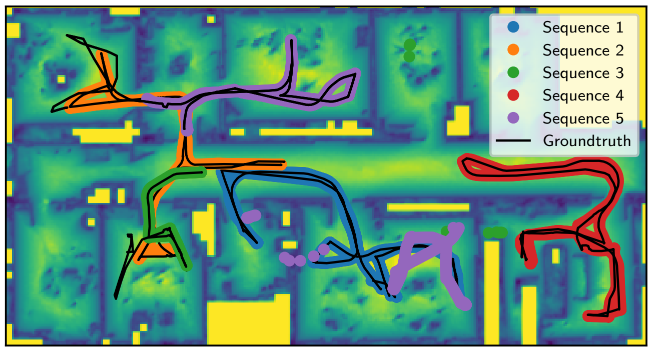


Fig. 6: Qualitative localization results of our approach on the five sequences. The estimated poses are mostly aligned with the groundtruth. In sequence 5 our approach converged to the wrong location but could recover later to the correct position. Sequence 4 took a long time to resolve ambiguities to finally converge to the correct position.

The results are depicted in Tab. II. As can be seen, our approach is able to outperform the baselines in terms of RMSE on most sequences showing that our approach can provide reliable pose information once it is converged. We believe this is due to the continuous map representation and therefore we are not limited by the grid resolution. The convergence rate of our approach to the correct position is highly competitive w.r.t. baselines. Our approach runs at an average framerate of around 2.6 Hz. A visualization of the localization is depicted in Fig. 6.

D. Ablation Studies

In this section, we will provide ablation studies on certain hyperparameters to validate our choices and provide a deeper insight in the behaviour of the approach. In the following, we will first look at the different loss terms, and later which influence the type of backbone has. We conduct the ablation studies on the scan registration task, as described in Sec. IV-B.

TABLE III: Ablation: Loss Function

	project. dist.	α_{Eik}	α_{end}	α_n	MAE($\boldsymbol{t}$) [m]	MAE($\boldsymbol{R}$) [deg]
[A]	✓	✗	✗	✗	0.103	0.269
[B]	✓	✗	✓	✓	0.063	0.103
[C]	✓	✓	✗	✓	0.198	0.449
[D]	✓	✓	✓	✗	-	-
[E]	✓	✓	✓	✓	0.062	0.100
[F]	✗	✓	✓	✓	0.313	0.886

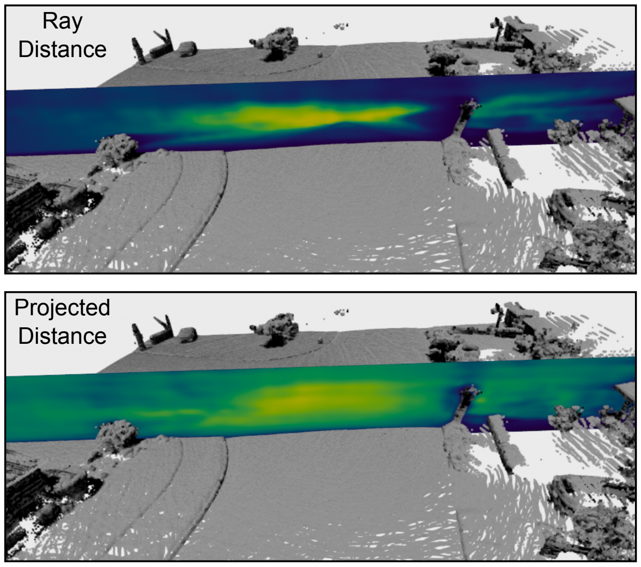


Fig. 7: A visualization of the distance fields when supervising the distance field with the ray distance (top), as well as with the projected distance (bottom). The distance fields are evaluated on a slice in the middle of the scene and depicted by the color. When using the ray distance, one can see the field of view of the LiDAR sensor, leading to a gradient from the surface to the LiDAR center. For the projected distance on the other hand it looks more like a true Euclidean distance field where the gradient ascents radialy from the surface. Note, that the NDF is only supervised for the regions within the field of view of the sensor, leading to wrong values in unobserved areas.

All numbers provided in the following are evaluated on the validation set.

1) Loss Function: In this experiment, we validate the choice of our loss function. For this, we enable (✓; taking the default α) or disable (✗; set $\alpha = 0$) certain parts of the loss functions. In Tab. III are the results of this experiment shown. Disabling all the additional losses [A] increases the error by a factor of around 2 w.r.t. enabling them [E], showing the importance of the regularization losses. Disabling the regularizations of the gradients [B], [D] deteriorates the performance slightly to completly. While only disabling α_{end} [C] results in even worse performance than disabling all [A], suggesting the loss is useful to mitigate undesired effects of the other terms. Lastly in [F], we supervise by the ray distance d_i instead of the projected distance $\tilde{d}_i$ where the performance substantially degrades. For better understanding this phenomen, we show a slice of the distance fields in Fig. 7. The gradient for the ray distance looks towards the LiDAR sensor, pulling the points not towards the closest surface but along the ray. The projected distance n_i points more towards the closest surface.

TABLE IV: Ablation: Backbone & Feature Size

	Backbone	Feature Size	MAE($\boldsymbol{t}$) [m]	MAE($\boldsymbol{R}$) [deg]	Memory [MB]
[G]	NeRF	32	0.215	0.402	0.934
[H]	NeRF	64	**0.061**	0.102	2.648
[I]	NeRF	128	0.104	0.257	8.989
[J]	NeRF	256	**0.061**	**0.100**	31.677
[K]	NeRF	512	0.084	0.122	125.289
[L]	SIREN	32	-	-	-
[M]	SIREN	64	0.126	0.395	1.541
[N]	SIREN	128	**0.062**	**0.100**	5.051
[O]	SIREN	256	0.068	0.101	17.732
[P]	SIREN	512	0.068	**0.099**	69.835

2) Backbone and Feature Size: In this experiment, we investigate the impact of the backbone and the network size on the localization ability. The first backbone we use is the classical NeRF [25] architecture with a positional encoding and an MLP with 8 layers, layer norm, and leaky ReLU. There is a skip connection from the positional encoding to the 6th layer. The second network is a SIREN [33], an MLP with 5 layers, layer norm, and sine nonlinearity. The results of this experiment are depicted in Tab. IV. With both backbones, we can see that a bigger network size does not necessarily mean a better localization performance. For the SIREN network, the results look more stable with the best performance for a hidden feature dimension of 128 ([N]). The classical NeRF network has less consistent but similar results ([G]-[K]). The results are in line with Sitzmann et al. [33] stating that the supervision of derivatives works better for SIRENs than for ReLU-based networks.

E. Limitations and Future Work

Despite these encouraging results, there is further space for improvement. The training time at each key pose takes around 20 min, which makes it only suitable for offline mapping but prohibits building the maps on the fly as it would be needed for online SLAM applications. In our case, we only optimized for obtaining the distance to the surface, but it would be interesting to regress point attributes such as semantics or colors as obtained from RGB-D sensors.

V. Conclusion

In this paper, we propose to use neural distance fields (NDFs) for robot localization. We showed how to directly learn the NDF from range sensor observations by projecting the measurements along the gradients of the network. The NDF provides us with a discretization-free and highly memory-effective distance field, that allows us to compute directions to the closest surface elegantly through the Jacobian. As a result of that, ICP can be used to register point clouds directly to the NDF without the need of searching for data associations. Additionally, we have shown how to globally localize in an NDF using Monte Carlo localization. Both use cases show competitive results considering multiple existing baseline approaches. We believe that having a continuous and memory efficient representation as the NDF can be useful for many robotic applications.

References

[1] D. Azinović, R. Martin-Brualla, D.B. Goldman, M. Nießner, and J. Thies. Neural RGB-D Surface Reconstruction. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2022.

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

[3] J. Behley, V. Steinhage, and A.B. Cremers. Efficient Radius Neighbor Search in Three-dimensional Point Clouds. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2015.

[4] J. Bentley. Multidimensional binary search trees used for associative searching. *Communications of the ACM*, 18(9):509–517, 1975.

[5] P. Besl and N. McKay. A Method for Registration of 3D Shapes. *IEEE Trans. on Pattern Analysis and Machine Intelligence (TPAMI)*, 14(2):239–256, 1992.

[6] F. Boniardi, A. Valada, R. Mohan, T. Caselitz, and W. Burgard. Robot localization in floor plans using a room layout edge extraction network. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2019.

[7] Y. Chen and G. Medioni. Object modeling by registration of multiple range images. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 1991.

[8] J. Chibane, A. Mir, and G. Pons-Moll. Neural Unsigned Distance Fields for Implicit Function Learning. In *Proc. of the Conf. on Neural Information Processing Systems (NeurIPS)*, 2020.

[9] B. Curless and M. Levoy. A Volumetric Method for Building Complex Models from Range Images. In *Proc. of the Intl. Conf. on Computer Graphics and Interactive Techniques (SIGGRAPH)*, pages 303–312. ACM, 1996.

[10] 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.

[11] 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.

[12] D. Fox. Kld-sampling: Adaptive particle filters. In *Proc. of the Advances in Neural Information Processing Systems (NIPS)*, 2001.

[13] S. Geman and D. McClure. Bayesian image analysis: An application to single photon emission tomography. *American Statistical Association*, pages 12–18, 1985.

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

[15] G. Grisetti, G. Tipaldi, C. Stachniss, W. Burgard, and D. Nardi. Fast and Accurate SLAM with Rao-Blackwellized Particle Filters. *Journal on Robotics and Autonomous Systems (RAS)*, 55(1):30–38, 2007.

[16] A. Gropp, L. Yariv, N. Haim, M. Atzmon, and Y. Lipman. Implicit Geometric Regularization for Learning Shapes. *arXiv preprint arXiv:2002.10099*, 2020.

[17] T. Guadagnino, X. Chen, M. Sodano, J. Behley, G. Grisetti, and C. Stachniss. Fast Sparse LiDAR Odometry Using Self-Supervised Feature Selection on Intensity Images. *IEEE Robotics and Automation Letters (RA-L)*, 7(3):7597–7604, 2022.

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

[19] P.J. Huber. Robust estimation of a location parameter. *Annals of Mathematical Statistics*, 35(1):73–101, 1964.

[20] 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.

[21] T. Lewiner, H. Lopes, A.W. Vieira, and G. Tavares. Efficient implementation of marching cubes cases with topological guarantees. *Journal of Graphics Tools*, 8(2):1–15, 2003.

[22] 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)*, pages 6389–6398, 2019.

[23] 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.

[24] D. Meagher. Octree Encoding: A New Technique for the Representation, Manipulation and Display of Arbitrary 3-D Objects by Computer. *Technical Report*, Image Processing Laboratory, Rensselaer Polytechnic Institute (IPL-TR-80-111), 1980.

[25] 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.

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

[27] M. Nießner, M. Zollhöfer, S. Izadi, and M. Stamminger. Real-time 3D Reconstruction at Scale using Voxel Hashing. In *Proc. of the SIGGRAPH Asia*, 2013.

[28] 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.

[29] C. Reiser, S. Peng, Y. Liao, and A. Geiger. KiloNeRF: Speeding Up Neural Radiance Fields With Thousands of Tiny MLPs. In *Proc. of the IEEE/CVF Intl. Conf. on Computer Vision (ICCV)*, 2021.

[30] A. Rosenfeld and J.L. Pfaltz. Sequential operations in digital picture processing. *Journal of the ACM*, 13(4):471–494, 1966.

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

[32] J. Serafin and G. Grisetti. NICP: Dense Normal Based Point Cloud Registration. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, pages 742–749, 2015.

[33] V. Sitzmann, J. Martel, A. Bergman, D. Lindell, and G. Wetzstein. Implicit Neural Representations with Periodic Activation Functions. In *Proc. of the Conf. on Neural Information Processing Systems (NeurIPS)*, 2020.

[34] T. Stoyanov, J. Saarinen, H. Andreasson, and A. Lilienthal. Normal Distributions Transform Occupancy Map Fusion: Simultaneous Mapping and Tracking in Large Scale Dynamic Environments. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2013.

[35] J. Stückler and S. Behnke. Multi-Resolution Surfel Maps for Efficient Dense 3D Modeling and Tracking. *Journal of Visual Communication and Image Representation (JVCIR)*, 25(1):137–147, 2014.

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

[37] I. Ueda, Y. Fukuhara, H. Kataoka, H. Aizawa, H. Shishido, and I. Kitahara. Neural Density-Distance Fields. In *Proc. of the Europ. Conf. on Computer Vision (ECCV)*, 2022.

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

[39] I. Vizzo, T. Guadagnino, J. Behley, and C. Stachniss. Vdbfusion: Flexible and efficient tsdf integration of range sensor data. *Sensors*, 22(3), 2022.

[40] I. Vizzo, T. Guadagnino, B. Mersch, L. Wiesmann, J. Behley, and C. Stachniss. KISS-ICP: In Defense of Point-to-Point ICP – Simple, Accurate, and Robust Registration If Done the Right Way. *IEEE Robotics and Automation Letters (RA-L)*, 8(2):1–8, 2023.

[41] L. Wiesmann, T. Guadagnino, I. Vizzo, G. Grisetti, J. Behley, and C. Stachniss. DCPCR: Deep Compressed Point Cloud Registration in Large-Scale Outdoor Environments. *IEEE Robotics and Automation Letters (RA-L)*, 7(3):6327–6334, 2022.

[42] L. Wiesmann, A. Milioto, X. Chen, C. Stachniss, and J. Behley. Deep Compression for Dense Point Cloud Maps. *IEEE Robotics and Automation Letters (RA-L)*, 6:2060–2067, 2021.

[43] F. Williams, M. Trager, J. Bruna, and D. Zorin. Neural Splines: Fitting 3D Surfaces with Infinitely-Wide Neural Networks. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2021.

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

[45] 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.

[46] 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.