2020 5th International Conference on Advanced
Robotics and Mechatronics (ICARM)

Pole-based Localization for Autonomous Vehicles in Urban Scenarios Using Local Grid Map-based Method

Fan Lu[1], Guang Chen[1], Jinhu Dong[1], Xiaoding Yuan[1], Shangding Gu[2] and Alois Knoll[3]

***Abstract*— Self-localization is a key component of autonomous vehicles in urban scenarios. In this work, we proposed a localization system which is based on pole-like objects such as trees and street lamps. Pole-like objects are extracted from 3D LiDAR point cloud using a cluster-based method. Based on the pole detection results, we propose a new map representation which consists of numerous local grid maps. In order to tackle the data association problem caused by the ambiguity of pole-like landmarks, the detected poles are directly transformed to the local grid map to define a cost function without pole-to-pole matching. The subsequent non-linear optimization method is utilized to minimize the cost function and generate the vehicle pose. We evaluate our localization system on our self-collected dataset. And the proposed system achieves a root mean square error of less than 18 cm for position and less than 0.52 ° for yaw.**

I. Introduction

Autonomous vehicles have gained great attention over the few years. Among all the fundamental components (e.g., perception, decision making, motion planning and localization) in the field of autonomous vehicle, localization is one of the most important and challenging problems.

The most common method to obtain the position of autonomous vehicle is using Global navigation satellite system (GNSS). And the performance can be further improved via fusion with Inertial navigation system (INS) [1]. However, the accuracy of low-cost GPS is meter-level [2], which can not meet the requirements of autonomous vehicles and high accurate GNSS systems with centimeter-level accuracy like RTK-GPS is extremely expensive. Moreover, GNSS signal is not stable in urban scenarios due to the signal block.

Map-based method is a choice to perform high-accuracy localization. There are two types of maps: dense maps [3]–[5] and landmark maps [6]–[13]. Dense maps like point cloud map or grid map require a large amount of memory to store. As a result, the size of the map will quickly be unacceptable in large scale environments. Comparatively, landmark maps are more compact and requires much less memory resource. In urban scenarios, pole-like objects such as trees and street lamps are suited to be used as landmarks due to their commonality, distinction and stability [6]–[11]. However, an obvious drawback of pole-like objects is that they lack distinctive features to perform data association. Most of the existing works utilize nearest neighbor search to perform pole-to-pole matching, which requires that the detected poles are close enough to the poles in the pre-built map. Nevertheless, the prerequisite will not be satisfied when the vehicle movement is slightly large or the poles are extremely dense and the algorithm can easily fail due to mismatching.

In this work, we presented a 2D pole-based vehicle localization system in urban environments. In order to solve the mentioned data association problem, we do not perform a pole-to-pole matching. Instead of that, we propose a new pole map representation which consists of numerous local grid maps. Each cell in the grid map contains a value which is relative to its distance to the closest pole in the map. The detected poles are projected into the local grid map frame based on the current vehicle pose and a cost function is defined to evaluate the matching between the detected poles and the local grid map. Then the cost function is optimized using non-linear optimization method to obtain the vehicle pose. Compared to the map with discrete locations of poles, the proposed map is smoother. Moreover, the cost function considers the overall matching results rather than pole-to-pole matching, which is more robust to misdetected poles and avoids the problem of mismatching.

The proposed localization system is evaluated in the self-collected dataset and shows a great performance. Our main contributions are:

1) A new landmark map representation which consists of numerous local grid maps is proposed for pole-based localization.
2) A cost function is defined based on the local grid map to perform the pose optimization.

II. Related Works

Vehicle localization using LiDAR can be divided into two categories: point cloud map-based and landmark-based. More relevant to this work is landmark-based localization methods, especially pole-like landmark-based methods. We will briefly review landmark-based localization methods in this section.

A. Pole-like Landmark-based Methods

Most of the existing methods utilize particle filter(PF) or kalman filter(KF) to perform Pole-like landmark-based localization, the main difference of these methods is their pole detection methods. [8] voxelizes the space to numerous voxels and then seeks out the lowest and highest grids that has a density larger than a threshold. If the height satisfies the condition and the point cloud within the entire height range satisfies the density condition, the points are considered as

Guang Chen is the corresponding author.

This work was supported by the Youth Program of National Natural Science Foundation of China (Grant No. 61906138) and Shanghai AI Innovation Development Program 2018.

[1]Tongji University, Shanghai, China; [2]Wuhan University of Technology, China; [3]Technical University of Munich, Munich, Germany.

978-1-7281-6479-3/20/$31.00 ©2020 IEEE

pole points. The data association in [8] is performed via nearest neighbor search and the localization framework is based on particle filter. In [6], the input point cloud is projected onto a horizontal grid and the neighboring cells are clustered based on occupancy and height, then a cylinder is fitted. Similar to [8], the authors also use nearest neighbor search to solve the data association problem, and use particle filter to implement the localization framework. [7] is the first one to detect the poles not only consider the point cloud, but also the free space in between the laser sensor and the laser endpoints. They also use nearest neighbor search and particle filter. [9] uses also poles as landmarks. However, the author extract poles from stereo camera images rather than LiDAR point cloud. Different from the above methods, [10] performs an optimization-based method like our system. Besides poles, they also utilize facades and road markings as the landmark. In order to solve the data association problem, they accumulating the landmarks detected during the past timesteps to a local map and project the local map to the global map to identify the most probable correspondence hypothesis. [11] use local pole patterns to solve the data association problem. One local pole pattern contains several poles and is more robust than single pole-to-pole matching.

B. Other Landmark-based Methods

Besides poles, there are also other landmarks such as road markings and curbs can be utilized in vehicle localization. [12] uses road markings as landmark to perform their particle filter-based localization method. [14] utilizes road markings and road borders as landmark. The Monte Carlo vehicle localization method in [13] is based on curbs and intersections.

III. Approach

A. Overview

The architecture of the pole-based localization system is shown in Fig. 1. The localization system is divided into a mapping process and a localization process. During the mapping process, the detected poles are registered to a global map via a given ground-truth vehicle pose. The global map is further converted to numerous local grid maps. Compared to the map with discrete points, the proposed map is smoother and more robust to misdetected poles. During the localization process, we utilize an optimization-based method instead of traditional filter-based methods. Local grid map search is first performed to find the closest local grid map. The detected poles are transformed into the local grid map frame based on the current vehicle pose, and a cost function is defined. Then subsequent non-linear optimization method is utilized to optimize the cost function and generate vehicle pose.

B. Pole Detection

The pole detection method is based on the approach in [15]. The method consists of three steps: voxelization, horizontal cluster and vertical cluster.

TABLE I
Parameters of pole detection

x_{res}	y_{res}	z_{res}	t_v	t_{s1}	t_{s2}	H_{min}	t_r
0.2m	0.2m	0.2m	5	15	3	1.0m	1.5

1) Voxelization: The ground plane is removed first and the input point cloud is restrict in a certain spatial range. Then the point cloud is voxelized to numerous voxels to make the input data structured. The resolution of three directions of the voxel grid are $(x_{res}, y_{res}, z_{res})$. If the number of points in the voxel is greater than a certain threshold t_v, the voxel is marked as valid, otherwise invalid.

2) Horizontal cluster: The 3D grid is further divided into several horizontal layers in its vertical direction. The horizontal sections of poles have several obvious features: the area is within a certain range and isolation. We first simply aggregate all the connected valid voxels to a segment. And the segment is kept if the number of voxels in the segment is smaller than a threshold t_{s1}. Then the isolation of the segment is checked. For each segment, we selected two boxes centered on the segment center. Based on the isolation, all voxels of the segment should be included in the smaller box and there should be no valid voxels between the smaller and the larger boxes. However, we find that given a certain threshold t_{s2} of the number of valid voxels between two boxes would make the method more robust. Segments that pass the above checks are marked as valid.

3) Vertical cluster: After horizontal cluster, several valid horizontal segments are obtained. Then these segments would be further clustered in the vertical direction. Similar to horizontal cluster, connected valid segments are aggregated to a cluster. There may be discontinuities in vertical direction because 3D LiDAR is relatively sparse in its vertical direction. So the vertical search range is expanded to two layers above and below. And the vertical search range should be adjusted according to the number of scan lines of the LiDAR. Then the height and the height to width ratio of the vertical cluster is checked. If the height is larger than H_{min} and the ratio is larger than t_r, the cluster is considered as a pole.

Generally the point cloud of the pole should be fitted with a cylinder model. However, the fitting can be difficult and unstable even with help of RANSAC due to the presence of outliers. So we directly take the middle layer of the point cloud of the detected pole in the vertical direction and calculate the mean position of the points as the position of the pole.

The values of the parameters mentioned above for Velodyne VLP-16 are summarized in Table I. The parameters should be adjusted according to the specific point cloud. Several examples of the results of pole detection on our self-collected dataset can be seen in Fig. 2.

C. Mapping

After detecting the poles, the 2D locations of the centers of the poles are calculated and stored. Given the ground-truth

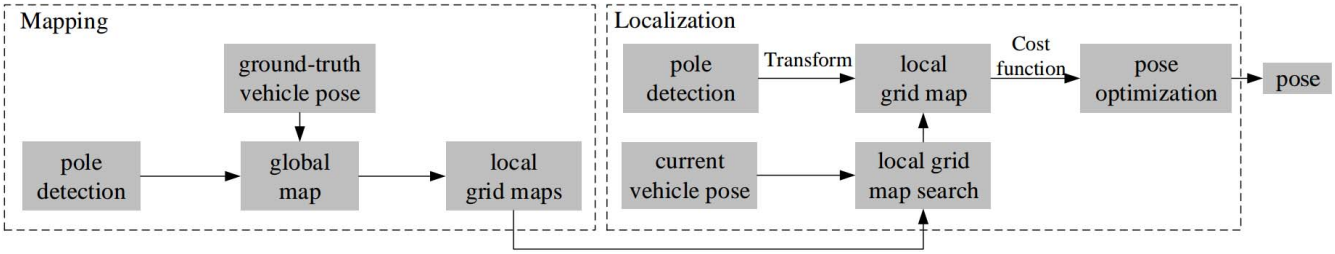


Fig. 1. The overview architecture of the pole-based localization system.

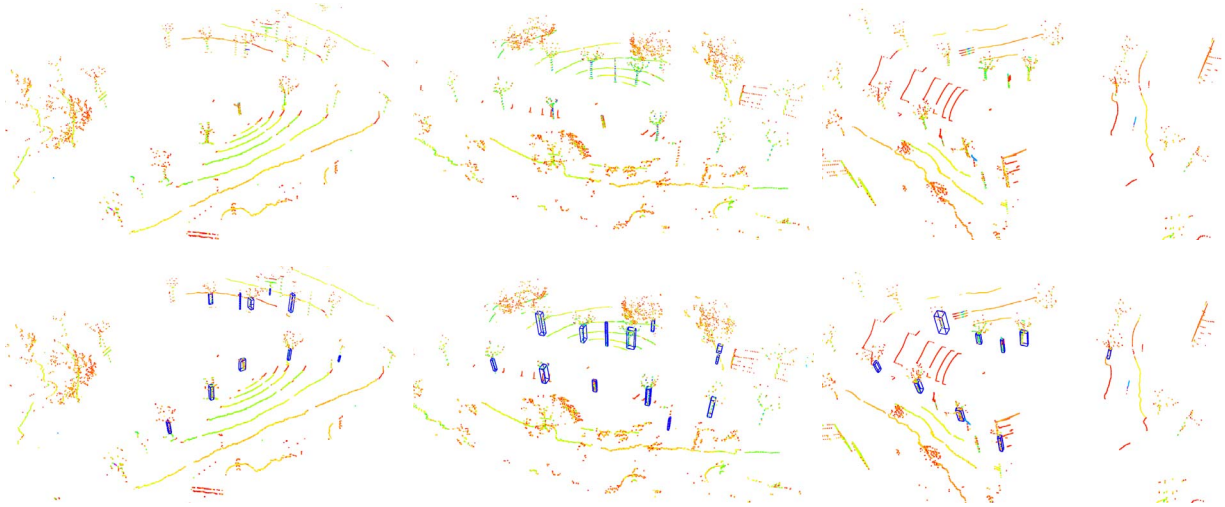


Fig. 2. The results of pole detection on the self-collected dataset. The first row is the input point cloud and the ground plane has been removed here. The second row is the detection results, the blue boxes is the detected bounding boxes of poles. The detection algorithm basically detects all poles in the point cloud (best viewed in color).

pose of the vehicle, the 2D locations is further transformed into the world coordinate frame. After a long trajectory, a global map with numerous points is created, and each point represents the location of poles. Due to the uncertain of vehicle pose and the detection results, the locations of the same pole on different vehicle poses would not be exactly the same. So a DBSCAN-based clustering method is utilized to obtain clusters of pole locations. And the center of each cluster is considered as the actual location of a single pole. Generally, the mapping process ends here and we will get a map consists of a number of discrete points. Under this form of map, the pole-to-pole matching is difficult because of the lack of distinctive features. Previous works usually implemented it by performing nearest neighbor search. As mentioned before, this method is unreliable when the vehicle has a large displacement or the poles are extremely dense. Misdetected poles or mismatched pairs can have a large impact on the localization results. And the match between single poles does not make good use of the geometric relationship between the locations of poles. Based on the above considerations, we propose a new map representation. A typical characteristic of vehicle movement is that it moves on the road. As a result, most of the regions on the global map make no sense. Only the landmarks around the road should be focused on. So we break the global map into many discrete local maps along the vehicle path (i.e., along the road), each local map has a location (x_m, y_m, θ_m). The distance between each local map should be chose according to the density of poles. And the distance should not be too large to ensure that there are sufficient overlaps between the detected poles and the local map even when the vehicle matched to a wrong local map because of a large movement. And the spatial size of the local map should be same as the receptive field of LiDAR. The local map is further converted to a grid map. The choice of resolution of the grid map should consider both accuracy and efficiency. It is worthy noting than the localization result can be more precise than the resolution of grid map under the optimization-based framework. The value of each cell in the grid is related to its distance to the closest pole. The value can be also considered as the probability that the cell contains pole. Suppose there are N poles in the local grid map and d_i represents its distance to the $i-th$ pole, the calculation equation of the value of each cell can be seen in Eq. 1.

$$value = \max\{\frac{1}{1+\alpha d_1}, ..., \frac{1}{1+\alpha d_N}\} \tag{1}$$

The 1 in the denominator is to ensure that the value will be not larger than 1. α is a coefficient to control the decay rate of the values. Another explanation is that α reflects the uncertainty of pole detection and $\alpha = 4$ in our implementation. Intuitively, the closer the cell to the closest

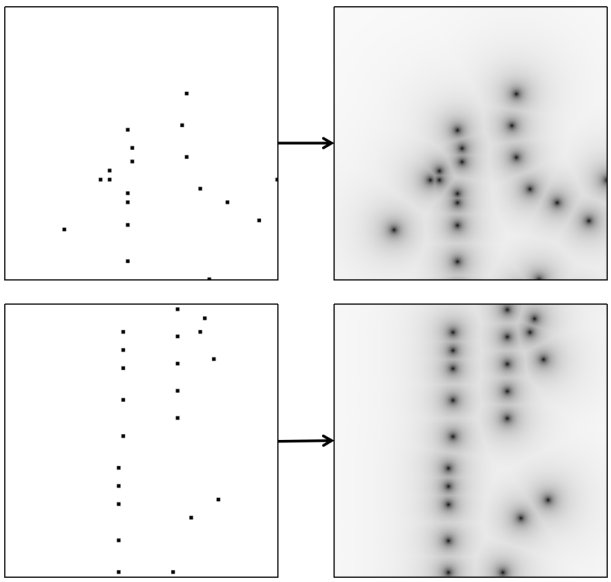


Fig. 3. Two examples of local grid maps. The left column is the original map and the black points represent the locations of poles. The right column is our local grid map, the gray value of the local grid map represents the value of the cell. The farther the cell from the pole, the lighter the color, which represents that the smaller the value of the cell. Visually, the right column is smoother than the left column and also reflects the uncertainty of pole detection.

pole, the larger the value of the cell and the cell contains a pole has a value of 1. Two examples of local grid maps are shown in Fig. 3.

D. Localization

The vehicle pose is initialized using GPS. After initialization, our localization system can run without GPS signal at all. Given a pre-built map, we first perform local grid map search to find the closest local grid map based on the current vehicle pose. And the detected poles are transformed into the local grid map frame with the vehicle pose and the local grid map location. After that, an interpolation is performed to obtain the corresponding values in the local grid map and a cost function is defined based on the values. At last, the new vehicle pose can be obtained by optimizing the cost function. The details are below.

1) Local grid map search: As mentioned before, the pre-built map is represented as numerous discrete local grid maps. The vehicle pose relative to the world coordinate frame is defined as (t_x, t_y, θ), where (t_x, t_y) is the 2D location and θ is the yaw angle. Given the current pose (t_x, t_y, θ), a nearest neighbor search is performed to find the closest local grid map. Even if the vehicle movement is large and a wrong local grid map is matched to the current vehicle pose, the overlap of the current detections and the local map is still sufficient to perform optimization.

2) Poles transform: Denote the location of the pole in vehicle frame as (x_p, y_p), the current vehicle pose as (t_x, t_y, θ), and the location of the local grid map as (x_m, y_m, θ_m), then the vehicle pose relative to the local grid map can be calculated as $(t_{xrel}, t_{yrel}, \theta_{rel}) = (t_x - x_m, t_y - y_m, \theta - \theta_m)$. The location of the pole (x_{pm}, y_{pm}) in the local grid map frame can be easily calculated based on the transformation.

3) Interpolation: After transforming the detected poles into the local grid map frame, the next step is to obtain the value of the local grid map on the transformed location. An obvious way is to round the position directly and query it in the local grid map to get the corresponding value. However, this method is not smooth and the localization accuracy will be severely limited by the resolution of the local grid map. Moreover, the round operation is not convenient to perform non-linear optimization because the gradient is hard to express. In order to solve the problem, we utilize bicubic interpolation like [16]. The bicubic interpolation is easy to implement because Ceres [17] has already integrated the interpolation method.

4) Pose optimization: We represent the pose estimation as a non-linear optimization problem and the problem is solved using Ceres [17]. The key of the optimization problem is the cost function. We denote the location of the $i-th$ detected poles in the local grid map frame as X_i and the interpolated value as $f(X_i)$. The definition of local grid map ensures $f(X_i) \leq 1$. The cost function is defined as Eq. 2. Note that we do not use the quadratic form cost function like [16] here. We tested several forms of cost function such as quadratic form $(\sum_{i=1}^{N}(1-f(X_i))^2)$ and root form $(\sum_{i=1}^{N}\sqrt{1-f(X_i)})$. The results shows that the proposed cost function achieves better performance. Intuitively, when the transformed location of the detected poles is closer to the poles in the pre-built map, the corresponding value in the local grid map is larger so the cost function is closer to 0. When the cost function is 0, the current detections are exactly coincides with the local grid map. In this form of map, detection errors and map errors are allowed and the vehicle pose will converge to a condition with the best overall matching with these errors. As a result, the vehicle pose can be obtained by optimizing the cost function. Although the resolution of the local grid map is limited, the optimization results can be more accurate than the resolution.

$$\sum_{i=1}^{N}(1-f(X_i)) \tag{2}$$

As mentioned before, it is difficult to perform pole-to-pole matching because they don't have distinctive signatures like SIFT for visual points. Generally, nearest neighbor search is performed to solve the data association problem. However, the method is not robust when the vehicle has a large movement or the poles are too dense. For example, when the current detected pole is exactly in the middle of two poles in the map or even more close to the wrong one because of a large movement in this frame, the nearest neighbor search would give out a wrong matching, which has a serious impact on the result. Comparatively, the proposed method consider the matching from an overall perspective rather the pole-to-pole matching. As a result, the proposed method will not suffer from mismatching problem and is also more robust to misdetected poles as long as sufficient poles are detected.

IV. Experiments

In order to evaluate the proposed localization system, we perform several experiments in the Jading campus of Tongji University in Shanghai. The point cloud is collected using a Velodyne VLP-16 LiDAR sensor, which is mounted on the roof of the vehicle. The ground truth pose is obtained using the LiDAR odometry algorithm: LeGO-LOAM [18]. It is worth noting that the short-term accuracy of LeGO-LOAM is guaranteed, and the cumulative error has little effect on the evaluation of our localization system. All the algorithms is implemented with help of ROS (Robot Operating System) framework. The pole detection module is implemented using PCL [19] toolkit and the pose optimization module is implemented with the help of Ceres [17]. In all our tests, we did not use data from sensors other than the LiDAR and the initial pose of the vehicle is supposed to be known by default. The computing platform is a PC with an Intel core i7-9700K CPU at the clock speed of 3.60 GHz. The average computing time for pose optimization is about 15ms, which depends on the pose difference between two consecutive frames. Intuitively, the larger the relative distance, the more steps it takes to converge to the optimal pose, which results in a longer time. Notice that we do not use GPS and IMU in our test, the computing time for pose optimization can be shorter if a better initial pose is given from the fusion of GPS and IMU.

As shown in Fig. 4, four sequences are selected to evaluate our method. The four sequences represent four different shaped trajectories: straight, small angle turn, right-angle turn and continuous turn. It is obvious that the curves of ground truth pose and the curves of localization results roughly coincide. Thanks to the proposed map representation, the proposed method achieves good performance even when the poles are extremely dense (e.g., the corner of Sequence 3). Besides, the system also demonstrates its robustness when the poles are relative sparse (e.g., Sequence 4). For more accurate analysis of system performance, we respectively calculate the mean absolute errors of longitudinal, lateral, position and yaw.

We also calculate Root-Mean-Square Error (RMSE) for longitudinal (denoted as $\mathrm{RMSE}_{\mathrm{lon}}$), lateral direction (denoted as $\mathrm{RMSE}_{\mathrm{lat}}$), position (denoted as $\mathrm{RMSE}_{\mathrm{pos}}$) and yaw (denoted as $\mathrm{RMSE}_{\mathrm{yaw}}$). RMSE simultaneously measures the accuracy and robustness of the system, which is used as the final evaluation standard. As shown in Table II, the proposed system shows a good performance in all the four sequences. The performance in Sequence 4 is the worst and the RMSE of yaw reached 0.516. The reason may be that the vehicle turns many times and the poles are sparse in this sequence. Except for Sequence 4, the proposed localization system achieves a RMSE of less than 0.18 m for position and less than 0.35 $^\circ$ for yaw. It is worth noting that the difference between RMSE and the mean absolute error of the proposed system is relatively small, which means that the number of outliers in the localization results is small, which also demonstrates the good robustness.

V. Conclusion and Future Work

In this paper we propose a pole-based vehicle localization system in urban scenarios. The main contribution is a new landmark map representation, which gives a new idea of data association of landmark matching. The new map representation enables us to perform an overall matching instead of pole-to-pole matching. Moreover, the map representation can be extended to other landmarks, such as road marking. The evaluation on the self-collected dataset has shown the accuracy and robustness of our system. The localization system achieves a RMSE of less than 0.18 m for position and 0.52 $^\circ$ for yaw.

However, the proposed system have several aspects can be further improved. First, the proposed system is basically a local optimization algorithm, which requires a good initialization for pose estimation. As the result, GPS and IMU data should be integrated into the entire system to improve the efficiency and accuracy. Then, other landmarks can be introduced to improve the performance such as curbs or road markings. Third, the proposed system can be extended to a SLAM (Simultaneous localization and mapping) system by introducing matching and pose estimation between frames.

References

[1] F. Caron, E. Duflos, D. Pomorski, and P. Vanheeghe, "Gps/imu data fusion using multisensor kalman filtering: introduction of contextual aspects," *Information fusion*, vol. 7, no. 2, pp. 221–230, 2006.

[2] M. Modsching, R. Kramer, and K. Ten Hagen, "Field trial on gps accuracy in a medium size city: The influence of built-up," in *3rd workshop on positioning, navigation and communication*, 2006, pp. 209–218.

[3] J. Levinson and S. Thrun, "Robust vehicle localization in urban environments using probabilistic maps," in *2010 IEEE International Conference on Robotics and Automation*, 2010, pp. 4372–4378.

[4] K. Yoneda, C. Yang, S. Mita, T. Okuya, and K. Muto, "Urban road localization by using multiple layer map matching and line segment matching," in *2015 IEEE Intelligent Vehicles Symposium (IV)*, 2015, pp. 525–530.

[5] L. Li, M. Yang, L. Guo, C. Wang, and B. Wang, "Hierarchical neighborhood based precise localization for intelligent vehicles in urban environments," *IEEE Transactions on Intelligent Vehicles*, vol. 1, no. 3, pp. 220–229, 2016.

[6] M. Sefati, M. Daum, B. Sondermann, K. D. Kreisköther, and A. Kampker, "Improving vehicle localization using semantic and pole-like landmarks," in *2017 IEEE Intelligent Vehicles Symposium (IV)*, 2017, pp. 13–19.

[7] 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 *2019 European Conference on Mobile Robots (ECMR)*, 2019, pp. 1–7.

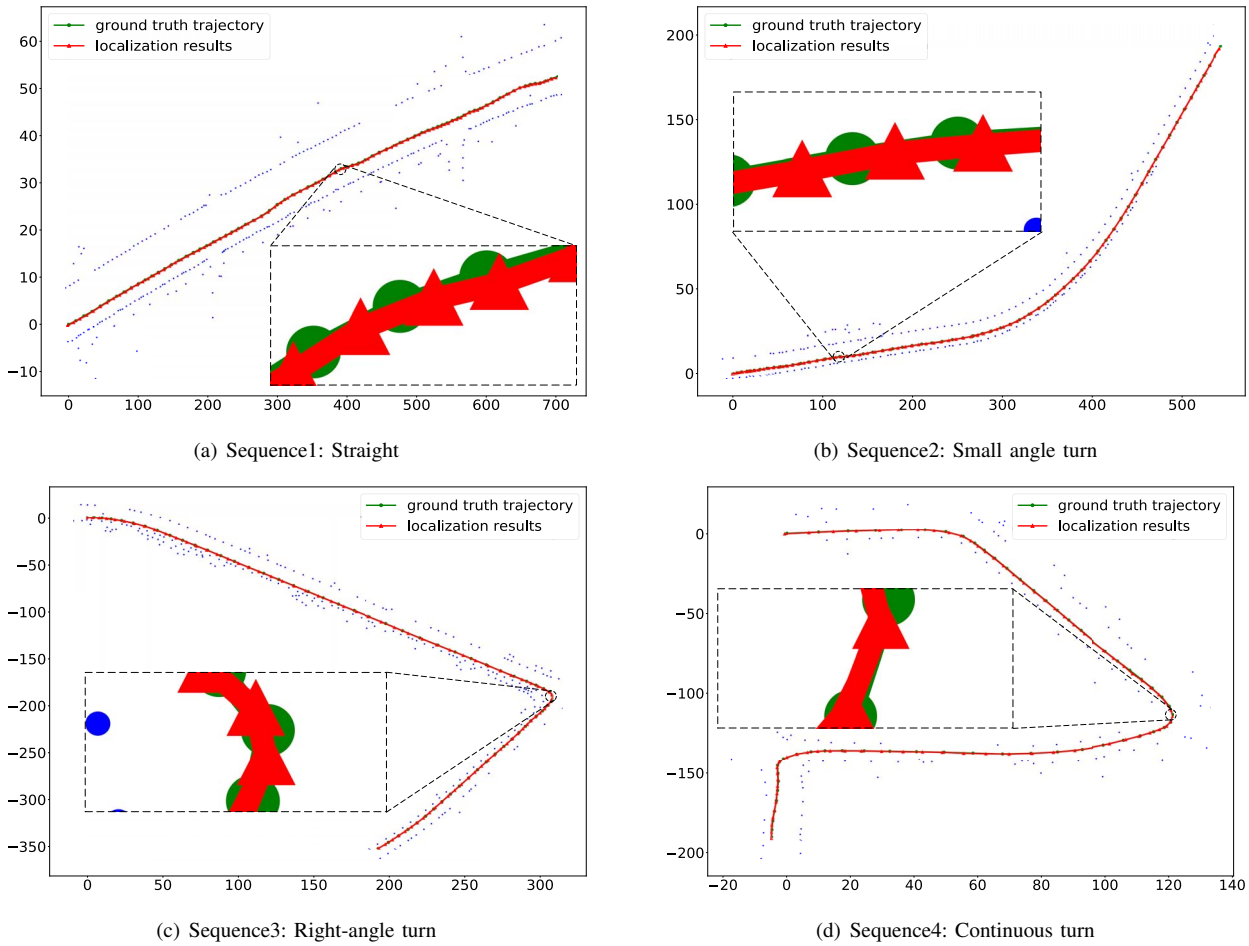
[8] L. Weng, M. Yang, L. Guo, B. Wang, and C. Wang, "Pole-based real-time localization for autonomous driving in congested urban scenarios," in *2018 IEEE International Conference on Real-time Computing and Robotics (RCAR)*, 2018, pp. 96–101.

[9] R. Spangenberg, D. Goehring, and R. Rojas, "Pole-based localization for autonomous vehicles in urban scenarios," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2016, pp. 2161–2166.

[10] 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 *2019 International Conference on Robotics and Automation (ICRA)*, 2019, pp. 5965–5971.

[11] C. Brenner, "Global localization of vehicles using local pole patterns," in *Joint Pattern Recognition Symposium*, 2009, pp. 61–70.

[12] M. Schreiber, C. Knöppel, and U. Franke, "Laneloc: Lane marking based localization using highly accurate maps," in *2013 IEEE Intelligent Vehicles Symposium (IV)*, 2013, pp. 449–454.



(a) Sequence1: Straight

(b) Sequence2: Small angle turn

(c) Sequence3: Right-angle turn

(d) Sequence4: Continuous turn

Fig. 4. Four trajectories in the Jiading campus of Tongji university. The blue points represent the poles in the global map. The red line is the localization results and the green line is the ground truth vehicle trajectory.

TABLE II

THE LOCALIZATION PERFORMANCE OF PROPOSED LOCALIZATION SYSTEM

Sequence	Δ_{lon}(m)	$RMSE_{lon}$(m)	Δ_{lat}(m)	$RMSE_{lat}$(m)	Δ_{pos}(m)	$RMSE_{pos}$(m)	Δ_{yaw}(deg)	$RMSE_{yaw}$(deg)
1	0.127	0.149	0.08	0.09	0.160	0.175	0.223	0.332
2	0.104	0.120	0.100	0.106	0.150	0.160	0.206	0.287
3	0.088	0.112	0.103	0.122	0.160	0.166	0.189	0.252
4	0.092	0.117	0.115	0.131	0.163	0.176	0.281	0.516

[13] B. Qin, Z. Chong, T. Bandyopadhyay, M. H. Ang, E. Frazzoli, and D. Rus, "Curb-intersection feature based monte carlo localization on urban roads," in *2012 IEEE International Conference on Robotics and Automation*, 2012, pp. 2640–2646.

[14] F. Poggenhans, N. O. Salscheider, and C. Stiller, "Precise localization in high-definition road maps for urban regions," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018, pp. 2167–2174.

[15] C. Cabo, C. Ordoñez, S. García-Cortés, and J. Martínez, "An algorithm for automatic detection of pole-like street furniture objects from mobile laser scanner point clouds," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 87, pp. 47–56, 2014.

[16] W. Hess, D. Kohler, H. Rapp, and D. Andor, "Real-time loop closure in 2d lidar slam," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, 2016, pp. 1271–1278.

[17] S. Agarwal, K. Mierle, and Others, "Ceres solver," http://ceres-solver.org.

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

[19] R. B. Rusu and S. Cousins, "3d is here: Point cloud library (pcl)," in *2011 IEEE international conference on robotics and automation*, 2011, pp. 1–4.