Accurate Localization of Autonomous Vehicles Based on Pattern Matching and Graph-Based Optimization in Urban Environments

Bingyi Cao, Claas-Norman Ritter, Daniel Göhring, and Raúl Rojas

***Abstract*— Accurate and reliable localization is a prerequisite for autonomous driving. Methods based on sparse landmarks, such as pole-like structures, have been widely studied because of their lower requirements for computing and storage. However, the number of landmarks of a single type is not always sufficient for reliable positioning. We propose a localization method using three different types of features in urban environments. The features we choose are poles, corners and walls which are persistent over time and can be reliably detected with LiDAR sensors. A pattern matching method for data association is introduced. Instead of using a filtering method, we adopt the graph-based optimization method to solve the pose estimation problem. Experiments conducted on two test roads show that the proposed method can provide accurate and reliable localization results in urban environments.**

I. INTRODUCTION

Autonomous vehicles require accurate and reliable localization especially driving in urban environments. Costly GNSS/INS navigation systems are limited by occlusions of the satellite signals, atmosphere changes, and multipath reflections. They cannot always guarantee accurate positioning in urban environments. In order to achieve this, a variety of alternative precise localization methods have been studied, involving sensors such as LiDARs, video cameras, radars, IMUs, or a combination thereof. In particular, multi-beam LiDARs provide fast and accurate measurements of the environment in form of three-dimensional point clouds. They are commonly used for ultrareliable localization and mapping in the field of autonomous driving.

In this paper, we present a localization algorithm based on sparse landmark features obtained with a 3D LiDAR sensor. LiDAR-based features vary in number, spatial distribution, and semantic level, depending on how they are selected. For example, a relatively large number of low-level geometric features can be chosen according to the local curvature or the gradients of surfaces. The detection of 3D objects, such as planes and poles, produces a sparser distribution of high-level features. Such 3D objects have the advantages of being sensor independent. Moreover, these kinds of features are convenient for manual checking of the detections and the correctness of the data association. They have lower storage and calculation requirements. Such features can be provided as part of the HD map of a city from third-party organizations or companies. Pole-like structures are one of the most frequently used features for localization of autonomous vehicles as they are stable across seasons and can be detected easily with a LiDAR system.

All Authors are with the Department of Mathematics and Computer Science, Freie Universität Berlin, Berlin, 14195, Germany

Several industrial companies and research institutes have collaborated to conduct automated and connected driving tests in Berlin. In a residential area for about one kilometer long in the test field, only a few pole-like objects such as streetlamps and trees can be found. They are not dense enough for reliable data association. This situation is common in residential areas outside of main roads. Therefore, we enrich the set of landmarks by adding fragments of walls and corners. Compared to other objects such as curbs, which are often occluded by parking cars, walls can be more reliably detected. The combination of walls, borders of windows and doors, can be detected as corners, which increases the total number of features at our disposal.

Regarding vehicle localization, filtering algorithms such as particle filters, or EKF, have been widely used for real time results. Particle filtering has disadvantages such as low efficiency and particle degeneracy. EKF relies on initial estimates for its linearization expansion, it has a relatively larger error when solving actual inherently nonlinear problems. We adopt the optimization method for pose estimation. Recent studies have explored factor graph-based methods for pose estimation and compared it to filtering methods [1].

The main contributions of this work are the following: (a) We provide a method for accurate and reliable localization for automated driving in urban environments. (b) The features we choose are three types of landmarks, which largely increase the number of features compared to solely pole-based methods. They can localize a vehicle in a residential area with few poles. (c) The results of our experiments demonstrate good repeatability and high accuracy.

II. RELATED WORK

In recent years, researchers have tackled the vehicle localization problem using pole-like landmarks extracted either from LiDAR sensors or cameras. Brenner et al. [2] discussed vehicle localization based on poles extracted from LiDAR data, and the automatic creation of maps of landmarks. In [3], a stereo camera system was used to detect pole-like landmarks for localization in urban scenarios. A particle filter was adopted to fuse wheel odometry and tracked poles to estimate the vehicle's pose. In [4], a pole-based localization method with local pattern matching was introduced to improve the correctness of the data association.

However, additional features such as planes, lane markings, and curbs can also be used for landmark-based localization in urban environments. The authors in [5] utilized camera-detected lane markings to assist in the localization process, and the position accuracy was improved. Study [6]

took advantages of reflection density information from laser scanners to detect various markings on the road surface, in conjunction with curb maps for vehicle positioning. In [7], the authors also used lane markings and curbs for localization, but the features were detected with a stereo camera. This study demonstrated accurate positioning result on rural roads.

Landmark-based localization and SLAM algorithms usually use the nearest neighbor method for data association, and it can be problematic when the initial position is inaccurate. Mismatches can also happen when the feature detection is noisy. One solution to this issue is to use the relations between local features for pattern matching. Previous research [8], [9] used the point pattern for data matching. Study [4] introduced a matching method using point and line features. Drawing from the related work, for this research, we use three different types of features, and therefore a new matching method is required.

Although different types of features have been studied. For data fusion and pose estimation, particle filters and EKF are most commonly used. The possibilities of using graph-based optimization for localization were explored in [10] and showed its increased robustness against outliers. In [1], factor graph-based localization using third-party maps for automated driving was recently studied, and it presented higher accuracy compared to particle filter methods.

III. SYSTEM OVERVIEW

In this section, we introduce the main parts of our proposed localization method in urban environments, including feature detection methods, *a priori* map creation, data association and pose estimation methods.

A. Feature Detection

The feature detection algorithm starts from generating range image from LiDAR measurements. The measurement points from the LiDAR sensor arrive in separate packages. We first need to eliminate the motion distortion while the car is moving. The wheel odometer provides movement information with a frequency of 100 Hz. It can be used to align LiDAR measured points. When a 360-degree scan is completed, the measurements are combined as a range image. Fig. 1 shows part of the range image after the motion distortion correction. The feature detection process can be outlined in two steps. First, the image is scanned line by line to find potential feature segments of pole, wall and corner. Feature segments of each type are stored in a segment buffer for follow-up processing. The second step is to merge the buffered segments into features.

Fig. 1. A segment of LiDAR Image after motion distortion correction. It has 64 pixels in vertical.

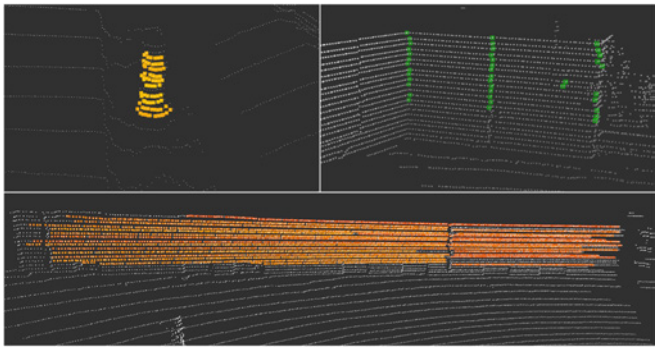

Fig. 2. (a) A potential pole segment, (b) a wall and corner segment.

In order to find a pole segment, we search for a set of measurement points is distinguished from the background detections. To find the potential wall segments, we intuitively search for points with similar distance. A sliding search window is adopted to tolerate outliers. In the wall segment finding process, we search for potential points of corners with large curvature. Therefore, only those corners located on the potential wall can be detected. Fig. 2 shows examples of pole, corner and wall segments.

Once the horizontal scan is complete, the segments are put into a histogram buffer according to their horizontal detection angle. Take pole segments as an example: if several segments have a close detection angle, we try to vertically merge them together. The center and the covariance of these points can be calculated. We then compute the eigenvalue decomposition of its covariance matrix. The corresponding eigenvector of the largest eigenvalue represents the pointing direction of the pole. For walls we project the points of the wall segments to the ground plan and then calculate the corresponding eigenvalues and the eigenvectors. The eigenvector corresponding to the largest eigenvalue represents the direction of the wall, and the second largest eigenvalue represents the expansion of these points in another direction. If the second value is small, it means that the points are mainly expanding in one direction on the projection plane, and the wall is perpendicular to the ground. Corners are relatively simple to calculate. If a set of corner segment points have close positions on the ground

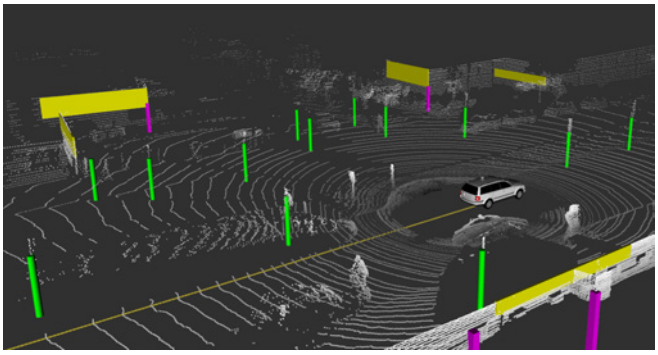

Fig. 3. Detected features, green cylinders are poles, yellow planes are segments of walls and magenta columns are corners.

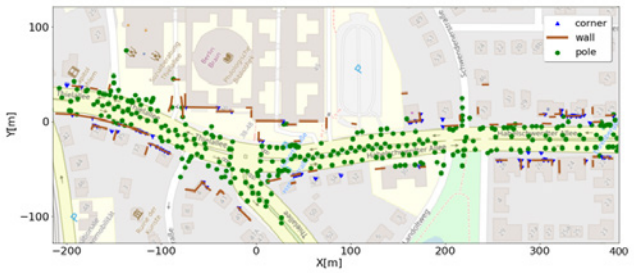

Fig. 4. Feature map of Thielallee, there are plenty of pole-like features on this street.

Fig. 5. "SAFARI" test field for automated and connected driving in Berlin, the main road is about 5.6 km. The zoomed in view shows the area where only a few poles exist but plenty of corners and walls can be detected.

plane, they basically form a corner. Fig. 3 shows the detected features: green cylinders are pole-like features, yellow planes are wall segments, and magenta columns are corners.

B. Feature Map

The localization algorithm needs a high precision *a priori* map for feature matching. The quality of this map directly affects the accuracy of the localization. Our test vehicle is equipped with an Applanix DGPS/INS navigation system, which can provide global position for localization and mapping. However, it can not always provide accurate data to create consistent maps. We adopted a SLAM method to produce our feature-based *a priori* map. The off-line batch solution globally minimizes motion and observation errors and creates a self-consistent map. We implemented a graph SLAM with loop closure aided by the Applanix navigation system as it provides highly accurate angular and linear velocity. Previous research [11] presented a GNSS/INS aid SLAM method to map trees, which gives inspiration for our work. We adopted the graph-based optimization framework g2o [12] in this study. For feature extraction we applied the method introduced above. Fig. 4 and Fig. 5 show the created feature map of two test tracks from previously recorded data. Our experiments are conducted on these two tracks and the results will be explained in detail in the next section.

C. Data Association

Extensive research has been conducted to explore the problem of matching two sets of 2D objects under arbitrary translations, rotations, and scale changes. It is widely used in fingerprint recognition [13], object recognition [14], image matching, scene recognition, and other areas. We use a pattern matching method to find the association between a set of detected features and the features in the *a priori* map. Different from a common affine invariant matching in computer vision area, the features detected by LiDAR have no scale changes, and the point features are of two different types (pole and corner). For our application scenario, we propose a new method for map matching.

Before each feature registration, we first update a local grid feature map. The map data within the sensor's detection range are encoded as a grid map as shown in Fig. 6. Through indexes, we can find out whether there is a certain type of feature that exists in a grid cell with a time complexity of $O(1)$.

For each feature detection, detected features are denoted as three sets:s_p, s_c, s_w, to represent the poles, corners and walls. Sets S_p, S_c, S_w contain three types of map features in the local area within the sensor range. Further we define the tuple sets $t_p = \{(p_i, p_j, d_{i,j}) | p_i, p_j \in s_p \, i \neq j\}, t_c = \{(p_i, p_j, d_{i,j}) | p_i, p_j \in s_c, i \neq j\}$ and $t_{pc} = \{(p_i, p_j, d_{i,j}) | p_i \in s_p, p_j \in s_c\}$ where $d_{i,j}$ is the distance between p_i and p_j. Similarly, we define tuple sets T_p, T_c, T_{pc} for the local map data. Algorithm 1 computes the matched points of detected features and the map elements. The transformation between these two sets of features are also calculated.

D. Graph Optimization for Pose Estimation

Next, we introduce how to use a graph-based optimization to estimate vehicle poses with 3-DOF. Fig. 7 illustrates the problem in form of a graph. Very similar to the SLAM problem, we denote the state as $x = [x_1, x_2, ..., x_K, m_1, m_2, ..., m_M]^T$, where K is the size of poses

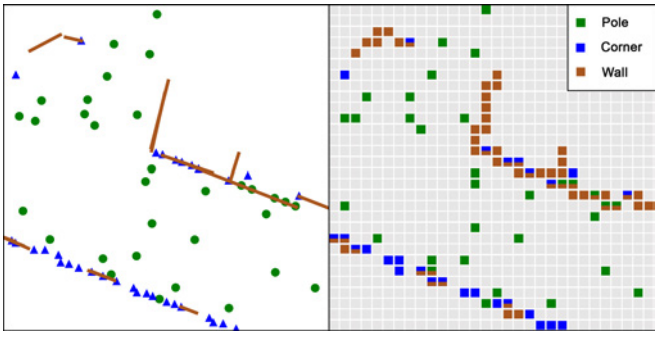

Fig. 6. Left: part of the feature map, it is encoded to local grid map (right) for fast retrieve.

Algorithm 1. ClusterMatch

Input: $s_p, s_c, s_w, S_p, S_c, S_w, Grid\ Map, K$
Output: $Matched\ Pairs,\ Transformation$

```
Generate t_p, t_c, t_pc, T_p, T_c, T_pc
for k = 0 → K do
    Randomly select a tuple (p_i, p_j, d_{i,j}) from t_p, t_c or t_pc
    Find a tuple (p_m, p_n, d_{m,n}) from T_p, T_c or T_pc, where d_{m,n} is the
    closest value to d_{i,j}
    Calculate Transformation Rt for point pairs (p_i, p_m), (p_j, p_n) or
    (p_i, p_n), (p_j, p_m)
    Set succeed ← false
    for each left element e in s_p, s_c, s_w do
        Apply Rt to e, get e'
        if e' matches to E in the Grid Map do
            Add pair (e', E) to Matched Pairs
        if size of Matched Pairs ≥ match_threshold do
            succeed ← true
    end for
    if succeed do
        Calculate Transformation from Matched Pairs
end for
```

included in each optimization, M is the total number of features that can be observed within these poses. The maximum *a posteriori* x_{MAP} can be written as:

$$x_{MAP} = argmax_x p(x|u,y). \tag{1}$$

Here u is the motion of the vehicle between two adjacent poses, we use the data from the wheel odometer in this study, and y is the measurement of all observed features within K poses. Applying Bayes theorem to (1), we get:

$$\begin{aligned} x_{MAP} &= argmax_x p(x|u,y) = argmax_x \frac{p(y|x)p(x|u)}{p(y|u)} \\ &= argmax_x p(y|x)p(x|u). \end{aligned} \tag{2}$$

The motion function and measurement function are:

$$x_k = f(x_{k-1}, u_k, w_k)(k = 1, ..., K) \tag{3}$$

$$y_{kj} = h(x_k, m_j, v_{kj})(k = 1, ..., K; j = 1, ..., M). \tag{4}$$

Assume the motion noises w_k and v_{kj} to be independent and additive. R_k and Q_{kj} are the covariances of w_k and v_{kj} respectively. Now the motion and measurement error can be written as follows:

$$e_{f,k} = x_k - f(x_{k-1}, u_k) \tag{5}$$

$$e_{h,kj} = y_{kj} - h(x_k, m_j). \tag{6}$$

Take the negative logarithm to $p(y|x)p(x|u)$ in (2)

$$-ln(p(y|x)p(x|u)) = -\sum_{k=1}^{K} lnp(y_k|x_k) - \sum_{k=1}^{K} lnp(x_k|x_{k-1}, u_k). \tag{7}$$

After substitution of (5), (6) to (7), removing the constant terms and normalizing it, we can get:

$$J(x) = \sum_k e_{f,k}^T R_k^{-1} e_{f,k} + \sum_k \sum_j e_{h,kj}^T Q_{kj}^{-1} e_{h,kj}. \tag{8}$$

Then we consider the constraints from the *a priori* map. As shown in Fig. 7, p_j represents *a priori* information from the prebuilt feature map. Assume the covariance of the *a priori* map is known as S_k. We can now simply calculate the map error as

$$e_{m,j} = m_j - p_j. \tag{9}$$

For point features, $e_{m,j}$ is the position difference of two points, and for wall features $e_{m,j} = (\Delta d_j, \Delta\theta_j)$. By adding the map error to (8), we can get the final objective function:

$$J(x) = \sum_k e_{f,k}^T R_k^{-1} e_{f,k} + \sum_k \sum_j e_{h,kj}^T Q_{kj}^{-1} e_{h,kj} + \sum_k e_{m,j}^T S_j^{-1} e_{m,j}. \tag{10}$$

Our goal is to minimize $J(x)$. We use the g2o [12] library to build graphs like Fig. 7 and use the Levenberg-Marquardt algorithm to iteratively solve the problem. The number of poses taken into optimization is limited by the size of the sliding window. In SLAM problems, to ensure computation efficiency, a marginalization technique is usually used to limit the number of states involved in the computation. Here we directly discard the previous states. The reason is that our *a priori* map built from the offline SLAM approach is self-consistent, we give high confidence in it. As a result, it provides strong constraints on the current states. If a large size for the sliding window is selected, the previous constraints have little effect on the current states. And this can also avoid the accumulation of linearization errors caused by marginalization.

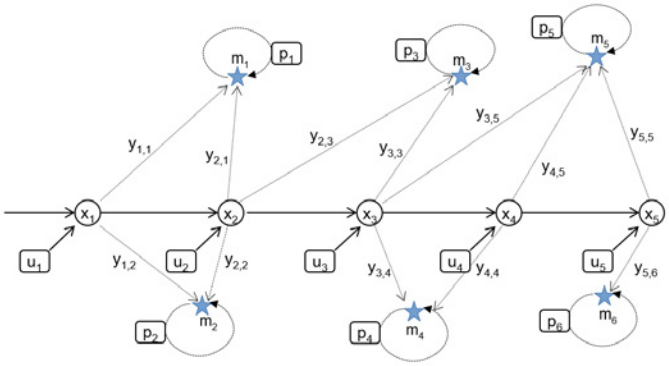

Fig. 7. Optimization problem represented as a graph. x_k are poses of vehicles, m_j are features, u_i are odometer inputs, y_{kj} are the measurement of features, and p_j are *a priori* information of features.

IV. RESULTS

We ran our method on an autonomous test vehicle and collected datasets. The vehicle is equipped with a Velodyne HDL-64E LiDAR scanner, an Applanix POS-LV 510 navigation system, and other sensors such as cameras, radar and other laser scanners. The algorithm was run on a laptop with an Intel Core i7-7700HQ CPU. A GPS position was required at the beginning to initialize the algorithm. The method provides 2D position and heading information at a frequency of 100Hz.

We use two test roads. Test Road 1 is "Thielallee", a street near our research institute. The length of one lap on this road is about 1.2 km. There are plenty of trees along the road. Test Road 2 is a test area for connected and automated driving in Berlin. It contains around 5.6 km main roads with complex traffic conditions. Fig. 4 and Fig. 5 show the feature map of

TABLE I

CORRECT MATCHING RATE AND AVERAGE CALCULATION TIME OF THE PROPOSED PATTERN MATCHING METHOD.

Road	Correct Matching	Average Matching Time
Test Road 1	99.52 %	0.27 ms
Test Road 2	98.65 %	0.45 ms

these two test roads. The feature maps are built based on the previously collected data using the off-line SLAM method introduced in the previous section. We conducted real-world tests on Road 1 and tests on the dataset of Road 2.

Three experiments were conducted. First, we tested our map matching method. We obtain feature detection results from LiDAR scanner every 0.1 second on average. The feature matching results on the Test Road 2 is shown in Fig. 8. The average number of successfully matched poles in this area was 13.9 at each time, and this number increased to 21.4 if all three types of features were used. We can find that on the left side of the test road, the number of successfully associated poles is under 5 for most parts of this region. It is significantly improved if corners and walls are added.

Table I shows the matching success rate on two test roads. A successful matching means at least half (6 at minimum) of the detected features are associated with feature map, and these associations lead to correct transformation result. The correct matching rate are 99.52% and 98.65% on the two test roads, respectively. Failed match is usually due to the number of detected features is insufficient to match with the feature map reliably. In the case of a failed match, the corresponding pose has only a constraint from wheel odometer. In terms of running time, the ClusterMatch algorithm has a very high efficiency and is sufficient to support real-time localization.

Compared with the commonly used Nearest Neighbor matching method, the ClusterMatch has several advantages. First, it is not sensitive to the accuracy of the initial position. In fact, it can perform global matching without knowing the initial position. But in practical, the detected features are only matched with a local sub-map in order to achieve a high computation efficiency. The size of the sub-map is the effective feature detection range with an extension, which is 60 meters around current position in total. Therefore, the initialization of the proposed method requires only a rough position, and an ordinary GPS is enough. Secondly, the Nearest Neighbor method requires high accurate and reliable *a priori* information of current pose before conducting the map matching. Otherwise, the correct data association may not be obtained. ClusterMatch performs feature matching within the scope of local sub-maps, and data association does not depend on the accuracy of pose prediction.

In the second experiment, we compared the feature localization results with the trajectory reported by the navigation system. With the RTK (Real-Time Kinematic) technique, it can provide a comparably precise positioning information. However, analysis of the results in the test roads showed that only when the vehicle is stationary, the error may fall below 10 cm. A larger error occurred during movement, and

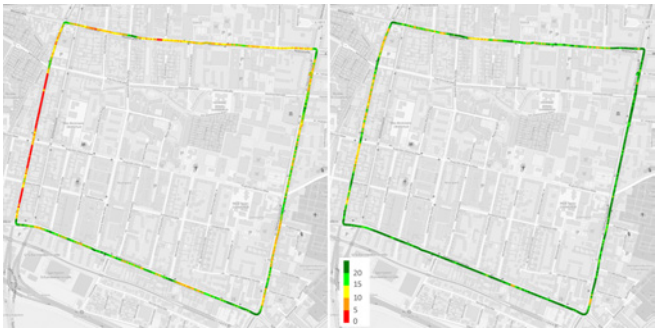

Fig. 8. Left: number of matched poles; Right: number of matched all features. Color coding as indicated.

sometimes it showed an unpredictable drift that can exceed 1 meter. This is a situation that our solution needs to avoid. Therefore, we conducted a repeatability test to evaluate the lateral accuracy. The test method was to let the human driver repeatedly drive on the short test road and try to keep the vehicle in the center of the lane. The repeatability can be estimated from the recorded positioning trajectories. As shown in Fig. 9, the localization method proposed shows better consistency. After removing the U-turn area where it is difficult to maintain a consistent driving trajectory, the average lateral error is around 15 cm. Considering the error introduced by the human driver, this result is acceptable. The error of positioning results reported by the Applanix system are significantly larger.

On the third experiment we evaluated the accuracy of our localization algorithm. As shown in the above experiment, the navigation system has an unpredictable drift error, so we did not use it as the ground truth. We ran the offline SLAM algorithm introduced earlier on the recorded test dataset to get optimized poses with timestamps as ground truth. Then we could compare our localization performance against it. Fig. 10 shows the distance error over time. Compared with the navigation system, the proposed feature localization method has a smaller distance error most of the time. And its results show a smaller variance.

Table II outlines the experiment results of the average distance error, RMSE and maximum distance error of the two methods. On Test Road 1, the average distance error and RMSE of the proposed method is 10.2 cm and 11.8 cm, respectively. The maximum distance error is 32.0 cm. On Test Road 2, the error became larger. It increased to 15.2 cm and 17.1 cm. The maximum distance error is 37.2 cm on this road. The Applanix system shows a relatively larger error on both test roads, especially on Test Road 2. The reason could be that Test Road 2 is in a residential area with plenty of buildings along the road, which may affect GPS signals. It can be seen that the proposed method presents accurate localization results, showing advantages over the positions reported by the Applanix system.

V. CONCLUSION

We introduced a LiDAR based localization method for autonomous vehicles in the urban environment. The features

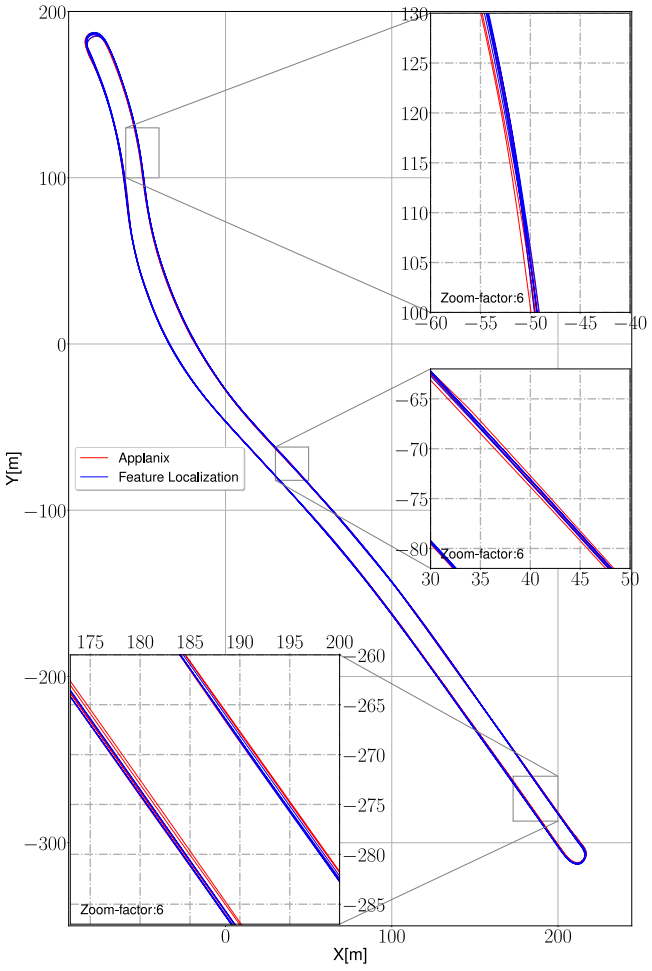

Fig. 9. Trajectory plot of feature localization results and Applanix reported positions.

TABLE II

COMPARISON OF THE PROPOSED METHOD AND A NAVIGATION SYSTEM.

Method	***Test Road 1***			***Test Road 2***		
	Average Distance Error	***RMSE***	***Max. Error***	***Average Distance Error***	***RMSE***	***Max. Error***
Applanix	0.162	0.211	0.881	0.699	0.749	1.374
Proposed	0.102	0.118	0.320	0.152	0.171	0.372

we used were pole-like structures, corners and walls. We also outlined the pattern-matching-based data association method and the optimization method for pose estimation. Experiments were conducted in practice and on datasets. The proposed method showed its capability of accurate localization in residential area where only a few poles exist. In both practical and dataset tests, the proposed method presented better repeatability and accuracy compared to a high-cost navigation system.

REFERENCES

[1] D. Wilbers, C. Merfels, and C. Stachniss, "Localization with sliding window factor graphs on third-party maps for automated driving," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 5951–5957.

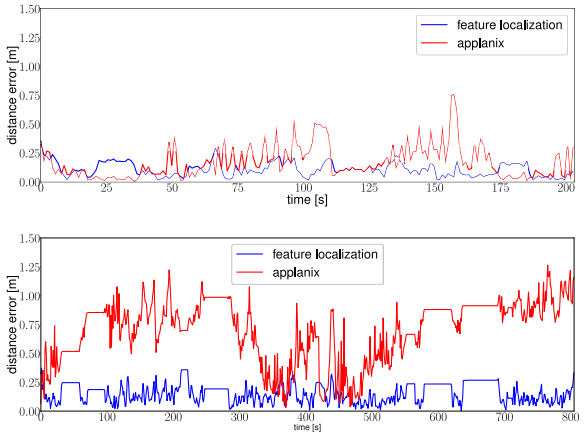

Fig. 10. Distance error over time, on Test Road 1 (top) and Test Road 2 (bottom).

[2] C. Brenner, "Vehicle localization using landmarks obtained by a lidar mobile mapping system," *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences:[PCV 2010-Photogrammetric Computer Vision And Image Analysis, Pt I] 38 (2010), Nr. Part 3A*, vol. 38, no. Part 3A, pp. 139–144, 2010.

[3] 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)*. IEEE, 2016, pp. 2161–2166.

[4] A. Schlichting and C. Brenner, "Localization using automotive laser scanners and local pattern matching," in *2014 IEEE Intelligent Vehicles Symposium Proceedings*. IEEE, 2014, pp. 414–419.

[5] Z. Tao, P. Bonnifait, V. Fremont, and J. Ibanez-Guzman, "Lane marking aided vehicle localization," in *16th International IEEE Conference on Intelligent Transportation Systems (ITSC 2013)*. IEEE, 2013, pp. 1509–1515.

[6] A. Hata and D. Wolf, "Road marking detection using lidar reflective intensity data and its application to vehicle localization," in *17th International IEEE Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2014, pp. 584–589.

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

[8] D. Ronzoni, R. Olmi, C. Secchi, and C. Fantuzzi, "Agv global localization using indistinguishable artificial landmarks," in *2011 IEEE International Conference on Robotics and Automation*. IEEE, 2011, pp. 287–292.

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

[10] C. Wu, T. A. Huang, M. Muffert, T. Schwarz, and J. Gräter, "Precise pose graph localization with sparse point and lane features," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017, pp. 4077–4082.

[11] C. Qian, H. Liu, J. Tang, Y. Chen, H. Kaartinen, A. Kukko, L. Zhu, X. Liang, L. Chen, and J. Hyyppä, "An integrated gnss/ins/lidar-slam positioning method for highly accurate forest stem mapping," *Remote Sensing*, vol. 9, no. 1, p. 3, 2017.

[12] G. Grisetti, R. Kümmerle, H. Strasdat, and K. Konolige, "g2o: a general framework for (hyper) graph optimization," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), Shanghai, China*, 2011, pp. 9–13.

[13] S.-H. Chang, F.-H. Cheng, W.-H. Hsu, and G.-Z. Wu, "Fast algorithm for point pattern matching: invariant to translations, rotations and scale changes," *Pattern recognition*, vol. 30, no. 2, pp. 311–320, 1997.

[14] A. C. Berg, T. L. Berg, and J. Malik, "Shape matching and object recognition using low distortion correspondences," in *2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05)*, vol. 1. IEEE, 2005, pp. 26–33.