Range Image-based LiDAR Localization for Autonomous Vehicles

Xieyuanli Chen   Ignacio Vizzo   Thomas Läbe   Jens Behley   Cyrill Stachniss

Abstract— Robust and accurate, map-based localization is crucial for autonomous mobile systems. In this paper, we exploit range images generated from 3D LiDAR scans to address the problem of localizing mobile robots or autonomous cars in a map of a large-scale outdoor environment represented by a triangular mesh. We use the Poisson surface reconstruction to generate the mesh-based map representation. Based on the range images generated from the current LiDAR scan and the synthetic rendered views from the mesh-based map, we propose a new observation model and integrate it into a Monte Carlo localization framework, which achieves better localization performance and generalizes well to different environments. We test the proposed localization approach on multiple datasets collected in different environments with different LiDAR scanners. The experimental results show that our method can reliably and accurately localize a mobile system in different environments and operate online at the LiDAR sensor frame rate to track the vehicle pose.

I. INTRODUCTION

Precise localization is a fundamental capability required by most autonomous mobile systems. With a localization system, a mobile robot or an autonomous car is capable to estimate its pose in a map based on observations obtained with onboard sensors. Precise and reliable LiDAR-based global localization is needed for autonomous driving, especially in GPS-denied environments or situations where GPS cannot provide accurate localization results.

Most autonomous mobile systems have a 3D LiDAR sensor onboard to perceive the environment and directly provide 3D range measurements. In this paper, we tackle the problem of vehicle localization based on such 3D LiDAR sensors. For localization, probabilistic state estimation techniques are used in most localization systems today. In particular, particle filters are a versatile tool as they do not need to restrict the motion or observation model to follow a specific distribution, such as a Gaussian. When utilizing particle filters, we need to design an appropriate observation model in lieu with a map representation. Frequently used observation models for LiDARs are the beam-end point model, also called the likelihood field [25], the ray-casting model [8], or models based on handcrafted features [23], [33]. These methods either only work efficiently with 2D LiDAR scanners or need carefully designed features to work properly. Recently, researchers also focused on data-driven learning of such observation models [12], [29], [6], which provide accurate
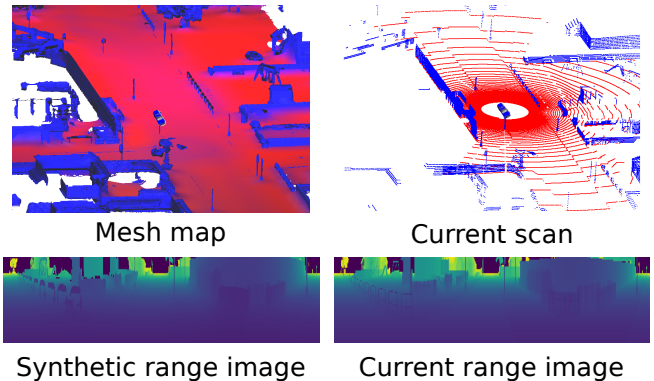
Fig. 1: Visualization of range images and triangular mesh map. On the left, we show the triangular mesh used as the map and the rendered synthetic range image from the mesh. In the mesh map, red parts correspond to ground planes and blue parts represent non-ground structures. On the right, we show the LiDAR point cloud at the same location and the corresponding range image generated from the LiDAR scan.

and reliable results as long as the environment is close to the environment used for learning the model.

Instead of using raw point clouds obtained from a 3D LiDAR sensor or features generated or learned from the point clouds, we investigate range images for 3D LiDAR-based localization for autonomous vehicles. As shown in Fig. 1, we project the point clouds into range images and localize the autonomous system with rendered views from a map that is represented with a triangular mesh. There are several reasons to use range image representation and maps represented by a mesh. The cylindrical range image is a natural and light-weight representation of the scan from a rotating 3D LiDAR, and a mesh map is more compact than a large point cloud. Those properties enable our approach to achieve global localization in large-scale environments. Furthermore, the rendering of range images from a mesh map can be performed efficiently using computer graphics techniques. Therefore, range images and mesh maps are a perfect match for achieving LiDAR-based global localization.

The main contribution of this paper is a novel observation model for 3D LiDAR-based localization. Our model is based on range images generated from both, the real LiDAR scans and synthetic renderings of the *mesh* map. We use the difference between them to formulate the observation model for a Monte Carlo localization (MCL) for updating the importance weights of the particles. Based on our novel observation model, our approach provides (x, y, θ)-pose estimates for the vehicle and achieves global localization using 3D LiDAR

scans. Furthermore, our approach generalizes well over different environments collected with different types of LiDAR scanners. In sum, we make three key claims: Our approach is able to (i) achieve global localization accurately and reliably using 3D LiDAR data, (ii) can be used for different types of LiDAR sensors, and, (iii) generalize well over different environments. These claims are backed up by the paper and our experimental evaluation. The source code of our approach is available at: https://github.com/PRBonn/range-mcl.

II. RELATED WORK

For localization given a map, one often distinguishes between pose tracking and global localization. In pose tracking, the vehicle starts from a known pose and the pose is updated over time. In global localization, no pose prior is available. In this work, we address global localization using 3D LiDAR data without assuming any pose prior from GPS or other sensors. Therefore, we concentrate here mainly on LiDAR-based approaches.

In the context of autonomous cars, many approaches were proposed for accurate pose tracking using multiple sensor modalities and high-definition (HD) maps. Levinson et al. [16] utilize GPS, IMU, and LiDAR scans to build HD maps for localization. They generate a 2D surface image of ground reflectivity in the infrared spectrum and define an observation model that uses these intensities. The uncertainty in intensity values has been handled by building a prior map [31]. Barsan et al. [12] use a fully convolutional neural network (CNN) together with HD maps to perform online-to-map matching for improving the robustness to dynamic objects and eliminating the need for LiDAR intensity calibration. Merfels and Stachniss [18] present an efficient chain-like pose graph for vehicle localization exploiting graph optimization techniques and different sensing modalities. Based on this work, Wilbers et al. [30] propose a LiDAR-based localization system performing a combination of local data association between laser scans and HD map features, temporal data association smoothing, and a map matching approach for robustification. The approaches above show good performance for tracking vehicles' poses but require GPS information for operating on HD maps. In contrast, our approach addresses global localization using only 3D LiDAR data without assuming any pose prior.

To achieve global localization, traditional approaches rely on probabilistic state estimation techniques [25]. A popular framework is Monte Carlo localization [8], [26], [10], which uses a particle filter to estimate the robot's pose and is widely used in robot localization systems [3], [6], [15], [24], [32].

Recently, several approaches exploiting deep neural networks and semantic information for 3D LiDAR localization have been proposed. For example, Ma et al. [17] combine semantic information such as lanes and traffic signs in a Bayesian filtering framework to achieve accurate and robust localization within sparse HD maps. Yan et al. [32] exploit buildings and intersections information from a LiDAR-based semantic segmentation system [19] to localize in Open-StreetMap data. Schaefer et al. [21] detect and extract pole
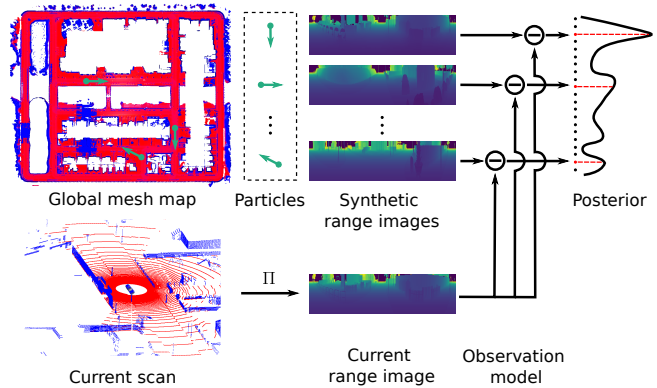

Fig. 2: Overview of our approach. We project the LiDAR point cloud into a range image and compare it to synthetic range images rendered at each particle location from a mesh map. Based on the range images, we propose a new observation model for localization and integrate it into a Monte Carlo localization system to estimate the pose posterior of the vehicle.

landmarks from 3D LiDAR scans for long-term urban vehicle localization whereas Tinchev et al. [27] propose a learning-based method to match segments of trees and localize in both urban and natural environments. Sun et al. [24] use a deep-probabilistic model to accelerate the initialization of the Monte Carlo localization and achieve a fast localization in outdoor environments. In our previous work [6], [5], we also exploit CNNs with semantics to predict the overlap between LiDAR scans as well as their yaw angle offset, and use this information to build a learning-based observation model for Monte Carlo localization. The learning-based methods perform well in the trained environments, while they usually cannot generalize well in different environments or different LiDAR sensors.

In sum, our method only uses LiDAR data to achieve global localization outdoors without using any GPS. Moreover, our approach uses range information directly without exploiting neural networks, semantics, or extracting landmarks. Therefore, it generalizes well to different environments and different LiDAR sensors and does not require new training data when moving to different environments.

III. OUR APPROACH

In this paper, we propose a probabilistic global localization system for autonomous vehicles using a 3D LiDAR sensor, see Fig. 2 for an illustration. To this end, we project the LiDAR point cloud into a range image (see Sec. III-A) and compare it to synthetic range images rendered at each particle location from a map represented by a triangular mesh (see Sec. III-B and Sec. III-C). Based on the range images, we propose a new observation model for LiDAR-based localization (see Sec. III-E) and integrate it into a Monte Carlo localization system (see Sec. III-D). Furthermore, we employ a tile map to accelerate the rendering and decide when the system converges (see Sec. III-F). Using an OpenGL-based rendering pipeline, the proposed system operates online at the frame rate of the LiDAR sensor after convergence.

A. Range Image Generation

The key idea of the proposed method is to use range images generated from LiDAR scans and rendered from the triangular mesh map for robot localization. To generate range images, we use a spherical projection [2], [19], [7], [5]. We project the point cloud $\mathcal{P}$ to the so-called vertex map $\mathcal{V}_D : \mathbb{R}^2 \mapsto \mathbb{R}^3$, where each pixel contains the nearest 3D point. Each point $\mathbf{p}_i = (x, y, z)$ is converted via the function $\Pi : \mathbb{R}^3 \mapsto \mathbb{R}^2$ to spherical coordinates and finally to image coordinates (u, v), i.e.,

$$\begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} \frac{1}{2} \left[1 - \arctan(y, x) \cdot \pi^{-1} \right] \cdot w \\ \left[1 - \left(\arcsin(z \cdot r^{-1}) + \mathrm{f}_{\mathrm{up}} \right) \mathrm{f}^{-1} \right] \cdot h \end{pmatrix}, \quad (1)$$

where $r = ||\mathbf{p}||_2$ is the range, $\mathrm{f} = \mathrm{f}_{\mathrm{up}} + \mathrm{f}_{\mathrm{down}}$ is the vertical field-of-view of the sensor, and w, h are the width and height of the resulting vertex map $\mathcal{V}_D$. Given the vertex map $\mathcal{V}_D$ and range r of points at each coordinate (u, v), we generate the corresponding range image $\mathcal{R}_D$, on which the subsequent computations are based upon.

B. Mesh-based Map Representation

We use a triangular mesh as a map $\mathcal{M}$ of the environment. A triangular mesh provides us with a compact representation that enables us to render the aforementioned range images at the frame rate of the LiDAR sensor.

To generate the map, we use LiDAR scans together with their poses provided a SLAM system uses for mapping. Note that it is not necessary for our approach to use the same LiDAR sensor for map generation and localization. We employ Poisson surface reconstruction (PSR) [13] to obtain the representation of the map as a triangular mesh from point clouds. The PSR algorithm requires an oriented point cloud, thus normals for all the points in the input cloud. To estimate the surface normals, we use a range image-based normal estimation [2], which computes normals in the so-called normal map $\mathcal{N}_D$ for each coordinate (u, v) using cross products over forward differences of the corresponding vertex map pixel, i.e.,

$$\mathcal{N}_D((u, v)) = (\mathcal{V}_D((u+1, v)) - \mathcal{V}_D((u, v)))$$
$$\times (\mathcal{V}_D((u, v+1)) - \mathcal{V}_D((u, v))). \quad (2)$$

To decrease the storage size of the map, we use a ground segmentation algorithm. We first compute the empirical covariance matrix of all the points in the cloud. We then compute the normalized Eigenvectors of $\Sigma : \mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3$ with corresponding Eigenvalues $\lambda_1 \geq \lambda_2 \geq \lambda_3$. We use a simple, yet effective, approach to label a point $\mathbf{p}_i = (x, y, z)$ with corresponding normal $\mathbf{n}_i$ as ground. $\mathbf{p}_i$ is considered to belong to the ground surface if it satisfies the following criteria: $\mathbf{n}_i \cdot \mathbf{e}_3 > \cos \alpha_{\mathrm{thres}}$ and $z < z_{\mathrm{thres}}$.

After all ground points have been labeled, we proceed to run the reconstruction algorithm that retains the labels encoded as two different colors of the vertices $\mathbf{p}_i$ in the reconstructed triangles. To simplify the ground mesh, we split the complete mesh into non-ground and ground mesh. For the ground mesh, we contract all vertices to a single
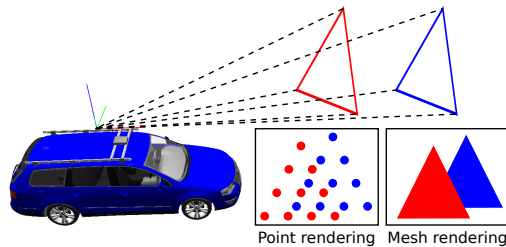

Fig. 3: Rendering example. In contrast to a map represented by a point cloud, triangular meshes are smoother and more compact. During the projection, for each triangle, only three vertices need to be projected. Moreover, triangles can better represent the occlusion relationship between different objects.

vertex that are inside a voxel of a given size s_{voxel}. Then, we filter the ground vertices using an average filter by replacing each vertex $\mathbf{v}_i$ with $\mathbf{v}_i^*$ averaging all adjacent vertices $v_n \in \mathcal{N}$:

$$\mathbf{v}_i^* = \frac{\mathbf{v}_i + \sum_{n \in \mathcal{N}} \mathbf{v}_n}{|\mathcal{N}| + 1}. \quad (3)$$

After averaging, invalid edges are removed. Once the ground surface has been simplified, it is combined with the rest of the mesh with no further processing. This simplification allows us to decrease the size of the mesh model to about 50% of its original size.

The PSR algorithm provides a global solution that considers the whole input data at once without resorting to heuristic partitioning [4] or blending [1], [22]. Therefore, instead of building the map incrementally [28], we aggregate all the oriented point clouds into a global reference frame using map poses, then, this oriented point cloud map is directly fed to the PSR algorithm, yielding a globally consistent triangular mesh of the environment.

C. Rendering Synthetic Range Images

Given a particle j with its state vector (x^j, y^j, θ^j) and the triangular mesh map $\mathcal{M}$, we use OpenGL to render a synthetic range image for that particle. Using the spherical projection, cf. Eq. (1), we project vertices of the triangles from the given particle pose and let OpenGL shade the triangle surface considering the occlusion, as shown in Fig. 3. To further accelerate rendering, we render batches of range images for multiple particles using instancing of the map, which allows us to avoid reading vertex positions multiple times and minimizes the number of draw calls.

D. Monte Carlo Localization

Monte Carlo localization (MCL) is commonly implemented using a particle filter [8]. In our case, each particle represents a hypothesis for the autonomous vehicle's 2D pose $\mathbf{x}_t = (x, y, \theta)_t$ at time t. When the robot moves, the pose of each particle is updated based on a motion model with the control input $\mathbf{u}_t$. The expected observation from the predicted pose of each particle in the map $\mathcal{M}$ is then compared to the actual observation $\mathbf{z}_t$ acquired by the robot to update the particle's weight based on an observation model. Particles are resampled according to their weight distribution

(a) Location heatmap   (b) Heading likelihood

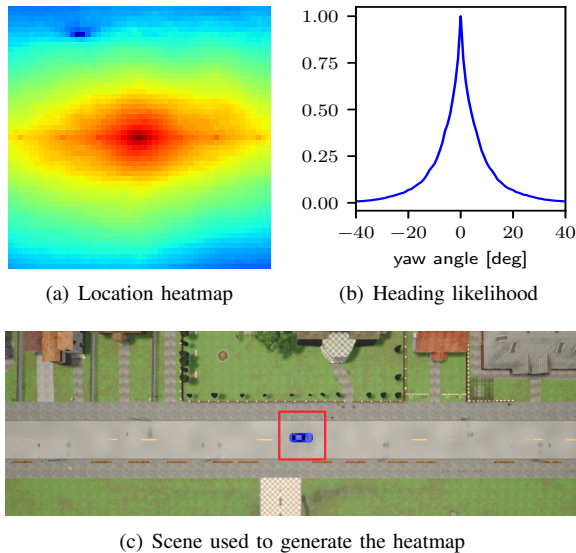
(c) Scene used to generate the heatmap

Fig. 4: Range image-based observation model. (a) A local heatmap shows the location likelihood of the scan at the car's position with respect to the map with the same heading. Red shades correspond to higher weights. (b) Heading likelihood of the observation model when changing the yaw angle with the same location. (c) A top-down view of the Carla scene used in this example.

and resampling is triggered whenever the effective number of particles drops below a specific threshold [11]. After several iterations of this procedure, the particles eventually converge around the true pose.

MCL realizes a recursive Bayesian filter estimating a probability density $p(\mathbf{x}_t \mid \mathbf{z}_{1:t}, \mathbf{u}_{1:t})$ over the pose $\mathbf{x_t}$ given all observations $\mathbf{z}_{1:t}$ up to time t and motion controls $\mathbf{u}_{1:t}$ up to time t. This posterior is updated as follows:

$$p(\mathbf{x}_t \mid \mathbf{z}_{1:t}, \mathbf{u}_{1:t}) = \eta \; p(\mathbf{z}_t \mid \mathbf{x}_t, \mathcal{M}) \cdot$$
$$\int p(\mathbf{x}_t \mid \mathbf{u}_t, \mathbf{x}_{t-1}) \; p(\mathbf{x}_{t-1} \mid \mathbf{z}_{1:t-1}, \mathbf{u}_{1:t-1}) \; d\mathbf{x}_{t-1}, \quad (4)$$

where η is a normalization constant, $p(\mathbf{x}_t \mid \mathbf{u}_t, \mathbf{x}_{t-1})$ is the motion model, $p(\mathbf{z}_t \mid \mathbf{x}_t, \mathcal{M})$ is the observation model, and $p(\mathbf{x}_{t-1} \mid \mathbf{z}_{1:t-1}, \mathbf{u}_{1:t-1})$ is the probability distribution for the prior state $\mathbf{v}_{t-1}$.

In this work, we focus on the observation model and employ a standard motion model for vehicles [25].

E. Range Image-based Observation Model

Based on the generated range image from the current LiDAR scan and the rendered synthetic range images for all particles, we design an observation model.

Each particle j represents a pose hypothesis $\mathbf{x}_t^j = (x, y, \theta)_t^j$ at time t. Given the corresponding synthetic rendered range image $\mathbf{z}^j$ for the j-th particle rendered at the particle's pose hypothesis, we compare it to the current range image $\mathbf{z}_t$ generated from the LiDAR point cloud. The likelihood $p(\mathbf{z}_t \mid \mathbf{x}_t, \mathcal{M})$ of the j-th particle is then approximated using a Gaussian distribution:

$$p(\mathbf{z}_t \mid \mathbf{x}_t, \mathcal{M}) \propto \exp\left(-\frac{1}{2} \frac{d(\mathbf{z}_t, \mathbf{z}^j)^2}{\sigma_d^2} \right), \quad (5)$$

where d corresponds to the difference between or similarity of the range images $\mathbf{z}_t$ and $\mathbf{z}^j$.

There are several ways to calculate this similarity. For example, we could directly compare two range images at pixel level with absolute differences or using a cross-correlation. One could also generate features and compare two images in the feature space. Recently, there are also many deep learning-based algorithms proposed [12], [6], [24], [29].

In this paper, our goal is to investigate the use of range images generated from LiDAR scans and triangular meshes for a Monte Carlo localization system and the particular choice of similarity computation is application dependent. To keep the whole system fast and easy to use, we opted for a fast-to-compute and effective method and use $d = N^{-1} \sum |\mathbf{z}_t - \mathbf{z}^j|$, i.e., the mean of the absolute pixel-wise differences, where N is the number of valid pixels in the current range image. Our results show that this choice is effective and generalizes well to datasets collected in different environments with different types of LiDAR sensors (see Sec. IV-B).

Fig. 4 shows the probabilities in a local area calculated by our proposed observation model and shows that it encodes the pose hypotheses very well. In contrast to our prior work [5], which decouples the observation model into two parts, location likelihood and heading likelihood, the proposed observation model can estimate the likelihood for the whole state space $\mathbf{x}_t = (x, y, \theta)_t$ at once using one model, which is elegant and fast.

F. Tiled Map Representation

We split the global mesh-based map into tiles to accelerate the Monte Carlo localization by more efficient rendering. We only use parts of the mesh associated to a tile, which are close to the particle position. Besides more efficient rendering, we also use tiles to determine when the localization has converged. If all particles are localized in at most N_{conv} tiles, we assume that the localization has converged and reduce the number of particles to track the pose. Tiles also enable the runtime of our method to be independent of the size of the whole environment after converging.

IV. EXPERIMENTAL EVALUATION

We present our experiments to show the capabilities of our method and to support our claims, that our approach is able to: (i) achieve global localization accurately and reliably using 3D LiDAR data, (ii) can be used for different types of LiDAR scanners, and, (iii) generalize well to different environments without changing parameters.

Implementation Details. We implement our code based on Python and OpenGL. To generate a triangular mesh map, we use the PSR implementation from Open3D [34]. For ground point extraction, we use $\alpha_{\mathrm{thres}} = 30°$ and z_{thres} as the sensor mounted height, and employ a voxel grid with voxel sizes $s_{\mathrm{voxel}} = 1.0\,m$. We use tiles of size $100 \times 100\,\mathrm{m}^2$ and we reduce the number of particles from initially $10,000$ to 100 particles after convergence, i.e., only $N_{\mathrm{conv}} = 1$ tile is covered by particles. The size of the tile map and the reduced number of particles are trade-offs between runtime

TABLE I: Dataset Overview

Dataset	Sensor	Sequence	Acquisition time	Length
Carla (Simulator)	8 - 128 beam LiDAR	map (0.2 Gb)	n/a	3.5 km
		00	n/a	0.7 km
IPB-Car (Germany)	Ouster 64	map (0.8 Gb)	02/2020	6.2 km
		00	09/2019	1.7 km
		01	11/2019	1.9 km
MulRan (Korea)	Ouster 64	map (0.5 Gb)	08/2019	6.0 km
		00	06/2019	6.1 km
Apollo (U.S.)	Velodyne 64	map (5.4 Gb)	09/2018	44.8 km
		00	10/2018	8.8 km

Fig. 5: Localization results using $10,000$ particles on the IPB-Car dataset. Shown are the mesh map, the ground truth trajectory (black), the overlap-based result (blue), and the result of our proposed method (orange).

and accuracy. We set $\sigma_d = 5$ in Eq. (5), and we only update the weights of particles when the car is moving. All parameters are tuned with one dataset (IPB-Car) and kept the same for all other experiments with different datasets and sensors.

Datasets. We evaluate the generalization ability of our method using multiple datasets, including Carla [9], IPB-Car [6], MulRan (KAIST) [14] and Apollo (Columbia-Park) [35]. These datasets are collected in different environments with different types of LiDAR scanners over different times, see Tab. I for more details. For the Carla simulator, we added objects for the test sequences, which are not present in the map to simulate a changing environment like in the real datasets. For all experiments on different datasets, we only change the intrinsic and extrinsic calibration parameters of the LiDAR sensors for the generation of the range images and keep all other parameters, especially those of the MCL, the same.

Baselines. In the following experiments, we use the same MCL framework and only change the observation models. We compare our method with three baseline observation models: the typical beam-end model [25], a histogram-based model derived from the work of Röhling et al. [20], and a deep learning-based model [6].

The beam-end observation model is often used for 2D LiDAR data. For 3D LiDAR scans, it needs many more particles to make sure that it converges to the correct pose, which causes the computation time to increase substantially. In this paper, we implement the beam-end model with a down-sampled point cloud map using voxelization with a resolution of $10\,\text{cm}$.

Our second baseline for comparison is inspired by Röhling et al. [20], which exploit the use of similarity measures on histograms extracted from 3D LiDAR data.

The third baseline is the overlap-based localization [6]. It uses a deep neural network to estimate the overlap and yaw angle offset between a query scan and map data, and on top of this builds an observation model for MCL. The overlap-based method utilizes a grid map and stores virtual frames for each grid cell. We refer to the corresponding paper [6] for more details.
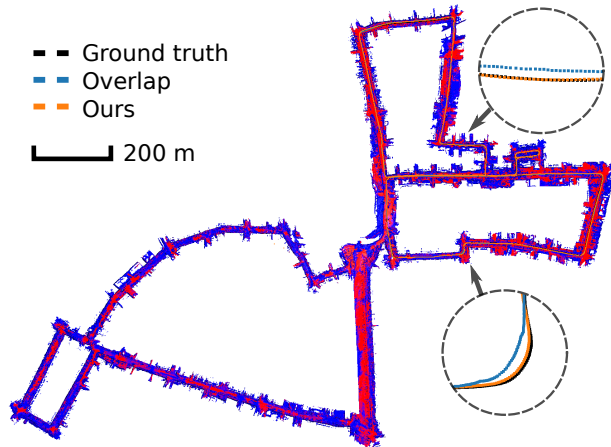
A. Localization Performance

The experiment presented in this section investigates the localization performance of our approach. It supports the claim that our approach achieves global localization accurately and reliably using 3D LiDAR data.

For qualitative results, we show the trajectories of the localization results tested on the IPB-Car dataset in Fig. 5. The results illustrate that the proposed method localizes well in the map using only LiDAR data collected in dynamic environments at different times. Comparing to the baseline methods, the proposed method tracks the pose more accurately.

For quantitative results, we first calculate the success rate for different methods with different numbers of particles comparing our approach to the aforementioned methods, see Fig. 6. We tested the methods using five different numbers of particles $N = \{1,000, 5,000, 10,000, 50,000, 100,000\}$. For each setup, we sample 10 trajectories and perform global localization. The x-axis represents the number of particles, while the y-axis is the success rate of different setups. The success rate for a specific setup of one method is calculated using the number of success cases divided by the total number of the tests. To decide whether one test is successful or not, we check the location error by every 100 frames after convergence. If the location error is smaller than $5\,\text{m}$, we count this run as a success.

Quantitative results of localization accuracy are shown in Tab. II. The upper part shows the location error of all methods tested with both sequences. The location error is defined as the root mean square error (RMSE) of each test in terms of (x, y) Euclidean error with respect to the ground truth poses. It shows the mean and the standard deviation of the error for each observation model. Note that the location error is only calculated for success cases with $10,000$ particles. The lower part shows the yaw angle error. It is the RMSE of each test in terms of yaw angle error with respect to the ground truth poses. The table shows the mean
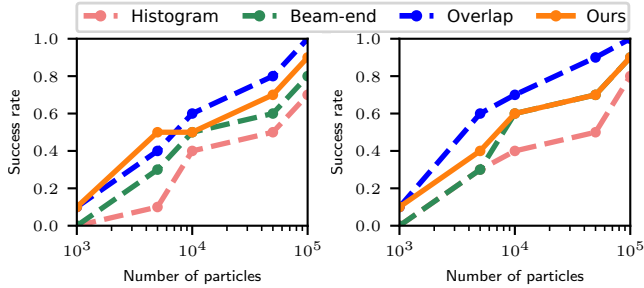
Fig. 6: Success rate of the different observation models for 10 globalization runs. Here, we use sequence 00 (left) and sequence 01 (right) to localize in the map of the IPB-Car dataset.

TABLE II: Localization results on the IPB-Car dataset

Sequence	Location error [m]			
	Beam-end	Histogram-based	Overlap-based	Ours
00	0.92 ± 0.27	1.85 ± 0.34	0.81 ± 0.13	$\mathbf{0.66 \pm 0.12}$
01	0.67 ± 0.11	1.86 ± 0.34	0.88 ± 0.07	$\mathbf{0.44 \pm 0.03}$

Sequence	Yaw angle error [deg]			
	Beam-end	Histogram-based	Overlap-based	Ours
00	1.87 ± 0.47	3.10 ± 3.07	1.74 ± 0.11	$\mathbf{1.69 \pm 0.11}$
01	2.10 ± 0.59	3.11 ± 3.08	$\mathbf{1.88 \pm 0.09}$	2.53 ± 0.79

and the standard deviation of the error for each observation model. As before, the yaw angle error is also only calculated for cases in which the global localization converged with $10,000$ particles.

The quantitative results show that our method outperforms all baseline methods in location accuracy while achieving similar heading accuracy. The reason is that our method uses online rendered range images and does not rely on discrete grids. Therefore, our method will not be affected by the resolution of the grid. However, this means that our method requires more particles to achieve the same success rate. Thus, we use a large amount of particles for initialization. It will then achieve a high success rate while without influencing the runtime after convergence, due to the use of tiles, see in Sec. IV-C.

B. Generalization

The experiment supports the claim that our method is able to use different types of LiDAR sensors to localize in the same mesh map. We test 5 different types of LiDAR sensors in the Carla simulator, including Quanergy MQ-8 (8-beams), Velodyne Puck (16-beams), Velodyne HDL-32E (32-beams), Velodyne HDL-64E (64-beams), and Ouster OS1-128 (128-beams). We use the real parameters from the aforementioned LiDAR sensors, which include the number of beams and the field of view. As shown in Tab. III, our method works well with all different types of sensors and achieves good localization results even with relatively sparse scans (location RMSE of 0.48 m with the 8-beam LiDAR).

Tab. III and Fig. 7 also verify the claim that our method generalizes well over different environments. We test our method on both the MulRan and Apollo datasets with the same parameters used in the Carla and IPB-Car dataset. Our method works well also in Korean and U.S. urban environments.

TABLE III: Localization results on datasets using different sensors.

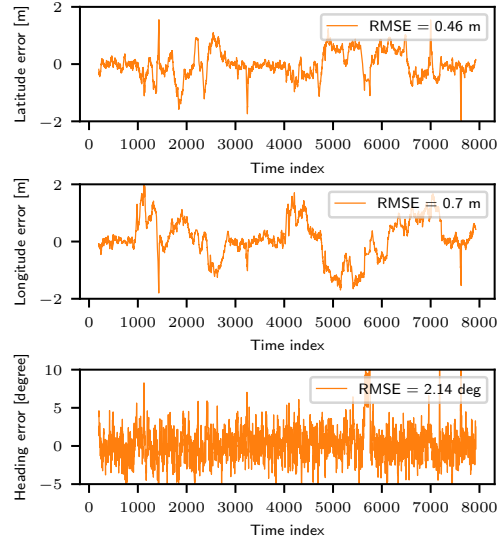| Dataset	Scanner	Location RMSE [m]	Yaw angle RMSE [deg]
Carla	8-beams	0.48	3.87
	16-beams	0.43	3.87
	32-beams	0.42	3.40
	64-beams	0.36	3.46
	128-beams	0.33	3.31
MulRan	Velodyne 64	0.83	3.14
Apollo	Ouster 64	0.57	3.40

Fig. 7: Localization results on the MulRan dataset w.r.t. provided GPS locations. The top figure shows the latitude error, the middle figure shows the longitude error, and the bottom figure shows the heading error.

C. Runtime

Here we show that our approach runs fast enough to support online processing on the robot at sensor frame rate. We tested our method on a regular computer with an Intel i7-8700 with 3.2 GHz and an Nvidia GeForce GTX 1080 Ti with 11 GB of memory. On the Carla dataset, before convergence, the maximum time for one query frame is 56.7 s with $10,000$ particles. After convergence, the average frame rates of our method is 21.8 Hz with 100 particles and the tile maps size of 100×100 m^2.

V. CONCLUSION

In this paper, we presented a novel range image-based online LiDAR localization approach. Our method exploits range images generated from LiDAR scans and a triangular mesh-based map representation. This allows us to localize autonomous systems in the given map successfully. We implemented and evaluated our approach on different datasets and provided comparisons to other existing techniques. The experiments suggest that our method is able to achieve global localization reliably and accurately. Moreover, our method generalizes well to different environments and can be used with different LiDAR sensors. After the localization convergence, our method can operate online at the sensor frame rate.

REFERENCES

[1] M. Alexa, J. Behr, D. Cohen-Or, S. Fleishman, D. Levin, and C.T. Silva. Point set surfaces. In *Proc. of the IEEE Visualization, (VIS)*, pages 21–29. IEEE, 2001.

[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] M. Bennewitz, C. Stachniss, W. Burgard, and S. Behnke. Metric Localization with Scale-Invariant Visual Features using a Single Perspective Camera. In H.I. Christensen, editor, *European Robotics Symposium 2006*, volume 22 of *STAR Springer Tracts in Advanced Robotics*, pages 143–157. Springer Verlag, 2006.

[4] J.C. Carr, R.K. Beatson, J.B. Cherrie, T.J. Mitchell, W.R. Fright, B.C. McCallum, and T.R. Evans. Reconstruction and representation of 3d objects with radial basis functions. In *Proc. of the Intl. Conf. on Computer Graphics and Interactive Techniques (SIGGRAPH)*, pages 67–76, 2001.

[5] X. Chen, T. Läbe, A. Milioto, T. Röhling, O. Vysotska, A. Haag, J. Behley, and C. Stachniss. OverlapNet: Loop Closing for LiDAR-based SLAM. In *Proc. of Robotics: Science and Systems (RSS)*, 2020.

[6] 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.

[7] X. Chen, A. Milioto, E. Palazzolo, P. Giguère, J. Behley, and C. Stachniss. SuMa++: Efficient LiDAR-based Semantic SLAM. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2019.

[8] F. Dellaert, D. Fox, W. Burgard, and S. Thrun. Monte carlo localization for mobile robots. In *IEEE International Conference on Robotics and Automation (ICRA)*, May 1999.

[9] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun. CARLA: An Open Urban Driving Simulator. In *Proc. of the Conf. on Robot Learning (CoRL)*, 2017.

[10] D. Fox, W. Burgard, F. Dellaert, and S. Thrun. Monte Carlo Localization: Efficient Position Estimation for Mobile Robots. *Proc. of the Conference on Advancements of Artificial Intelligence (AAAI)*, 1999(343-349), 1999.

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

[12] I. Andrei I.A. Barsan, S. Wang, A. Pokrovsky, and R. Urtasun. Learning to Localize Using a LiDAR Intensity Map. In *Proc. of the Second Conference on Robot Learning (CoRL)*, pages 605–616, 2018.

[13] M. Kazhdan, M. Bolitho, and H. Hoppe. Poisson surface reconstruction. In *Proceedings of the fourth Eurographics symposium on Geometry processing*, volume 7, 2006.

[14] G. Kim, Y.S. Park, Y. Cho, J. Jeong, and A. Kim. Mulran: Multimodal range dataset for urban place recognition. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2020.

[15] R. Kümmerle, M. Ruhnke, B. Steder, C. Stachniss, and W. Burgard. Autonomous Robot Navigation in Highly Populated Pedestrian Zones. *Journal of Field Robotics (JFR)*, 2014.

[16] J. Levinson, M. Montemerlo, and S. Thrun. Map-Based Precision Vehicle Localization in Urban Environments. In *Proc. of Robotics: Science and Systems (RSS)*, 2007.

[17] W. Ma, I. Tartavull, I. A. Bârsan, S. Wang, M. Bai, G. Mattyus, N. Homayounfar, S. K. Lakshmikanth, A. Pokrovsky, and R. Urtasun. Exploiting Sparse Semantic HD Maps for Self-Driving Vehicle Localization. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2019.

[18] Ch. Merfels and C. Stachniss. Pose fusion with chain pose graphs for automated driving. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2016.

[19] A. Milioto, I. Vizzo, J. Behley, and C. Stachniss. RangeNet++: Fast and Accurate LiDAR Semantic Segmentation. In *Proceedings of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2019.

[20] T. Röhling, J. Mack, and D. Schulz. A Fast Histogram-Based Similarity Measure for Detecting Loop Closures in 3-D LIDAR Data. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, pages 736–741, 2015.

[21] A. Schaefer, D. Büscher, J. Vertens, L. Luft, and W. Burgard. Long-term urban vehicle localization using pole landmarks extracted from 3-D lidar scans. In *Proc. of the Europ. Conf. on Mobile Robotics (ECMR)*, pages 1–7, 2019.

[22] C. Shen, J.F. O'Brien, and J.R. Shewchuk. Interpolating and approximating implicit surfaces from polygon soup. In *Proc. of the Intl. Conf. on Computer Graphics and Interactive Techniques (SIGGRAPH)*, pages 896–904, 2004.

[23] B. Steder, M. Ruhnke, S. Grzonka, and W. Burgard. Place Recognition in 3D Scans Using a Combination of Bag of Words and Point Feature Based Relative Pose Estimation. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2011.

[24] L. Sun, D. Adolfsson, M. Magnusson, H. Andreasson, I. Posner, and T. Duckett. Localising Faster: Efficient and precise lidar-based robot localisation in large-scale environments. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2020.

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

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

[27] G. Tinchev, A. Penate-Sanchez, and M. Fallon. Learning to see the wood for the trees: Deep laser localization in urban and natural environments on a CPU. *IEEE Robotics and Automation Letters (RA-L)*, 4(2):1327–1334, 2019.

[28] 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.

[29] X. Wei, I. A. Bârsan, S. Wang, J. Martinez, and R. Urtasun. Learning to Localize Through Compressed Binary Maps. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 10316–10324, 2019.

[30] D. Wilbers, Ch. Merfels, and C. Stachniss. Localization with Sliding Window Factor Graphs on Third-Party Maps for Automated Driving. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2019.

[31] R.W. Wolcott and R.M. Eustice. Fast lidar localization using multiresolution gaussian mixture maps. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, pages 2814–2821, 2015.

[32] F. Yan, O. Vysotska, and C. Stachniss. Global Localization on OpenStreetMap Using 4-bit Semantic Descriptors. In *Proc. of the Europ. Conf. on Mobile Robotics (ECMR)*, 2019.

[33] C. Zhang, M. H. Ang, and D. Rus. Robust lidar localization for autonomous driving in rain. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, pages 3409–3415, 2018.

[34] Q.Y. Zhou, J. Park, and V. Koltun. Open3D: A modern library for 3D data processing. *arXiv:1801.09847*, 2018.

[35] Y. Zhou, G. Wan, S. Hou, L. Yu, G. Wang, X. Rui, and S. Song. Da4ad: End-to-end deep attention-based visual localization for autonomous driving. In *Proc. of the Europ. Conf. on Computer Vision (ECCV)*, 2020.