Online Range Image-based Pole Extractor for Long-term LiDAR Localization in Urban Environments

Hao Dong* Xieyuanli Chen* Cyrill Stachniss

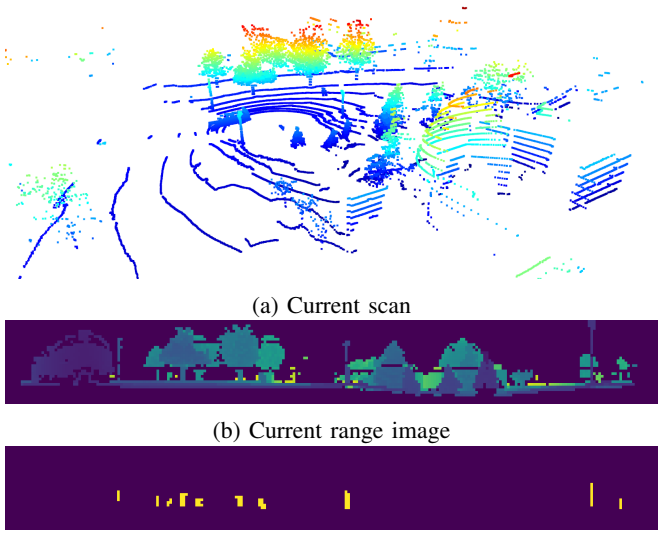
***Abstract*— Reliable and accurate localization is crucial for mobile autonomous systems. Pole-like objects, such as traffic signs, poles, lamps, etc., are ideal landmarks for localization in urban environments due to their local distinctiveness and long-term stability. In this paper, we present a novel, accurate, and fast pole extraction approach that runs online and has little computational demands such that this information can be used for a localization system. Our method performs all computations directly on range images generated from 3D LiDAR scans, which avoids processing 3D point cloud explicitly and enables fast pole extraction for each scan. We test the proposed pole extraction and localization approach on different datasets with different LiDAR scanners, weather conditions, routes, and seasonal changes. The experimental results show that our approach outperforms other state-of-the-art approaches, while running online without a GPU. Besides, we release our pole dataset to the public for evaluating the performance of pole extractor, as well as the implementation of our approach.**

I. Introduction

Accurate and reliable localization is a basic requirement for an autonomous robot. The accurate estimation of its state helps the robot to avoid collision with the obstacles, follow the traffic lanes, and perform other tasks. Reliability means here that the robot should adapt to changes in the environment, such as different weather, day and night, seasonal changes, etc.

GPS or GNSS-based localization systems are robust to appearance changes of the environments. However, in urban areas, they may suffer from low availability due to building and tree occlusions. Additional, map-based approaches are needed for precise and reliable localization for mobile robots. Multiple different types of sensors have been used to build the map of the environments, e.g. LiDAR scanners [10], [2], [24], monocular cameras [18], stereo cameras [8], etc. Among them, LiDAR sensors are more robust to the illumination changes, and multiple LiDAR-based effective and efficient mapping approaches have been proposed, for example, the work by Droeschel and Behnke [10] and by Behley and Stachniss [2]. However, these approaches often need large amounts of memory due to map representations, thus cannot generalize easily to large-scale scenes. If only specific features are used to build the map, such as traffic signs, trunks and other pole-like structures, the map size can be reduced significantly.

*Authors contributed equally. H. Dong is with the Aalto University, Finland. X. Chen and C. Stachniss are with the University of Bonn, Germany. This work has partially been funded by the European Unions Horizon 2020 research and innovation programme under grant agreement No 101017008 (Harmony), and by the Chinese Scholarship Committee.
978-1-6654-1213-1/21/$31.00



(a) Current scan

(b) Current range image

(c) Pole extraction result by our approach

Fig. 1: Visualization of range image and pole extraction result. On the top is the raw LiDAR scan. The corresponding range image generated from this scan is in the middle. The bottom is the pole extraction result based on range image.

The main contribution of this paper is a novel range image-based pole extractor for long-term localization of autonomous mobile systems. Instead of using directly the raw point clouds obtained from 3D LiDAR sensors, we investigate the use of range images for pole extraction. Range image is a light and natural representation of the scan from a rotating 3D LiDAR such as a Velodyne or Ouster sensors. Operating on such an image is considerably faster than on the raw 3D point cloud. Besides, range image keeps the neighborhood information implicitly in its 2D structure and we can use this information for segmentation. As shown in Fig. 1, in the mapping phase, we first project the raw point cloud into a range image and then extract poles from that image. After obtaining the position of poles in the range image, we use the ground-truth poses of the robot to reproject them into the global coordinate system to build a global map. During localization, we utilize Monte Carlo localization (MCL) for updating the importance weights of the particles by matching the poles detected from online sensor data with the poles in the global map.

In sum, we make three key claims: Our approach is able to (i) extract more reliable poles in the environment compared to the baseline method, (ii) as a result, achieve better localization performance in different environments, and (iii) run online without a GPU. These claims are backed up by the

paper and our experimental evaluation. The source code of our approach together with all parameters and the pole dataset can be accessed at: https://github.com/PRBonn/pole-localization.

II. Related Work

For localization given a map, there exists a large amount of research. While a lot of different types of sensors have been used to tackle this problem [22], in this work, we mainly concentrate on LiDAR-based approaches.

Traditional approaches to robot localization rely on probabilistic state estimation techniques. A popular framework is Monte Carlo localization [9], which uses a particle filter to estimate the pose of the robot and is still widely used in robot localization systems [5], [7].

Besides the traditional geometric-based methods, more and more approaches recently exploit deep neural networks and semantic information for 3D LiDAR localization. For example, Ma et al. [16] combine semantic information such as lanes and traffic signs in a Bayesian filtering framework to achieve accurate and robust localization within sparse HD maps, whereas Tinchev et al. [23] propose a learning-based method to match segments of trees and localize in both urban and natural environments. Sun et al. [21] use a deep-probabilistic model to accelerate the initialization of the Monte Carlo localization and achieve a fast localization in outdoor environments. Wiesmann et al. [26] propose a deep learning-based 3D network to compress the LiDAR point cloud, which can be used for large-scale LiDAR localization. In our previous work [5], [4], 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.

Instead of using dense semantic information estimated by neural networks [17], [15], a rather lighter solution has been proposed for long-term localization, which extracts only pole landmarks from point clouds. There are usually two parts in pole-based approaches, pole extraction and pose estimation. For pole extraction, Sefati et al. [20] first remove the ground plane from the point cloud and project the rest points on a horizontal grid. After that, they cluster the grid cells and fit a cylinder for each cluster. Finally, a particle filter with nearest-neighbor data association is used for pose estimation. Weng et al. [25] and Schaefer et al. [19] use similar particle filter-based methods to estimate the pose of the robot with different pole extractors. Weng et al. [25] discretize the space and extract poles based on the number of laser reflections in each voxel. Based on that, Schaefer et al. [19] consider both the starting and end points of the scan and thus model the occupied and free space explicitly. Kümmerle et al. [14] use a nonlinear least-squares optimization method to refine the pose estimation.

In contrast to the aforementioned approaches, we use a projection-based method and avoid the comparable costly processing of 3D point cloud data. Thus, our implementation is fast. Besides, our approach uses range information directly without exploiting neural networks or deep learning, and thus, it generalizes well to different environments and different LiDAR sensors and does not require new training data when moving to different environments.

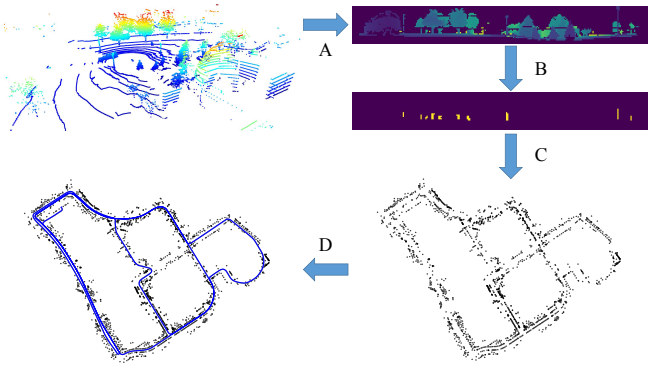


Fig. 2: Overview of our approach. A. we project the LiDAR point cloud into a range image and B. extract poles in the image. C. based on the extracted poles, we then build a 2D global pole map of the environment. D. we finally propose a pole-based observation model for MCL to localize the robot in the map.

III. Our Approach

In this paper, we propose a range image-based pole extractor for long-term localization using a 3D LiDAR sensor. As shown in Fig. 2, we first project the LiDAR point cloud into a range image (Sec. III-A) and extract poles from it (Sec. III-B). Based on the proposed pole extractor, we then build a global pole map of the environment (Sec. III-C). In the localization phase, we extract poles online using the same extractor, and use a novel pole-based observation model for Monte Carlo localization (Sec. III-D).

A. Range Image Generation

The key idea of our approach is to use range images generated from LiDAR scans for pole extraction. Following the prior work [6], [7], we utilize a spherical projection for range images generation. Each LiDAR point $\mathbf{p} = (x, y, z)$ is mapped to spherical coordinates via a mapping $\Pi : \mathbb{R}^3 \mapsto \mathbb{R}^2$ and finally to image coordinates, as defined by

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

where (u, v) are image coordinates, (h, w) are the height and width of the desired range image, $\mathrm{f} = \mathrm{f}_{\mathrm{up}} + \mathrm{f}_{\mathrm{down}}$ is the vertical field-of-view of the sensor, and $r = ||\mathbf{p}_i||_2$ is the range of each point. This procedure results in a list of (u, v) tuples containing a pair of image coordinates for each $\mathbf{p}_i$, which we use to generate our proxy representation. Using these indices, we extract for each $\mathbf{p}_i$, its range r, its x, y, and z coordinates, and store them in the image.

B. Pole Extraction

We extract poles based on the range images generated in the previous step. The general intuition behind our pole extraction algorithm is that the range values of the poles are usually significantly smaller than the backgrounds. Based on this idea and as specified in Alg. 1, our first step is to cluster the pixels of the range image into different small regions based on their range values. We first pass through all pixels in the range image, from top to bottom, left to right.

We put all pixels with valid range data in an open set $\mathbf{O}$. To remove ground points, we ignore pixels under a certain height. For each valid pixel $\mathbf{p}$, we check its neighbors including the left, right and below ones. If there exists a neighbor with a valid value and the range differences between the current pixel and its neighbour is smaller than a threshold ϵ, we add the current pixel to a cluster set $\mathbf{c}$ and remove it from the open set $\mathbf{O}$. We do the same check iteratively with the neighbors until no neighbor pixel meets the above criteria, and we then get a cluster of pixels. After checking all the pixels in $\mathbf{O}$, we will get a set $\mathbf{C}$ with several clusters and each cluster represents one object. If the number of pixels in one cluster is smaller than a threshold ζ, we regard it as an outlier and ignore it.

The next step is to extract poles from these objects using geometric constraints. To this end, we exploit both the range information and the 3D coordinates (x, y, z) of each pixel. We first check the aspect ratio of each cluster. Since we are only interested in pole-like objects, whose height is usually larger than its width, we therefore throw away the cluster with aspect ratio $h/w<1$. Another heuristic we use is the fact that a pole usually stands alone and has a significant distance from background objects. N_{SmallR} is the number of points in cluster $\mathbf{c}$ whose range value is smaller than its neighbour outside $\mathbf{c}$, we throw away the cluster if N_{SmallR} is smaller than δ times the number of all points in the cluster.

To exploit 3D coordinates (x, y, z) of each pixel, we calculate $\max(z) - \min(z)$ of each cluster and only take a cluster as a pole candidate if $\max(z) - \min(z) > \gamma$. Besides, we are only interested in poles whose height is higher than α. Based on experience, we also set a threshold β for the lowest position of the pole to filter outliers. For each pole candidate, we then fit a circle using the x and y coordinates of all points in the cluster and get the center and the radius of that pole. We filter out the candidates with too small or too large radiuses and candidates that connect to other objects by checking the free space around them. After the above steps, we finally extract the positions and radiuses of poles.

C. Mapping

To build the 2D global pole map for localization, we following the same setup as introduced by Schaefer et al. [19], splitting the ground-truth trajectory into shorter sections with equal length. Since the provided poses are not very accurate for mapping [19], instead of aggregating a noisy submap, we only use the middle LiDAR scan of each section to extract poles. We merge multiple overlapped pole detections by averaging over their centers and radiuses and apply a counting model to filter out the dynamic objects. Only those poles that appear multiple times in continuous sections are regarded as real poles.

Algorithm 1: Pole Extraction Based on Range Image

Input: Range Image $\mathbf{I}_{range}$
Output: PoleParameters $\mathbf{P}$ with circle centers and radiuses

Let $\mathbf{O}$ be the set of all valid pixel coordinates in $\mathbf{I}_{range}$.
while $\mathbf{O} \neq \emptyset$ **do**
  create a new $\mathbf{c}$ in $\mathbf{C}$; $\mathbf{p} \leftarrow \mathbf{O}[0]$; $\mathbf{N} \leftarrow neighbor(\mathbf{p})$
  while $\mathbf{N} \neq \emptyset$ **do**
    if *any* $neighbor(\mathbf{p}) \in \mathbf{O}$ *and* $distance(neighbor(\mathbf{p}), \mathbf{p}) < \epsilon$ **then**
      $\mathbf{c} \leftarrow \mathbf{c} + \{\mathbf{p}\}$; $\mathbf{p} \leftarrow \mathbf{N}[0]$
      $\mathbf{N} \leftarrow \mathbf{N} + neighbor(\mathbf{p})$
    end
  end
  $N_{pixel} \leftarrow$ the number of pixels in $\mathbf{c}$
  if $N_{pixel} < \zeta$ **then**
    $\mathbf{C} \leftarrow \mathbf{C} - \mathbf{c}$
  end
end
foreach $\mathbf{c} \in \mathbf{C}$ **do**
  $w, h \leftarrow width(\mathbf{c}), height(\mathbf{c})$
  $N_{SmallR} \leftarrow$ the number of pixels in $\mathbf{c}$ whose range value is smaller than its neighbour outside $\mathbf{c}$
  if $h/w < 1$ *or* $N_{SmallR} < \delta \cdot len(\mathbf{c})$ **then**
    $\mathbf{C} \leftarrow \mathbf{C} - \mathbf{c}$
  end
end
foreach $\mathbf{c} \in \mathbf{C}$ **do**
  $\mathbf{x}, \mathbf{y}, \mathbf{z} \leftarrow$ 3D coordinates of pixels in $\mathbf{c}$
  if $\max(\mathbf{z}) > \alpha$ *and* $\min(\mathbf{z}) < \beta$ *and* $(\max(\mathbf{z}) - \min(\mathbf{z})) > \gamma$ **then**
    $x_{\mathbf{c}}, y_{\mathbf{c}}, r_{\mathbf{c}} \leftarrow FitCircle(\mathbf{x}, \mathbf{y})$
    $N_{Free} \leftarrow$ the number of the pixels in a small free space outside the radius of the pole
    if $r_{\mathbf{c}} < a$ *and* $r_{\mathbf{c}} > b$ *and* $N_{Free} < \mu \cdot len(\mathbf{z})$ **then**
      $\mathbf{P} \leftarrow \mathbf{P} + \{x_{\mathbf{c}}, y_{\mathbf{c}}, r_{\mathbf{c}}\}$
    end
  end
end

D. Monte Carlo Localization

Monte Carlo localization (MCL) is commonly implemented using a particle filter [9]. 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 $\mathbf{z}_{1:t}$ and motion controls $\mathbf{u}_{1:t}$ up to time t. This posterior is updated as follows:

$$p(\mathbf{x}_t \mid \mathbf{z}_{1:t}, \mathbf{u}_{1:t}) = \eta \; p(\mathbf{z}_t \mid \mathbf{x}_t) \cdot \int p(\mathbf{x}_t \mid \mathbf{u}_t, \mathbf{x}_{t-1}) \; p(\mathbf{x}_{t-1} \mid \mathbf{z}_{1:t-1}, \mathbf{u}_{1:t-1}) \; d\mathbf{x}_{t-1}, \quad (2)$$

where η is a normalization constant, $p(\mathbf{x}_t \mid \mathbf{u}_t, \mathbf{x}_{t-1})$ is the motion model, $p(\mathbf{z}_t \mid \mathbf{x}_t)$ 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 our case, each particle represents a hypothesis for the 2D pose $\mathbf{x}_t = (x, y, \theta)_t$ of the robot at time t. When the robot moves, the pose of each particle is updated based on a motion model with the control input $\mathbf{u}_t$ or the odometry

measurements. For the observation model, the weights of the particles are updated based on the difference between expected observations and actual observations. The observations are the positions of the poles. We match the online observed poles with the poles in the map via nearest-neighbor search using a k-d tree. The likelihood of the j-th particle is then approximated using a Gaussian distribution:

$$p\left(\mathbf{z}_t \mid \mathbf{x}_t\right) \propto \prod_{i}^{N} \exp\left(-\frac{1}{2}\frac{d\left(\mathbf{z}_t^i, \mathbf{z}_j^i\right)^2}{\sigma_d^2}\right), \tag{3}$$

where d corresponds to the difference between the online observed pole $\mathbf{z}_t^i$ and matched pole in the map $\mathbf{z}_j^i$ given the position of the particle j. N is the number of matches of current scan. We use the Euclidean distance between the pole positions to measure this difference. If the number of effective particles decreases below a threshold [12], the resampling process is triggered and particles are sampled based on their weights.

IV. Experimental Evaluation

The main focus of this work is an accurate and efficient pole extractor for long-term LiDAR localization. We present our experiments to show the capabilities of our method and to support our key claims, that our method is able to: (i) better extract poles compared to the baseline method, (ii) as a result, achieve better performance on long-term localization in different environments, and (iii) achieve online operation without using GPUs.

A. Pole Extractor Performance

The first experiment evaluates the pole extraction performance of our approach and supports the claim that our range image-based method outperforms the baseline method in pole extraction.

To the best of our knowledge, there is no specific public dataset available to evaluate pole extraction performance. To this end, we label the poles in session 2012-01-08 of NCLT dataset by hand and release this dataset for public research use. For the reason that the original NCLT ground-truth poses are inaccurate [19], the aggregated point cloud is a little blurry. Therefore, to create the ground-truth pole map of the environment, we partition the ground-truth trajectory into shorter segments of equal length. For each segment, we aggregate the point cloud together and use Open3D [27] to render and label the pole positions. We only label those poles with high certainty and ignore those blurry ones. Besides our own labelled data, we also reorganize the SemanticKITTI [1] dataset sequence 00-10 by extracting the pole-like objects, e.g. traffic signs, poles and trunk, and then clustering the point clouds to generate the ground-truth pole instances. We evaluate our method and compare it to the state-of-the-art pole-based LiDAR localization method proposed by Schaefer et al. [19] in both datasets.

During the matching phase, we find the matches via nearest-neighbor search using a k-d tree with $1\,m$ distance bounds. Tab. I summarizes the precision, recall and F1 score of our method and Schaefer et al. [19] compared to the ground-truth pole map on both NCLT dataset and SemanticKITTI dataset. As can be seen, our method has better performance and extracts more poles in both environments.

TABLE I: Pole Extraction Accuracy on NCLT and KITTI datasets.

Dataset	Method	Precision	Recall	F1 Score
NCLT	Schaefer [19]	0.518	0.664	0.582
	Ours	**0.525**	**0.713**	**0.605**
Semantic KITTI	Schaefer [19]	0.621	0.380	0.455
	Ours	**0.687**	**0.439**	**0.515**

B. Localization Performance

The second experiment is presented to support the claim that our approach achieves higher accuracy on localization in different environments. To assess the localization reliability and accuracy of our method, we use three datasets for evaluation, NCLT dataset [3], MulRan dataset [13], and KITTI Odometry dataset [11]. Note that, these three datasets are collected in different environments (U.S., Korea, Germany) with different LiDAR sensors (Velodyne HDL-32E, Ouster OS1-64, Velodyne HDL-64E). In the NCLT and MulRan dataset, the robot passes through the same place multiple times with month-level temporal gaps, hence ideal to test the long-term localization performance. We compare our methods to both pole-based method from Schaefer et al. [19] and the range image-based method proposed by Chen et al. [7]. For the KITTI dataset, we follow the setup introduced by Schaefer et al. [19] and compare our method with methods proposed by Schaefer et al. [19], Kümmerle et al. [14], Weng et al. [25] and Sefati et al. [20]. For all the experiments, we use the same setup as used in the baselines and report their results from the original work.

1) Localization on the NCLT Dataset: The NCLT dataset contains 27 sessions with an average length of $5.5\,km$ and an average duration of $1.3\,h$ over the course of 15 months. The data is recorded at different times over a year, different weather and seasons, including both indoor and outdoor environments, and also lots of dynamic objects. The trajectories of different sessions have a large overlap. Therefore, it is an ideal dataset for testing long-term localization in urban environments.

We first build the map following the setup introduced by Schaefer et al. [19], which uses the laser scans and the ground-truth poses of the first session. Since in later sessions, the robot sometimes moves into unseen places for the first session, we therefore also use those scans whose position is $10\,m$ away from all previously visited poses to build the map. During localization, we use 1000 particles and use the same initialization as [19] by uniformly sampling positions around the first ground-truth pose within a $2.5\,m$ circle. The orientations are uniformly sampled from -5 to 5 degree. We resample particles if the number of effective particles is less than 0.5. To get the pose estimation, we use the average poses of the best $10\,\%$ of the particles.

Tab. III shows the position and orientation errors for every session. We run the localization 10 times and compute the average means and RMSEs to the ground-truth trajectory. The

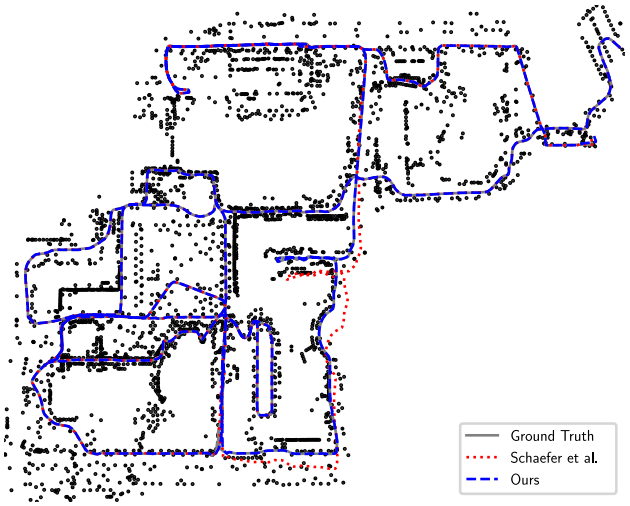


Fig. 3: Comparison of localization results of Schaefer et al. [19] and our method in session 2012-02-23 on NCLT dataset. The black dots are the poles on the map. The grey line is the ground-truth trajectory. The blue line is our result and the red one is of the baseline method. As can be seen, Schaefer's method loses the track of the robot in some places, while our method always tracks the correct robot poses with respect to the ground truth.

TABLE II: Localization Results on MulRan Dataset.

	Schaefer [19]	Chen [7]	ours
RMSE_{pos} [m]	1.82	0.83	**0.48**
RMSE_{ang} [°]	0.56	3.14	**0.27**

results show that our method surpasses Schaefer et al. [19] in almost all sessions with an average error of $0.173\,m$. Besides, in session 2012-02-23, the baseline method fails to localize resulting in an error of $2.470\,m$, while our method never loses track of the robot position (Fig. 3). This is because our pole extractor can robustly extract poles even in an environment where there are fewer poles.

2) Localization on the MulRan Dataset: In MulRan dataset, we use KAIST 02 sequence (collected on 2019-08-23) to build the global map and use KAIST 01 sequence (collected on 2019-06-20) for localization. Tab. II shows the location and yaw angle RMSE errors on MulRan Dataset. As can be seen, our method consistently achieves a better performance than both baseline methods [19], [7].

3) Localization on the KITTI Dataset: For the KITTI dataset, we follow the setup introduced in Schaefer et al. [19], where sequence number 9 data is used for both mapping and localization. Tab. IV shows the localization results. Our method consistently achieves a better performance than all four baseline methods.

C. Runtime

This experiment has been conducted to support the claim that our approach runs online at the sensor frame rate without using GPUs. We compare our method to the baseline method proposed by Schaefer et al. [19]. As reported in their paper, the baseline method takes an average of $1.33\,s$ for pole extraction on a PC using *a dedicated GPU*. We tested our method without using a GPU and our method only needs $0.09\,s$ for pole extraction and all MCL steps take less than $0.1\,s$ yielding a run time faster than the LiDAR frame rate of $10\,Hz$.

V. Conclusion

In this paper, we presented a novel range image-based pole extraction approach for online long-term LiDAR localization. Our method exploits range images generated from LiDAR scans. This allows our method to process point cloud data rapidly and run online. We implemented and evaluated our approach on different datasets and provided comparisons to other existing techniques and supported all claims made in this paper. The experiments suggest that our method can extract more poles in the environments accurately and achieve better performance in long-term localization tasks. Moreover, we release our implementation and pole dataset for other researchers to evaluate their algorithms. In the future, we plan to explore the usage of other features such as road markings, curb and intersection features, to improve the robustness of our method.

References

[1] J. Behley, M. Garbade, A. Milioto, J. Quenzel, S. Behnke, C. Stachniss, and J. Gall. SemanticKITTI: A Dataset for Semantic Scene Understanding of LiDAR Sequences. In *Proc. of the IEEE/CVF International Conf. on Computer Vision (ICCV)*, 2019.

[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] N. Carlevaris-Bianco, A. Ushani, and R. Eustice. University of Michigan North Campus long-term vision and lidar dataset. *Intl. Journal of Robotics Research (IJRR)*, 35(9):1023–1035, 2016.

[4] 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.

[5] 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.

[6] X. Chen, A. Milioto, E. Palazzolo, P. Gigure, 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.

[7] 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.

[8] I. Cvisic, J. Cesic, I. Markovic, and I. Petrovic. SOFT-SLAM: Computationally Efficient Stereo Visual SLAM for Autonomous UAVs. *Journal of Field Robotics (JFR)*, 35(4):578–595, 2018.

[9] 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.

[10] 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.

[11] A. Geiger, P. Lenz, and R. Urtasun. Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 3354–3361, 2012.

[12] 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.

[13] 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.

Session Date	f_{map} [%]	Δ_{pos} [m]		RMSE_{pos} [m]		Δ_{ang} [°]		RMSE_{ang} [°]	
		Schaefer [19]	Ours	Schaefer [19]	Ours	Schaefer [19]	Ours	Schaefer [19]	Ours
2012-01-08	100.0	0.130	**0.115**	0.178	**0.146**	0.663	**0.626**	0.857	**0.810**
2012-01-15	8.5	0.156	**0.146**	0.225	**0.204**	0.760	**0.750**	0.999	**0.982**
2012-01-22	5.1	0.172	**0.149**	0.222	**0.190**	0.939	**0.911**	1.291	**1.238**
2012-02-02	0.4	0.155	**0.136**	0.205	**0.172**	0.720	**0.699**	0.975	**0.922**
2012-02-04	0.1	0.144	**0.134**	0.195	**0.167**	0.684	**0.671**	0.903	**0.876**
2012-02-05	0.5	0.148	**0.138**	0.210	**0.206**	**0.690**	0.694	0.947	**0.937**
2012-02-12	0.8	0.269	**0.253**	1.005	**1.002**	0.802	**0.786**	1.040	**1.019**
2012-02-18	0.8	0.149	**0.131**	0.221	**0.175**	0.699	**0.676**	0.938	**0.905**
2012-02-19	0.0	0.148	**0.137**	0.194	**0.183**	0.704	**0.692**	0.944	**0.923**
2012-03-17	0.0	0.149	**0.137**	0.191	**0.174**	0.830	**0.798**	1.062	**1.031**
2012-03-25	0.0	0.200	**0.178**	0.262	**0.235**	1.418	**1.379**	1.836	**1.789**
2012-03-31	0.0	0.143	**0.135**	0.184	**0.176**	0.746	**0.729**	0.973	**0.936**
2012-04-29	0.0	0.170	**0.154**	0.251	**0.222**	0.829	**0.820**	1.079	**1.069**
2012-05-11	5.5	0.161	**0.132**	0.225	**0.163**	0.773	**0.747**	0.998	**0.965**
2012-05-26	0.4	0.158	**0.142**	0.217	**0.183**	0.690	**0.672**	0.889	**0.871**
2012-06-15	0.4	0.180	**0.145**	0.238	**0.186**	0.659	**0.646**	0.879	**0.874**
2012-08-04	0.3	0.210	**0.169**	0.340	**0.230**	0.884	**0.843**	1.143	**1.093**
2012-08-20	3.8	0.189	**0.156**	0.264	**0.207**	0.711	**0.688**	0.941	**0.906**
2012-09-28	0.3	0.206	**0.171**	0.311	**0.241**	0.731	**0.726**	0.952	**0.949**
2012-10-28	1.4	0.217	**0.185**	0.338	**0.281**	0.693	**0.680**	0.919	**0.909**
2012-11-04	2.5	0.257	**0.208**	0.456	**0.317**	0.746	**0.718**	0.996	**0.973**
2012-11-16	2.7	0.403	**0.296**	0.722	**0.435**	1.467	**1.403**	2.031	**1.919**
2012-11-17	0.4	0.243	**0.201**	0.377	**0.323**	0.686	**0.685**	0.959	**0.948**
2012-12-01	0.0	0.266	**0.226**	0.492	**0.429**	0.674	**0.665**	0.930	**0.887**
2013-01-10	0.0	0.217	**0.187**	0.278	**0.226**	0.689	**0.627**	0.911	**0.806**
2013-02-23	0.0	2.470	**0.236**	5.480	**0.567**	1.083	**0.592**	1.769	**0.846**
2013-04-05	0.0	0.365	**0.295**	0.920	**0.869**	0.654	**0.642**	**1.028**	1.036
Average		0.284	**0.174**	0.526	**0.293**	0.801	**0.761**	1.081	**1.016**

TABLE III: Results of our experiments with the NCLT dataset compared to Schaefer [19], averaged over ten localization runs per session. The variables Δ_{pos} and Δ_{ang} denote the mean absolute errors in position and heading, respectively, RMSE_{pos} and RMSE_{ang} represent the corresponding root mean squared errors, while f_{map} denotes the fraction of lidar scans per session used to build the reference map.

Approach	Δ_{pos} [m]	RMSE_{pos} [m]	Δ_{lat} [m]	σ_{lat} [m]	Δ_{lon} [m]	σ_{lon} [m]	Δ_{ang} [°]	σ_{ang} [°]	RMSE_{ang} [°]
Kümmerle et al. [14]	0.12	—	0.07	—	0.08	—	0.33	—	—
Weng et al. [25]	—	—	—	0.082	—	0.164	—	0.329	—
Sefati et al. [20]	—	0.24	—	—	—	—	—	—	0.68
Schaefer et al. [19]	0.096	0.111	**0.061**	0.075	0.060	0.067	0.133	0.188	0.214
Ours	**0.091**	**0.106**	0.066	**0.067**	**0.046**	**0.058**	**0.084**	**0.102**	**0.102**

TABLE IV: Comparison of the accuracies of Sefati et al. and Schaefer et al.'s method and the proposed localization approach on the KITTI dataset. The results of Weng et al. and Kümmerle et al. are not directly comparable and are stated for qualitative analysis only.

[14] J. Kümmerle, M. Sons, F. Poggenhans, T. Kühner, M. Lauer, and C. Stiller. Accurate and efficient self-localization on roads using basic geometric primitives. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, pages 5965–5971. IEEE, 2019.

[15] S. Li, X. Chen, Y. Liu, D. Dai, C. Stachniss, and J. Gall. Multi-scale interaction for real-time lidar data segmentation on an embedded platform. *arXiv preprint arXiv:2008.09162*, 2020.

[16] 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.

[17] 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.

[18] R. Mur-Artal, J. Montiel, and J.D. Tardos. ORB-SLAM: a versatile and accurate monocular SLAM system. *IEEE Trans. on Robotics (TRO)*, 31(5):1147–1163, 2015.

[19] 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.

[20] M. Sefati, M. Daum, B. Sondermann, K.D. Kreisköther, and A. Kampker. Improving vehicle localization using semantic and pole-like landmarks. In *Proc. of the IEEE Vehicles Symposium (IV)*, pages 13–19. IEEE, 2017.

[21] 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.

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

[23] 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.

[24] 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.

[25] L. Weng, M. Yang, L. Guo, B. Wang, and C. Wang. Pole-based real-time localization for autonomous driving in congested urban scenarios. In *Proc. of the Intl. Conf. on Real-time Computing and Robotics (RCAR)*, pages 96–101. IEEE, 2018.

[26] 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.

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