Robotica (2022), **40**, pp. 817–833
doi:10.1017/S0263574721000862

CAMBRIDGE UNIVERSITY PRESS

RESEARCH ARTICLE

Autonomous vehicle self-localization in urban environments based on 3D curvature feature points – Monte Carlo localization

Qi Liu[1], Xiaoguang Di[1,*] and Binfeng Xu[2]

[1]Control and Simulation Center, Harbin Institute of Technology, Harbin, China and [2]Department of Intelligent Driving, Horizon Robotics, Shanghai, China
*Corresponding author. Email: dixiaoguang@hit.edu.cn

Received: 4 July 2020; **Revised:** 26 May 2021; **Accepted:** 26 May 2021; **First published online:** 23 June 2021

Keywords: Autonomous vehicle localization, Pose graph, Information matrix, Feature points, Particle filter

Abstract

This paper proposes a map-based localization system for autonomous vehicle self-localization in urban environments, which is composed of a pose graph mapping method and 3D curvature feature points – Monte Carlo Localization algorithm (3DCF-MCL). The advantage of 3DCF-MCL is that it combines the high accuracy of the 3D feature points registration and the robustness of particle filter. Experimental results show that 3DCF-MCL can provide an accurate localization for autonomous vehicles with the 3D point cloud map that generated by our mapping method. Compared with other map-based localization algorithms, it demonstrates that 3DCF-MCL outperforms them.

1. Introduction

Accurate and robust self-localization is an essential task for autonomous vehicles. Traditional vehicle localization uses GNSS [1], which is low cost and attains good performance in the open environments. However, in complex urban environments, GNSS has difficulty in providing accurate localization for autonomous vehicles due to the occlusion of tall buildings [2]. On the contrary, in such environments, LiDAR and vision sensors can obtain rich structural information, which is highly suitable for autonomous vehicles to estimate their poses. However, in terms of pose estimation, the LiDAR-based method is more accurate and reliable than the vision-based method for two reasons: (1) accuracy: the distance detected by LiDAR is more accurate than the vision-based method [3]; (2) reliability: LiDAR is hardly affected by illumination change [3]. Consequently, LiDAR has been intensively applied to the pose estimation of autonomous vehicles.

LiDAR-based autonomous vehicle localization can be divided into two categories [4]: the map-known method and the map-unknown method known as SLAM. When the map is unknown, SLAM calculates the poses of the autonomous vehicle based on the data scanned by the LiDAR and simultaneously constructs the map [5, 6]. The shortcoming of SLAM is that small errors inevitably occur in most registrations between two adjacent frames. As a result, the localization error increases gradually with the increasing driving distance. Although loop closure is widely used in SLAM for reducing cumulative errors, correcting the past poses using future information does not meet the real-time requirement for autonomous vehicle localization. In recent years, researchers have widely investigated map-based localization [7, 8]. The map in map-based localization can be generated by pose graph mapping method with multi-sensor fusion [9, 10]. The relationship between SLAM and map-based localization is that SLAM is used for building a prior map for map-based localization, and in order to build a high-definition map,

© The Author(s), 2021. Published by Cambridge University Press.

https://doi.org/10.1017/S0263574721000862 Published online by Cambridge University Press

some updates and corrections must be done on the map, which is built by SLAM. Notably, the difference between SLAM and map-based localization is that SLAM calculates the poses of the autonomous vehicle according to the point cloud registration of two adjacent frames, while map-based localization calculates the poses of the autonomous vehicle according to the registration of the point cloud scanned by LiDAR to the prior map. Compared with SLAM, map-based localization can obtain abundant information from the prior map, which provides more constraints on the localization and each registration is independent of the previous registrations in map-based localization. As a result, map-based localization has no cumulative errors. Therefore, map-based localization is more suitable for autonomous driving because it can provide more accurate localization.

The aim of this work is to solve the localization problem for autonomous vehicles in the urban environments, where GNSS and SLAM cannot provide the accurate localization. As discussed in the previous two paragraphs, GNSS cannot provide an accurate localization due to the occlusion of tall obstacles and SLAM suffers from the cumulative errors in the long driving distance. In this paper, an autonomous vehicle localization system based on map is proposed, which includes two parts: the generation of the 3D point cloud map and the map-based localization algorithm. For mapping, we construct the pose graph [11] using LiDAR and GNSS data. Then, the 3D point cloud map is generated by using the optimized poses. For localization, a novel map-based localization algorithm is proposed. For the poses of the autonomous vehicle, we pay more attention to x (latitude position), y (longitude position), and yaw angle (heading angle) components because the z (vertical position), pitch angle, and roll angle of autonomous vehicles change little in urban environments.

The main contributions of this work are as follows:

- A pose graph mapping method based on adaptive information matrix is proposed using LiDAR and GNSS data. The information matrices are constructed according to the result of the point cloud registration. GNSS data are used as prior pose information, which improves the accuracy of the pose graph.
- A novel realistic application localization algorithm based on 3D curvature feature points – Monte Carlo Localization (3DCF-MCL) is proposed, which combines the high accuracy of 3D feature point registration and the robustness of particle filter.

The rest of this paper is organized as follows. The related works of mapping and localization are discussed in Section 2. The pose graph mapping method based on adaptive information matrix and 3DCF-MCL algorithm are described in Section 3. Experiments are described in Section 4. Finally, the conclusion and future works are presented in Section 5.

2. Related works

For SLAM method of LiDAR-based autonomous vehicle localization, iterative closest points (ICP) [12] is a prevailing 3D point cloud registration algorithm, which has been widely used in LiDAR SLAM and there are several ICP variants [13, 14, 15]. Ethzasl_icp_mapper [16] provided a real-time 2D and 3D ICP-based SLAM system. However, there was no loop closure detection in ethzasl_icp_mapper, which prevented it from providing highly accurate localization and high-quality map in large-scale environments. Normal distributions transform (NDT) [17] is another prevailing registration algorithm that models a two-dimensional environment by normal distributions for the registration. The authors divided the whole map space into fixed 2D cells, and a corresponding normal distribution was generated for each cell. Then, the point cloud of the new frame was matched to the normal distributions rather than to the original point cloud. Martin Magnusson [18] extended NDT to 3D domain. In addition, Martin Magnusson et al. [19] demonstrated that NDT was more robust than ICP. However, NDT was more sensitive to the initial guess of the pose than the ICP. Zhang et al. [20] extracted feature points from the original point cloud for registration. In all the methods discussed above, the localization errors increase gradually with the increasing driving distance. Such circumstance is known as the global consistent problem.

https://doi.org/10.1017/S0263574721000862 Published online by Cambridge University Press

In the map-known method of LiDAR-based autonomous vehicle localization, the map can be divided into grid-based map, topological map, point cloud map, and octomap. Hans Moravec et al. [21] proposed grid-based map and applied it to the field of robots. A grid map is easy to implement and maintain. However, grid-based approaches are computationally expensive and typically require a huge amount of memory [22]. Tim Bailey [23] proposed the topological map. Topological map is a compact map representation method based on topological structure that occupies very little storage space. However, topological maps only retain the key positions in the environment and the relationship among these key positions [23]. As a result, topological maps are highly suitable for the path planning task of the mobile robot. The main problem of this method is perceptual aliasing. In other words, this method always has a risk of making two distinct locations appear identical to the robot's sensors [24]. The point cloud map is composed of the data scanned by LiDAR and the data are spliced according to the pose sequence of the robot. Point cloud map retains the complete information of the environment, which is suitable for the robot to achieve high-precision localization. Compared with point cloud map, Octomap [25] has good performance in compression and is ideal for the obstacle avoidance of robot navigation. However, it cannot be used in the scenes where the robot needs accurate localization. To guarantee highly accurate localization, we selected point cloud map as the prior map.

Mapping with LiDAR has been widely investigated [5]. In recent years, graph-based SLAM has prevailed. In graph-based SLAM, the poses of the autonomous vehicle are the variables that need to be optimized. The LiDAR odometry, loop closure results, and measurements from other sensors are constraints for the optimized variables. To determine the most likely trajectory of the autonomous vehicle, a global optimization is used to optimize the poses once a graph for the SLAM problem has been built. Berkeley Localization and Mapping (BLAM) [26] improved Lidar Odometry and Mapping in Real-Rime (LOAM) [20] by adding loop closure. Renaud Dubé et al. [27] proposed a 3D LiDAR-based loop closure detection approach by matching 3D segments. This approach was also a graph-based SLAM. Ellon Mendes et al. [28] proposed an online 3D ICP-based mapping method, and its graph was optimized by iSAM2 [29]. Kenji Koide et al. [10] proposed NDT-based graph SLAM that combined LiDAR and GNSS data. In their method, the poses of the autonomous vehicle were the variables that need to be optimized, and NDT registration results of adjacent frames were the constraints for the adjacent pose nodes. GNSS data were added to the graph as prior information to correct LiDAR odomtery. However, the accuracy of NDT is not the best for 3D point cloud registration on the KITTI odometry benchmark [30].

As for the robustness of autonomous vehicle localization, Shinpei Kato et al. [7] proposed an autonomous vehicle localization method using NDT/ICP registration with a 3D prior point cloud map. Both NDT and ICP typically require defining a registration cost function between the point clouds of the LiDAR and map and reducing the cost function by using gradient descent iteratively. These methods are highly dependent on the initial pose guess of the autonomous vehicle and easily fall into local minima because of the nature of the gradient-based optimization method. Compared with the gradient-based optimization method, Monte Carlo Localization (MCL) [31] is more robust in avoiding the local minima problem. Numerous studies have been conducted on MCL in terms of robot localization [32, 33]. MCL in these references usually uses 2D prior map, however, we can build 3D prior map using multilayer LiDAR. The 3D map can provide more constraints on pose estimation, which will improve the accuracy of pose estimation. Furthermore, we can reduce the computational complexity by changing the number of particles.

3. System overview and approach

Our localization system consists of two parts: mapping and localization. Given that the accuracy of LOAM [20] is the best of which is achieved by the LiDAR-only estimation method on the KITTI odometry benchmark site, our point cloud registration method is similar to the registration of LOAM by adding wheel odometry to provide the initial pose estimation. For pose graph mapping, this study constructs the information matrix between the two LiDAR odometry nodes according to the result of the point cloud

https://doi.org/10.1017/S0263574721000862 Published online by Cambridge University Press

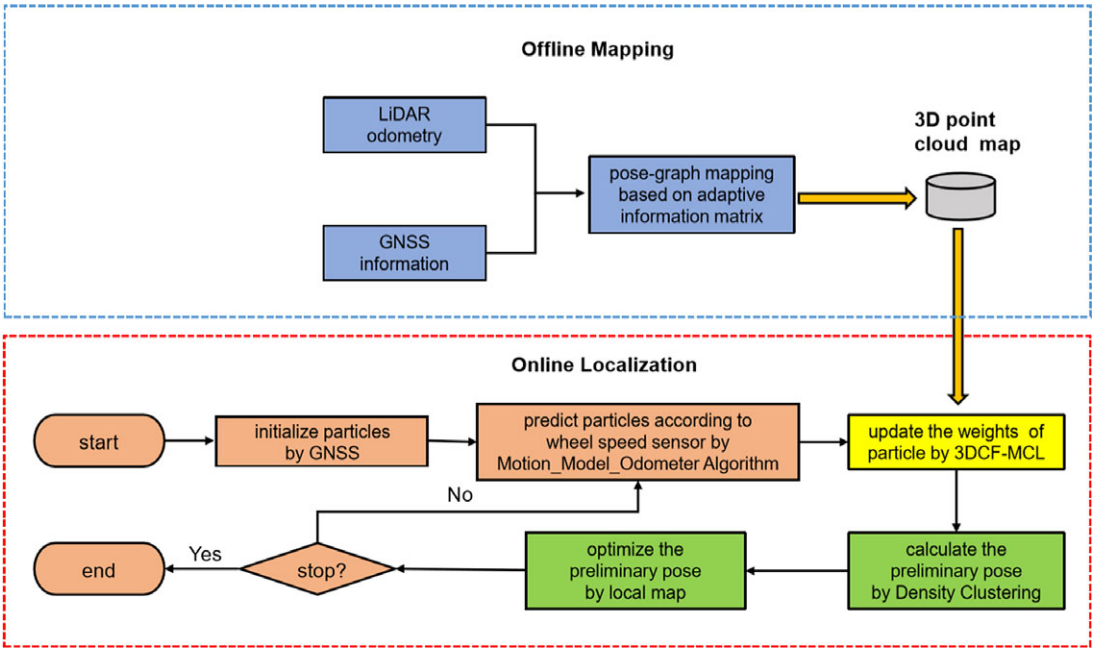


Figure 1. *The flow diagram of our mapping method and 3DCF-MCL.*

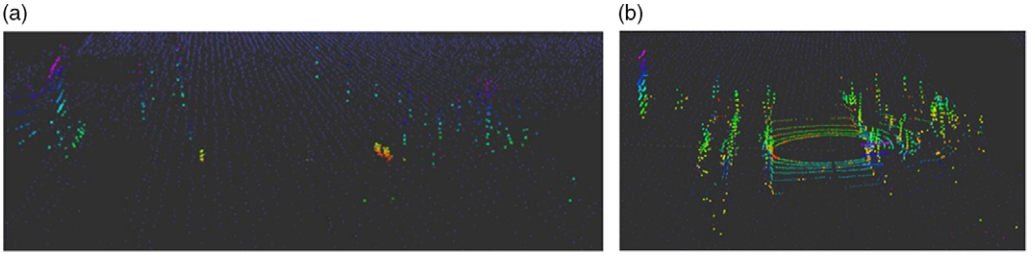


Figure 2. *3D feature points. (a) Edge feature points. (b) Flat feature points.*

registration of the two adjacent frames. When a loop closure is detected, an edge is added to the graph according to the registration result of these two frames. To get more accurate trajectory, GNSS measurements are added to the graph as prior pose information. Robust and accurate localization is achieved by 3DCF-MCL, which combines the high accuracy of 3D feature point registration and the robustness of particle filter. The flow diagram of our mapping method and 3DCF-MCL is shown in Fig. 1. Figure 2 shows the extracted edge feature points and flat feature points as in [20].

3.1. Pose graph mapping based on adaptive information matrix

Although LOAM has good performance in LiDAR odometry, we still find a large drift of localization in the course of experiments due to the cumulative errors when the autonomous vehicle travels over a long distance. To obtain more accurate poses that will be used for generating maps, this paper adds external constraints from other sensors to correct the autonomous vehicle poses calculated by LiDAR odometry only. Although vision-based methods, such as Mono-SLAM and RGBD-SLAM, are effective in indoor environments, their localization accuracy degrades significantly in some outdoor environments due to strong illumination changes or fewer features. Therefore, we build a pose graph using LiDAR and GNSS data.

https://doi.org/10.1017/S0263574721000862 Published online by Cambridge University Press

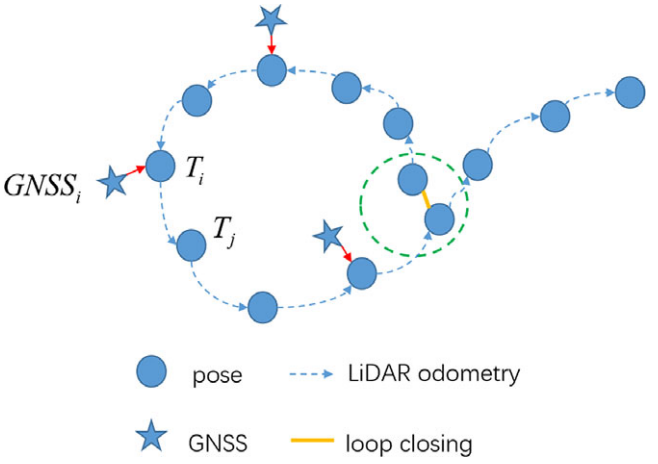


Figure 3. *The schematic diagram of the pose graph mapping method based on adaptive information matrix. The blue circles represent the autonomous vehicle poses (key frames) calculated by LiDAR odometry, and the stars represent the autonomous vehicle poses measured by GNSS. The blue dashed lines represent the relative transformation between the adjacent poses calculated by the point cloud registration; the red solid lines represent the links between GNSS and the poses of LiDAR odometry; the yellow solid line represents loop closing.*

The first step in our localization system is the generation of the 3D point cloud map. For the generation of the 3D point cloud map, we propose a pose graph (Graph SLAM) mapping method, which can construct the information matrix adaptively. In Graph SLAM, the SLAM problem is solved by constructing and optimizing a graph whose nodes represent parameters to be optimized, such as the poses of the autonomous vehicle. Meanwhile, the edges represent constraints, such as relative poses between the poses of the autonomous vehicle. The graph is optimized so that the errors (Eq. (3)) between the parameters (the poses of the autonomous vehicle) and the constraints (Eq. (2), the relative poses between the poses of the autonomous vehicle) are minimized. Once the graph is constructed, the pose sequence of the vehicle can be obtained by calculating a set of poses that are mostly consistent with the constraints (for more details, see [11] Section IV). In Graph SLAM, the uncertainty of the relative poses can be modeled by an information matrix (see [11] Section IV). Adaptive information matrix (calculated by Eq. (5)) means that we construct the information matrix between the two LiDAR odometry nodes according to the result of the point cloud registration of the two adjacent frames. Furthermore, GNSS measurements are added to the graph as prior pose information of the vehicle. Two factors have an influence on the construction of the information matrices between GNSS prior information and LiDAR odometry nodes. One is the covariance matrix measured by GNSS and the other is the difference of timestamps between GNSS prior pose and its nearest LiDAR odometry node. For the prior poses measured by GNSS, the confidence of these measurements are determined by the covariance of the autonomous vehicle poses measured by GNSS. The greater the confidence level is, the stronger the constraints to the poses of LiDAR odometry are. During the optimization of the graph, GNSS data are considered fixed poses. Notably, GNSS prior information is excluded from the optimized processing. When a loop closure is detected, an edge will be added to the graph according to the registration result of the two frames. Finally, all the nodes and edges form the pose graph. Solving the pose graph is essentially a nonlinear least squares problem. The optimized variables are the poses of the autonomous vehicle and the edges are the measurements of the LiDAR odometry and GNSS. The 3D map is generated by stitching the point clouds according to the optimized poses calculated by the pose graph. The schematic diagram of the pose graph is shown in Fig. 3.

https://doi.org/10.1017/S0263574721000862 Published online by Cambridge University Press

We first define one error term. Then, we define the total cost function. As shown in Fig. 3, we assume that LOAM calculates the transformation matrix (see [34] Chapter 6) T_i (4 * 4 matrix) and T_{i+1}.

We can obtain the transformation $\Delta T_{i,i+1}$ between T_i and T_{i+1}:

$$\Delta T_{i,i+1} = T_i^{-1} T_{i+1} \tag{1}$$

If the timestamp of $GNSS_i$ (t_{GNSS_i}) is close enough to the timestamp of T_i (t_{T_i}), (t_{GNSS_i} - t_{T_i} <δt, δt=0.01), we define $GNSS_i$ is the prior pose of T_i. As a result, Eq. (1) will not be accurate due to the constraint of $GNSS_i$:

$$\Delta T_{i,i+1} \neq T_i^{-1} T_{i+1} \tag{2}$$

We define one error term $e_{i,i+1}$:

$$e_{i,i+1} = \ln \left(\Delta T_{i,i+1}{}^{-1} T_i^{-1} T_{i+1}\right)^{\vee} \tag{3}$$

where $^{\vee}$ represents the exponential map and $e_{i,i+1}$ is a 1*6 vector (see [34] Chapter 7.1.3 or [11] Chapter APPENDIX). The total cost function of the nonlinear optimization can be defined as follows:

$$\arg\min_{\zeta} \frac{1}{2} \sum_{i,i+1 \in \varepsilon} e_{i,i+1}^T \left(\sum\nolimits_{i,i+1}^{-1}\right) e_{i,i+1} \tag{4}$$

where ζ is the variable to be optimized. In fact, ζ represents the sequence of the vehicle poses. $i, i+1$ represent the indexes of two consecutive nodes. They represent two vehicle poses. ε represents all the nodes. For two consecutive nodes, the information matrix $\sum_{i,i+1}^{-1}$ [11] (a 6*6 matrix) is defined as

$$\sum\nolimits_{i,i+1}^{-1} = \frac{1}{S_{i,i+1}} \times \begin{bmatrix} C & 0 & 0 & 0 & 0 & 0 \\ 0 & C & 0 & 0 & 0 & 0 \\ 0 & 0 & C & 0 & 0 & 0 \\ 0 & 0 & 0 & C & 0 & 0 \\ 0 & 0 & 0 & 0 & C & 0 \\ 0 & 0 & 0 & 0 & 0 & C \end{bmatrix} \tag{5}$$

where C is an integer constant and $S_{i,i+1}$ is defined as follows:

$$S_{i,i+1} = b^2 + (a^2 - b^2) \frac{1 - e^{20 fit_score}}{1 - e^{20a^2}} \tag{6}$$

where *fit_score* is calculated according to the registration method of LOAM [20], a and b are constant numbers. We set $a = 5$ and $b = 0.1$. Specifically, $fit_score \propto d_{\mathcal{E}} + d_{\mathcal{H}}$. $d_{\mathcal{E}}$ is the sum of the distance for all the edge feature points to their corresponding feature lines in the reference frame. Similarly, $d_{\mathcal{H}}$ is the sum of the distance for all the flat feature points to their corresponding feature planar patch in the reference frame. More details of the computation of $d_{\mathcal{E}}$ and $d_{\mathcal{H}}$ are described in ref. [20].

For the constraints between GNSS prior poses and LiDAR odometry nodes, this paper constructs the information matrix according to the covariance matrix measured by GNSS and the difference of the timestamp between GNSS prior pose and its closest LiDAR odometry node.

$$\sum\nolimits_{gnss,odom}^{-1} = \frac{1}{\Delta t} \times \begin{bmatrix} \frac{1}{g_1} & 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{1}{g_2} & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{g_3} & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{g_4} & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{g_5} & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{1}{g_6} \end{bmatrix} \tag{7}$$

In Eq. (7), $\Delta t = |t_{odom} - t_{gnss}|$ represents the difference of timestamp between GNSS and its closest LiDAR odometry node. We set Δt as a penalty item. Because if Δt is large, it means that the measurement from GNSS for the autonomous vehicle pose is not suitable for correcting the corresponding

https://doi.org/10.1017/S0263574721000862 Published online by Cambridge University Press

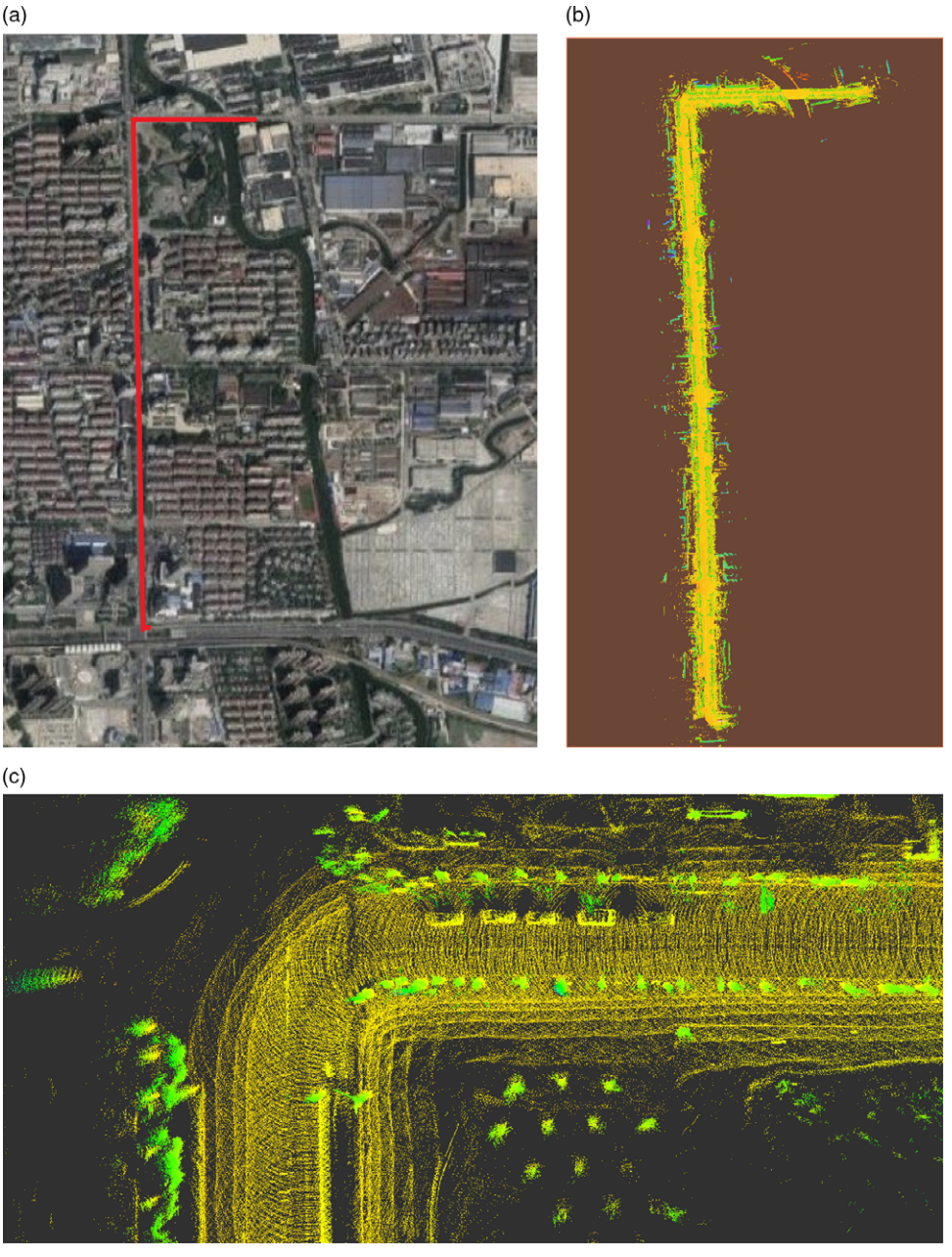


Figure 4. *The 3D point cloud map generated by our method. (a) Google map. (b) The 3D point cloud map. (c) The local enlarged 3D point cloud map.*

pose obtained by LiDAR odometry directly at this timestamp. g_1, g_2, g_3, g_4, g_5, and g_6 represent the uncertainty of the six-dimensionl pose of the autonomous vehicle calculated by RTK-GNSS.

Equatiom (4) can be solved by Gauss–Newton or Levenberg–Marquart method [35]. Afterward, the optimized poses will be obtained, and the 3D map can be generated by stitching point clouds according to the optimized poses. Figure 4(b) is the 3D point cloud map generated by our method. The trajectory of the vehicle is shown in Fig. 4(a). In addition, the driving distance is 1776.99 m. Figure 4(c) is the local enlargement of Fig. 4(b). The vehicles parking on the road can be seen clearly in Fig. 4(c), which

https://doi.org/10.1017/S0263574721000862 Published online by Cambridge University Press

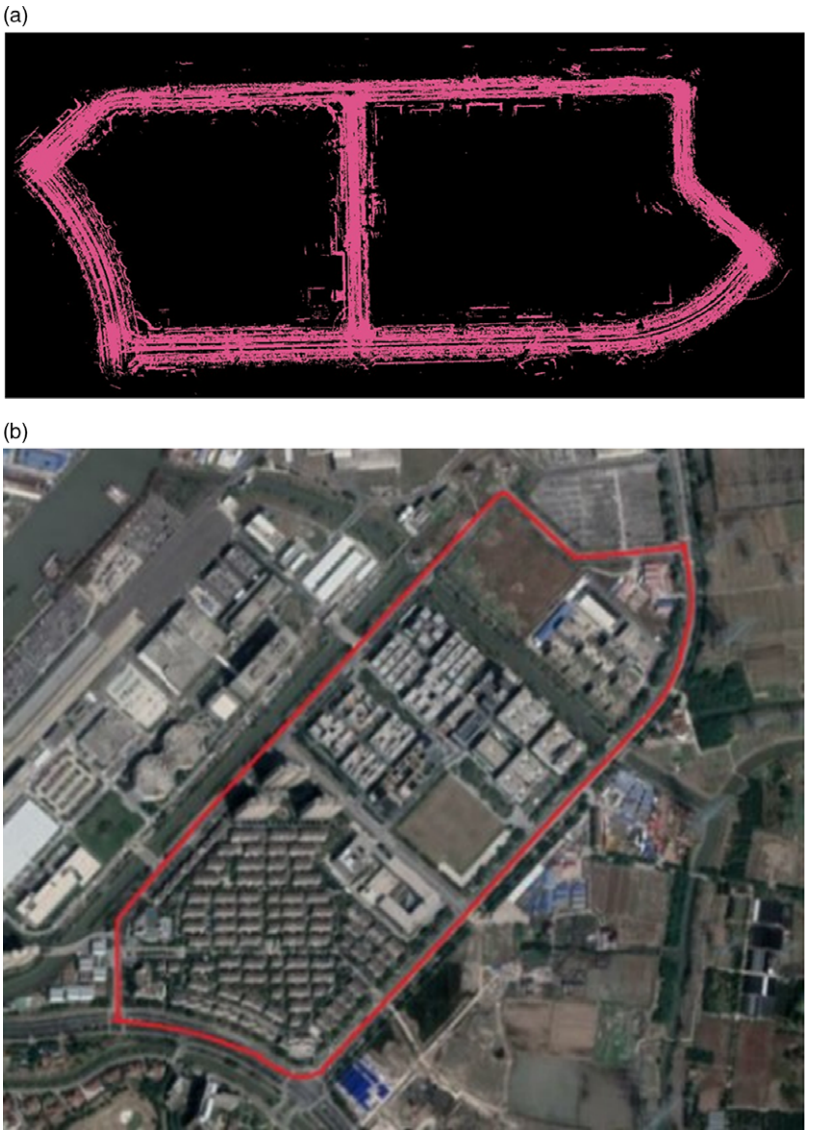


Figure 5. *The 3D point cloud map generated by our pose graph mapping method based on adaptive information matrix. (a) The 3D point cloud map. (b) Google map.*

demonstrates that our mapping method is effective. Figure 5 is another example of the 3D point-cloud map generated by our method.

3.2. 3D curvature features – Monte Carlo localization

The 3D point cloud map, which is available from Section 3.1, will be used as a prior map to achieve localization in this section. This section will describe our localization algorithm, 3DCF-MCL. 3DCF-MCL is composed of three parts: initialization, prediction, and updating. The main contribution of our algorithm in localization is the updating method. Thus, we will describe the updating method in detail. To complete our localization algorithm, we also briefly introduce the process of initialization and the prediction. The flow diagram of 3DCF-MCL is shown in Fig. 1.

https://doi.org/10.1017/S0263574721000862 Published online by Cambridge University Press

3.2.1. Initialization and prediction

For the initialization, this paper uses the GNSS data whose signal is good to provide the initial autonomous vehicle pose $p_{gnss0}(x, y, yaw)$ in the world coordinate system. Then, it initializes the particles with a Gaussian distribution $p_0 \in N(p_{gnss0}, \sigma^2_{x,y,yaw})$ (this paper sets $\sigma^2_x = \sigma^2_y = \sigma^2_{yaw} = 1.0$). As for the prediction, we use Odometry Motion Model [36] as our vehicle motion model, and the wheel odometry is obtained by wheel encoder information. Given that the distance between two adjacent frames is short enough, the wheel odometry is sufficiently accurate during this period. Therefore, using wheel odometry in prediction phase is valid. Furthermore, Sample Motion Model Odometry Algorithm [36] (see [36] Chapter 5.4 the sample_motion_model_odometry algorithm) is applied to the process of the prediction. This paper sets $\alpha_1 = 0.03, \alpha_2 = 0.01, \alpha_3 = 0.1, \alpha_4 = 0.0001$ in the sample_motion_model_odometry algorithm. Notably, we can calculate (x, y, yaw) from the transformation matrix [34]. The system dynamics of the autonomous vehicle is a two-wheeled differentially driven nonholonomic mobile robot [37].

3.2.2. Updating

In this part, we first describe the principle of updating the weight of particles. Then, we describe the calculation of the autonomous vehicle pose.

Update the weights of particles. We assume that t_k is the starting time of the kth frame, t_{k+1} is the ending time of the kth frame and P_k represents the point clouds obtained during $(t_k, t_{k+1}]$. Then we extract feature points using the same method as LOAM. We assume that P_m is the local prior map point cloud which is around the autonomous vehicle at t_{k+1}. In order to quickly find the corresponding feature lines or feature planar patches, which are in P_m for the feature points extracted from P_k, we store P_m into a 3D $KDtree_{pm}$ [38]. More specifically, to find the corresponding feature line for a edge feature point p_i which is in P_k, we first search its nearest point $p_j (p_j \in P_m)$ in $KDtree_{pm}$. Then we search the nearest point $p_l (p_l \in P_m)$ of p_j in the different scan lines. Finally, (p_j, p_l) forms a straight line, that is L. The distance from the edge feature point p_i to line L is defined as

$$d_e = \frac{\left| \left(X_{p_i} - X^m_{p_j} \right) \times \left(X_{p_i} - X^m_{p_l} \right) \right|}{\left| X^m_{p_j} - X^m_{p_l} \right|} \tag{8}$$

where X_{p_i}, $X^m_{p_j}$, and $X^m_{p_l}$ are the 3D coordinates of points p_i, p_j, and p_l. The m in the superscript of $X^m_{p_j}$ and $X^m_{p_l}$ represents $p_j, p_l \in P_m$ (m means the map). For a flat feature point $p_i \in P_k$, similar to the edge feature point, we search its nearest point $p_j (p_j \in P_m)$ in $KDtree_{pm}$. Then, we search the nearest point in the same scan line $p_l (p_l \in P_m)$ for p_j. Finally, we search the nearest point in the consecutive scan line $p_n (p_n \in P_m)$ for p_j. (p_j, p_l, p_n) forms a surface patch S. The distance from the flat feature point p_i to S is defined as

$$d_f = \frac{\left| \left(X_{p_i} - X^m_{p_j} \right) \cdot \left[\left(X^m_{p_j} - X^m_{p_l} \right) \times \left(X^m_{p_j} - X^m_{p_n} \right) \right] \right|}{\left| \left(X^m_{p_j} - X^m_{p_l} \right) \times \left(X^m_{p_j} - X^m_{p_n} \right) \right|} \tag{9}$$

where X_{p_i}, $X^m_{p_j}$, $X^m_{p_l}$, and $X^m_{p_n}$ is the 3D coordinates of points p_i, p_j p_l, and p_n. $(X_{p_i} - X^m_{p_j})$ is a vector composed of point p_i and inner point p_j of the plane S. $(X^m_{p_j} - X^m_{p_l}) \times (X^m_{p_j} - X^m_{p_n})$ is the normal vector of plane S. As we all know, the distance from a point to a plane is the projection of the vector (which is composed of this point to any point in the plane) on the normal vector of this plane. Thus, Eq. (9) is the distance from the point p_i to the plane S. Assuming that M edge feature points and N flat feature points exist in P_k, the sum of the distances for all the edge feature points in P_k can be calculated as

$$d_{edge} = \sum_{m=1}^{M} (d_e)_m \tag{10}$$

https://doi.org/10.1017/S0263574721000862 Published online by Cambridge University Press

where d_e is defined in Eq. (8). The sum of the distances for all the flat feature points in P_k can be calculated as

$$d_{flat} = \sum_{n=1}^{N} (d_f)_n \tag{11}$$

where d_f is defined in Eq. (9). The total distance d can be calculated as

$$d = \alpha d_{edge} + \beta d_{flat} \tag{12}$$

where α and β are the weights for d_{edge} and d_{flat}, $\alpha + \beta = 1$. Given that edge feature points have more structural information, which is highly suitable for estimating the vehicle poses, we put more weight on d_{edge} ($\alpha > \beta$). The weight of each particle can be updated as follows:

$$\omega_i \propto \frac{1}{d_i} \tag{13}$$

where ω_i is the weight of the ith particle. d_i is the total distance calculated by Eq. (12) with the assumption that the vehicle pose is the same as the pose of the ith particle.

Calculate the vehicle pose. The preliminary vehicle pose can be calculated by density clustering:

$$pose = \sum_{i=1}^{N} \omega_i p_i \tag{14}$$

where *pose* is the preliminary vehicle pose. Notably, the preliminary pose means that we will update it later using Eqs. (15) and (16). ω_i is the weight of the ith particle calculated by Eq. (13) and p_i is the corresponding pose of the ith particle. Assuming that $T_k = pose$, we calculate the total distance d between each feature point to its corresponding feature line or feature planar patch according to Eqs. (8) and (9). Finally, we obtain a nonlinear function:

$$d = f(T_k) \tag{15}$$

where the fuction f represents the functional relationship between T_k and d. We calculate the Jacobian matrix of f with respect to T_k, $J = \frac{\partial f}{\partial T_k}$. Then, we can solve Eq. (15) through nonlinear iterations by minimizing d to zero:

$$T_k \leftarrow T_k - (J^T J + \lambda diag(J^T J))^{-1} J^T d \tag{16}$$

where λ is a factor determined by the Levenberg–Marquardt method. The iteration will stop when the number of iterations exceeds a threshold (i.e., 5–10). Finally, T_k is the final autonomous vehicle pose.

As described in this Section (Section 3), this paper proposes a pose graph mapping method based on the adaptive information matrix using LiDAR and GNSS data and 3DCF-MCL. The relationship between 3DCF-MCL and the traditional MCL are twofold. The similarity is described in Section 3.2.1. 3DCF-MCL and traditional MCL use the same method in the initialization and prediction phases. The difference is the updating phase (Section 3.2.2): the traditional MCL uses a 2D prior map and carries out 2D point cloud registration between the 2D prior map and the point cloud collected by the LiDAR. 3DCF-MCL builds a 3D prior map using the proposed pose graph mapping method and extracts edge feature and flat feature points. Then, it carries out 3D point cloud registration between the 3D prior map and the point cloud collected by the LiDAR. Compared with the traditional MCL, the 3D map and the edge feature and flat feature points in 3DCF-MCL can provide more constraints on the pose estimation, which will improve the accuracy of the pose estimation.

https://doi.org/10.1017/S0263574721000862 Published online by Cambridge University Press

Table I. *3DCF-MCL localization error.*

Data	Distance(m)	APE mean(m)
data1	200.23	0.36
data2	2000	0.32
data3	604.491	0.42

4. Experiments

We collected outdoor data using Velodyne-16. The wheel speed sensors provide the source data of the wheel odometry, and GNSS data were acquired from RTK-GPS (real-time kinematic). The LiDAR is placed on the top of a vehicle. During experiments, the algorithms ran on a laptop computer with 2.2 GHz main frequency and 16 GB RAM, on Robot Operating System (ROS) [39] in Linux.

4.1. Results and analysis of the localization

The paper used real-world data collected in outdoor environment to verify the performance of our proposed algorithm. To generate the 3D point cloud map with good quality, the vehicle moved at a slow speed, which was approximately 10 km/h. To calculate the accuracy of 3DCF-MCL, we collected the second set of LiDAR data in the same roads at approximately 40 km/h, which was the normal speed of the car driving on the urban road. During the data collection process, the vehicle trajectory for localization was different from the vehicle trajectory for mapping. To evaluate the accuracy of the localization, we used the optimized poses calculated by pose graph as the ground truth. Notably, when the road around the autonomous vehicle was empty and not blocked by tall buildings, the ground truth and GNSS measurement were almost the same, as the quality of GNSS signals was sufficient. However, when the GNSS signals were blocked by tall buildings or trees, the ground truth calculated by the pose graph was useful. Furthermore, in some special situations, GNSS could not provide accurate localization for the vehicle (e.g., the signals of satellites were blocked by tall buildings or trees). However, our method was able to provide accurate localization.

4.1.1. GNSS signal is good

Figure 6(a) is the vehicle trajectory with a driving distance of 1096.993 m. Figure 6(b) is the result of 3DCF-MCL. Figure 6(c) demonstrates that the trajectory obtained by 3DCF-MCL almost coincide with the ground truth. Furthermore, the Absolute Pose Error (APE) [40] of the trajectory was used to evaluate the accuracy of localization. Figure 6(d), (e) show the statistics of APE. It can be seen from the represented results that the mean error of 3DCF-MCL is approximately 0.28 m.

Figure7(a) is the vehicle trajectory with a driving distance of 181.895 m. Figure 7(b) is the result of 3DCF-MCL. It can be seen from Fig. 7(c) that the trajectory obtained by our localization algorithm almost coincides with the ground truth. Furthermore, Fig. 7(d) and (e) show the statistics of the APE. The represented results show that the mean error of 3DCF-MCL is approximately 0.30 m. Table I also shows the performance of 3DCF-MCL.

4.1.2. GNSS signal is weak

In Fig. 8(a), it shows a test environment where the road is narrow and there are many tall buildings on the both sides of the road. In Fig. 9(a), it shows a test environment where the road is narrow and there are many tall trees on the both sides of the road. Figures 8(b) and 9(b) demonstrate that the localization of GNSS is intermittent due to the occlusion of buildings and trees (red trajectory). However, 3DCF-MCL can locate the vehicle accurately (the green trajectory).

https://doi.org/10.1017/S0263574721000862 Published online by Cambridge University Press

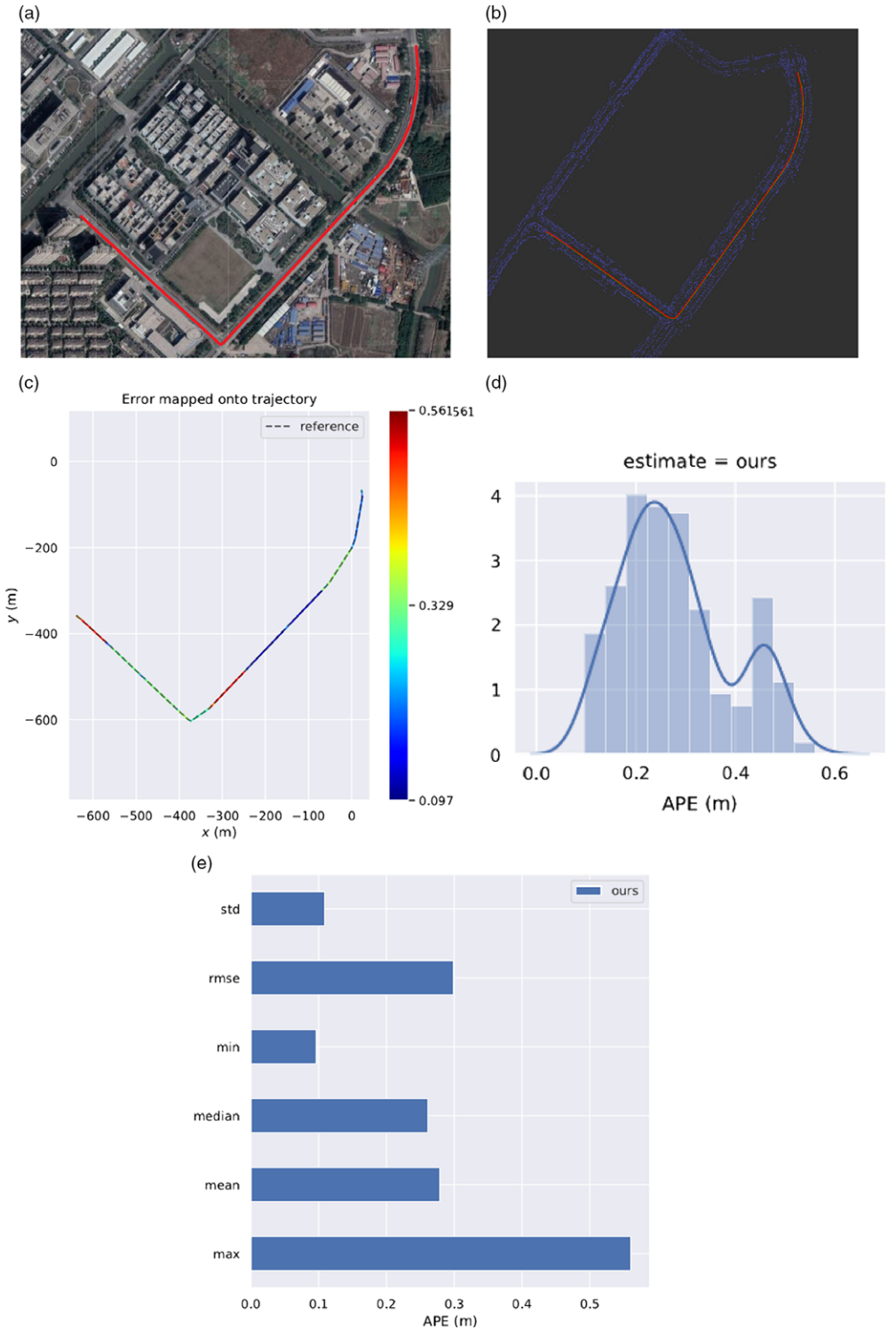


Figure 6. *The visualization of 3DCF-MCL error. (a) The driving trajectory. (b) The result of 3DCF-MCL. (c) The APE of 3DCF-MCL is mapped onto the trajectory (m) compared with ground truth. (b)(c) The statistics of APE. (a) Google map. (b) The result of 3DCF-MCL. (c) APE Error is mapped onto the trajectory. (d) The histogram of 3DCF-MCL APE. (e) The statistics of 3DCF-MCL APE.*

https://doi.org/10.1017/S0263574721000862 Published online by Cambridge University Press

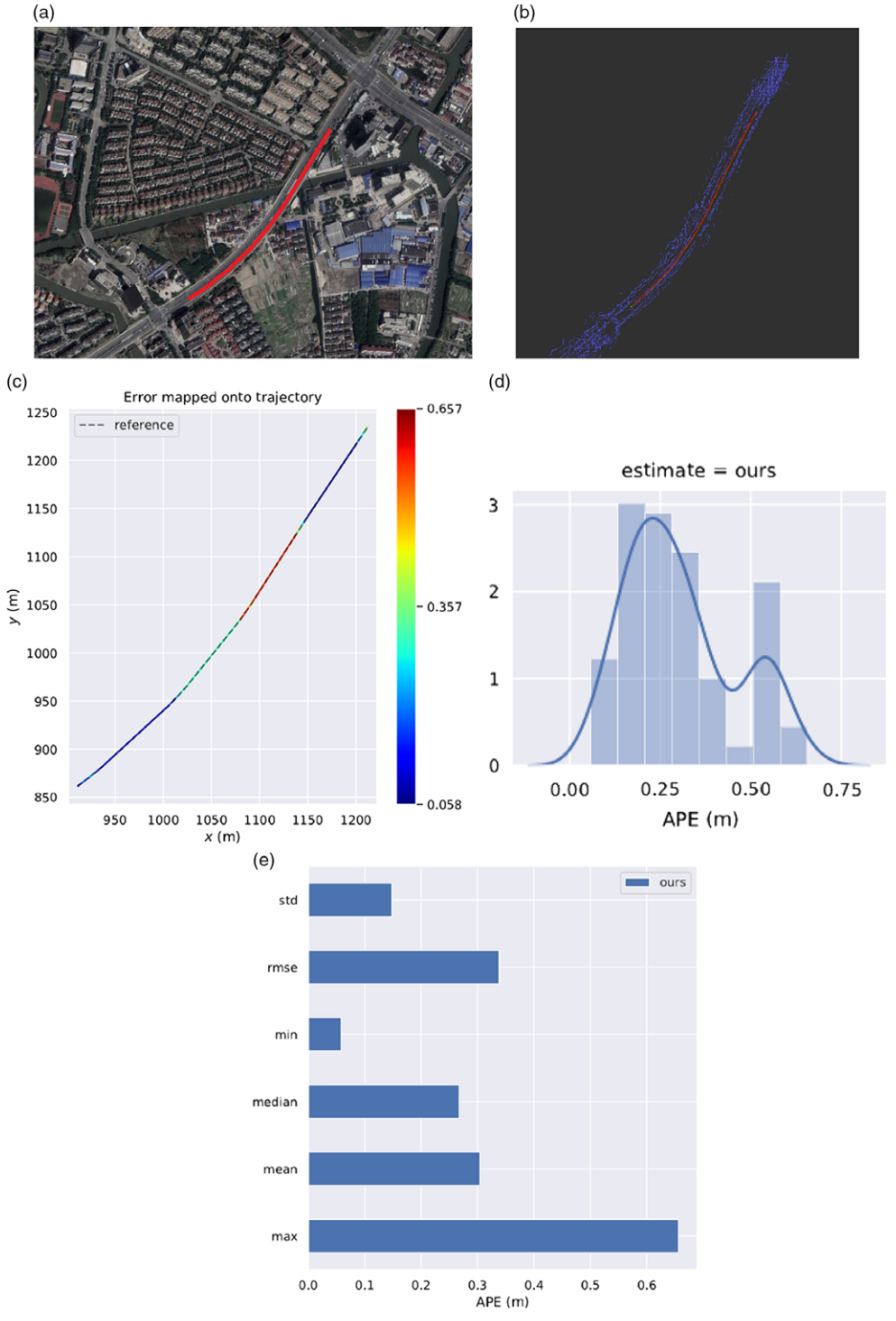


Figure 7. *The visualization of 3DCF-MCL error. (a) The driving trajectory. (b) the result of 3DCF-MCL (c) The APE of 3DCF-MCL is mapped onto the trajectory (m) compared with ground truth. (b)(c) The statistics of the APE. (a) Google map. (b) The result of 3DCF-MCL. (c) APE Error is mapped onto the trajectory. (d) The histogram of 3DCF-MCL APE. (e) The statistics of 3DCF-MCL APE.*

https://doi.org/10.1017/S0263574721000862 Published online by Cambridge University Press

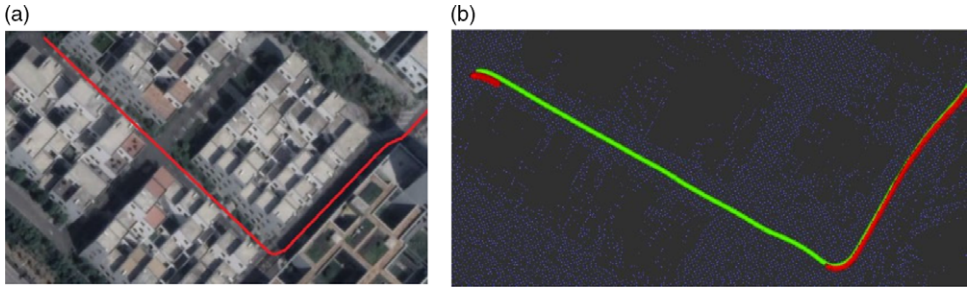


Figure 8. *The result of the localization where the road is narrow and there are many tall buildings on the both sides of the road. (a) Google map. (b) The result of 3DCF-MCL.*

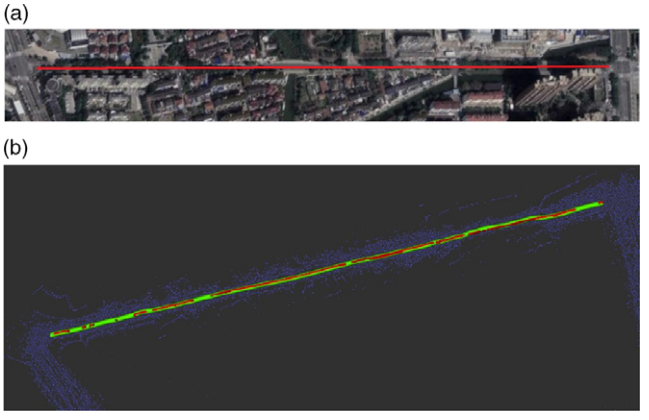


Figure 9. *The result of the localization where the road is narrow and there are many tall trees on the both sides of the road. (a) Google map. (b) The result of 3DCF-MCL.*

4.1.3. Localization robustness analysis

As described in our LiDAR data collection process, we first collected data at approximately 10 km/h to generate a map. Then, we collected the second set of LiDAR data in the same roads at approximately 40 km/h. During the data collection process, the vehicle trajectory for localization was different from the vehicle trajectory for mapping. The two periods of data collection for the same road did not occur on the same day. Hence, the road conditions were also completely different. However, the experimental results in this paper showed that 3DCF-MCL could still locate the vehicle accurately.

As described in Sections 4.1.1, 4.1.2, and 4.1.3, we believe that the 3D prior map built by the proposed pose graph method (Section 3.1), and the combination of the high accuracy of 3D feature point cloud registration and the robustness of MCL (Section 3.2) make contributions to these high-quality performances. Figures 8(a) and 9(a) demonstrate that GNSS could not provide an accurate localization due to the occlusion of tall obstacles. However, 3DCF-MCL was able to locate the vehicle accurately with the built 3D prior map.

4.2. Comparison with other methods

We compared our localization method with ICP localization and NDT localization [7]. The results are shown in Fig. 10. The driving distance was approximately 5 km and the vehicle speed was approximately 40 km/h. The represented results of Fig. 10 demonstrated that the localization error of 3DCF-MCL was

https://doi.org/10.1017/S0263574721000862 Published online by Cambridge University Press

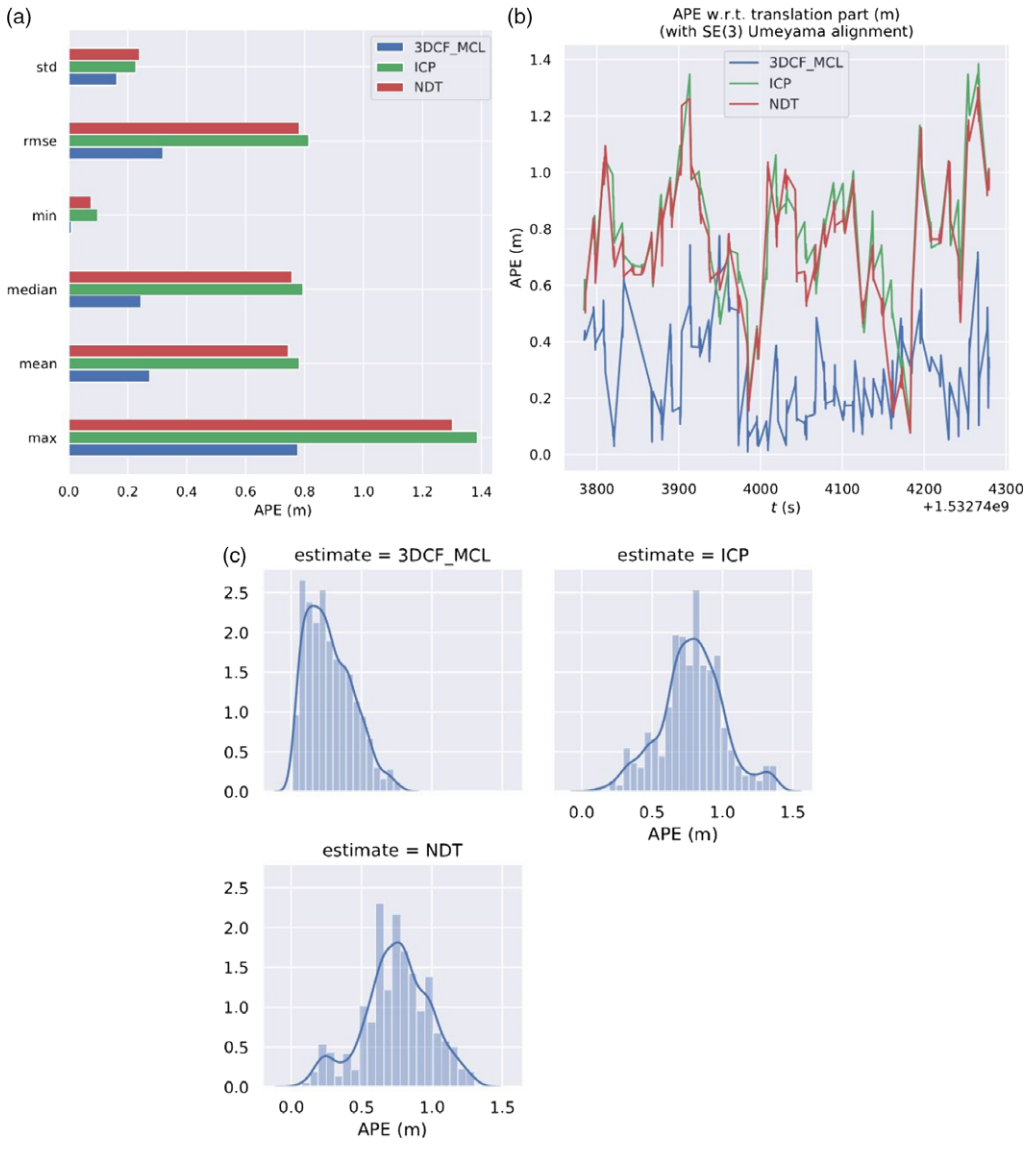


Figure 10. *The comparison of APE on 3DCF-MCL, NDT and ICP. (a) The statistics of APE. (b) The curve of APE. (c) The histograms of APE.*

the smallest compared with the ICP and NDT among various statistics. More specially, the mean error of the localization of 3DCF-MCL was approximately 0.25 m, which was less than the error of NDT and ICP. As shown in Fig. 10, we believe that the combination of the 3D feature point cloud registration and the robustness of MCL (Section 3.2) make contributions to these high-quality performances of 3DCF-MCL. The edge feature and flat feature points in 3DCF-MCL can provide more constraints on pose estimation, which improves the accuracy of pose estimation.

We did not compare LOAM with 3DCF-MCL in this research because the experimental results showed that the localization of LOAM drifted a lot due to the cumulative error when the driving distance was long. Nevertheless, 3DCF-MCL can still provide accurate localization with the prior map.

https://doi.org/10.1017/S0263574721000862 Published online by Cambridge University Press

5. Conclusion

In this paper, a complete localization framework for autonomous vehicles is proposed. This framework has two main contributions. First, a pose graph mapping method based on adaptive information matrix using LiDAR and GNSS data are proposed. Information matrices are constructed according to the results of the point cloud registrations. GNSS data are used as prior pose information, which improves the accuracy of the pose graph. Experimental results demonstrate that our mapping method is effective. Second, a novel localization algorithm based on 3D curvature feature points – Monte Carlo Localization is proposed, which combines the high accuracy of the 3D feature point registration and the robustness of particle filter. To validate our algorithm, we use realistic outdoor data to test our method. Experimental results demonstrate that 3DCF-MCL can provide accurate and robust localization for autonomous vehicles with 3D point cloud map generated by our method. The localization error is approximately 0.3m. Compared with NDT and ICP LiDAR-based localization algorithms, the localization performance of 3DCF-MCL performs better. However, if more moving objects are present in the test environments, the performance of the mapping and the localization will degrade. In further research, we will attempt to solve the above problems.

Acknowledgments. This work was supported by the National Natural Science Foundation of China under Grant 61775048 and Shenzhen basic research program JCYJ20180507183837726.

References

[1] L. Heng, T. Walter, P. Enge and G. X. Gao, "GNSS multipath and jamming mitigation using high-mask-angle antennas and multiple constellations," *IEEE Trans. Intell. Transp. Syst.* **16**(2), 741–750 (2015).
[2] S.-H. Kong, "Statistical analysis of urban gps multipaths and pseudo-range measurement errors," *IEEE Trans. Aerospace Electr. Syst.* **47**(2), 1101–1113 (2011).
[3] E. Yurtsever, J. Lambert, A. Carballo and K. Takeda, "A survey of autonomous driving: Common practices and emerging technologies," *IEEE Access* **8**, 58443–58469 (2020). doi: 10.1109/ACCESS.2020.2983149
[4] E. Javanmardi, Y. Gu, M. Javanmardi and S. Kamijo, "Autonomous vehicle self-localization based on abstract map and multi-channel lidar in urban area," *IATSS Res.* **43**(1), 1–13 (2019).
[5] C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, I. Reid and J. J. Leonard, "Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age," *IEEE Trans. Rob.* **32**(6), 1309–1332 (2016).
[6] J.-E. Deschaud, "IMLS-SLAM: Scan-to-Model Matching Based on 3D Data," *2018 IEEE International Conference on Robotics and Automation (ICRA)* (IEEE, 2018) pp. 2480–2485.
[7] S. Kato, S. Tokunaga, Y. Maruyama, S. Maeda, M. Hirabayashi, Y. Kitsukawa, A. Monrroy, T. Ando, Y. Fujii and T. Azumi, "Autoware on Board: Enabling Autonomous Vehicles with Embedded Systems," *2018 ACM/IEEE 9th International Conference on Cyber-Physical Systems (ICCPS)* (IEEE, 2018) pp. 287–296.
[8] Y. Zhang, L. Wang, X. Jiang, Y. Zeng and Y. Dai, "An efficient lidar-based localization method for self-driving cars in dynamic environments," *Robotica*, 1–18 (2021). doi: 10.1017/S0263574721000369
[9] R. Kümmerle, G. Grisetti, H. Strasdat, K. Konolige and W. Burgard, "G2o: A General Framework for Graph Optimization," *2011 IEEE International Conference on Robotics and Automation* (IEEE, 2011) pp. 3607–3613.
[10] K. Koide, J. Miura and E. Menegatti, "A portable 3d lidar-based system for long-term and wide-area people behavior measurement." *Int. J. Adv. Robot. Syst.* **16**(2), (2018).
[11] G. Grisetti, R. Kummerle, C. Stachniss and W. Burgard, "A tutorial on graph-based SLAM," *IEEE Intell. Transp. Syst. Mag.* **2**(4), 31–43 (2010).
[12] P. J. Besl and N. D. McKay, "Method for Registration of 3-D Shapes," **In:** *Sensor Fusion IV: Control Paradigms and Data Structures*, vol. 1611 (International Society for Optics and Photonics, 1992) pp. 586–607.
[13] A. Segal, D. Haehnel and S. Thrun, "Generalized-ICP," **In:** *Robotics: Science and Systems*, vol. 2 (2009) p. 435.
[14] S. Hong, H. Ko and J. Kim, "VICP: Velocity Updating Iterative Closest Point Algorithm," *2010 IEEE International Conference on Robotics and Automation* (IEEE, 2010) pp. 1893–1898.
[15] J. Serafin and G. Grisetti, "NICP: Dense Normal Based Point Cloud Registration," *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (IEEE, 2015) pp. 742–749.
[16] ethz-asl. ethzasl_icp_mapping. `https://github.com/ethz-asl/ethzasl_icp_mapping` (2018).
[17] P. Biber and W. Straßer, "The Normal Distributions Transform: A New Approach to Laser Scan Matching," *Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003) (Cat. No. 03CH37453)*, vol. 3 (IEEE, 2003) pp. 2743–2748.

https://doi.org/10.1017/S0263574721000862 Published online by Cambridge University Press

[18] M. Magnusson, The Three-Dimensional Normal-Distributions Transform: An Efficient Representation for Registration, Surface Analysis, and Loop Detection *Ph.D. Thesis* (Örebro universitet, 2009).
[19] M. Magnusson, A. Nuchter, C. Lorken, A. J. Lilienthal and J. Hertzberg, "Evaluation of 3D Registration Reliability and Speed-a Comparison of ICP and NDT," *2009 IEEE International Conference on Robotics and Automation* (IEEE, 2009) pp. 3907–3912.
[20] J. Zhang and S. Singh, "LOAN: Lidar Odometry and Mapping in Real-Time," **In:** *Robotics: Science and Systems*, vol. 2 (2014) p. 9.
[21] H. Moravec and A. Elfes, "High Resolution Maps from Wide Angle Sonar," *Proceedings. 1985 IEEE International Conference on Robotics and Automation*, vol. 2 (IEEE, 1985) pp. 116–121.
[22] K. M. Wurm, C. Stachniss and G. Grisetti, "Bridging the gap between feature-and grid-based SLAM," *Rob. Auto. Syst.* **58**(2), 140–148 (2010).
[23] T. Bailey, *Mobile Robot Localisation and Mapping in Extensive Outdoor Environments* (Citeseer, 2002).
[24] J. Boal, Á. S. Miralles and A. Arranz, "Topological simultaneous localization and mapping: A survey," *Robotica* **32**(5), 803–821 (2014).
[25] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss and W. Burgard, "OctoMap: An efficient probabilistic 3d mapping framework based on octrees," *Auto. Robots* **34**(3), 189–206 (2013).
[26] E. Nelson, BLAM. https://github.com/erik-nelson/blam (2016).
[27] R. Dubé, D. Dugas, E. Stumm, J. Nieto, R. Siegwart and C. Cadena, "Segmatch: Segment Based Place Recognition in 3D Point Clouds," *2017 IEEE International Conference on Robotics and Automation (ICRA)* (IEEE, 2017) pp. 5266–5272.
[28] E. Mendes, P. Koch and S. Lacroix, "ICP-based Pose-Graph SLAM," *2016 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)* (IEEE, 2016) pp. 195–200.
[29] M. Kaess, H. Johannsson, R. Roberts, V. Ila, J. J. Leonard and F. Dellaert. "ISAM2: Incremental smoothing and mapping using the bayes tree," *Int. J. Rob. Res.*, **31**(2), 216–235 (2012).
[30] Karlsruhe Institute of Technology. kitti_odometry. http://www.cvlibs.net/datasets/kitti/eval_odometry.php (2020).
[31] F. Dellaert, D. Fox, W. Burgard and S. Thrun, "Monte Carlo Localization for Mobile Robots," *ICRA*, vol. 2 (1999) pp. 1322–1328.
[32] Y. Chen, W. Chen, L. Zhu, Z. Su, X. Zhou, Y. Guan and G. Liu, "A study of sensor-fusion mechanism for mobile robot global localization," *Robotica* **37**(11), 1835–1849 (2019).
[33] D. Chen, J. Weng, F. Huang, J. Zhou, Y. Mao and X. Liu, "Heuristic monte carlo algorithm for unmanned ground vehicles realtime localization and mapping," *IEEE Trans. Veh. Technol.* **69**(10), 10642–10655 (2020).
[34] T. D. Barfoot, *State Estimation for Robotics* (Cambridge University Press, Cambridge, UK, 2017).
[35] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision* (Cambridge University Press, UK, 2003).
[36] S. Thrun, W. Burgard and D. Fox, *Probabilistic Robotics* (MIT Press, Cambridge, MA, 2005).
[37] F. C. Vieira, A. A. D. Medeiros, P. Alsina and A. P. Araújo, "Position and Orientation Control of a Two-Wheeled Differentially Driven Nonholonomic Mobile robot," *ICINCO* (2004).
[38] M. Berg, M. Kreveld, M. Overmars and O. C. Schwarzkopf, Computational geometry: Algorithms and applications (Springer Berlin/Heidelberg, Berlin, Heidelberg, 2008).
[39] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler and A. Y. Ng, "ROS: An Open-Source Robot Operating System," *ICRA Workshop on Open Source Software*, vol. 3, Kobe, Japan (2009) p. 5.
[40] M. Grupp, evo: Python package for the evaluation of odometry and SLAM. https://github.com/MichaelGrupp/evo (2017).

Cite this article: Q. Liu, X. Di and B. Xu (2022). "Autonomous vehicle self-localization in urban environments based on 3D curvature feature points – Monte Carlo localization", Robotica **40**, 817–833. https://doi.org/10.1017/S0263574721000862

https://doi.org/10.1017/S0263574721000862 Published online by Cambridge University Press