文章编号：0253-374X(2021)S1-0174-12 **DOI**：10.11908/j.issn.0253-374x.22721

融合码盘和激光雷达的里程计与建图

陈贤钦，陈　慧
（同济大学 汽车学院，上海 201804）

摘要：提出了一种用于自动驾驶汽车的低漂移、低延迟的里程计与高精度建图的算法。该方法融合了多种传感器的测量结果，包括车轮编码器、转向盘转角编码器、激光雷达及可选GPS等的测量结果。里程计算法由车轮里程计和激光里程计组成：前者基于车辆运动学模型，高频、实时估计位姿增量，用于点云去畸变和为后者优化位姿提供可用的初值；后者以较低的频率估计车辆的精确位姿变化，以补偿前者累计的误差，其核心是一种基于角度度量的两阶段特征提取方法。建图算法基于因子图，包含激光里程计因子、回环因子和可选GPS因子，通过增量平滑和建图算法优化全局轨迹，在线生成全局地图，其中GPS因子能够自动对齐GPS坐标系和里程计坐标系，逐步融合GPS测量值，解除了算法初始化过程对于GPS的依赖。所提出的方法在自动驾驶汽车平台数据集上进行了评估，并和已开源的部分相关工作进行对比，结果表明它具有更低的漂移率，在本文进行的最大规模的测试中达到了0.53%。相关代码以开源形式供交流参考(https://github.com/Saki-Chen/W-LOAM)。

关键词：自动驾驶汽车；同时定位与建图；激光里程计；车轮里程计；因子图优化；多传感器融合
中图分类号：U495；TP391　　**文献标志码**：A

Wheel-LiDAR Odometry and Mapping for Autonomous Vehicles

CHEN Xianqin, *CHEN Hui*
(School of Automotive Studies, Tongji University, Shanghai 201804, China)

Abstract: This paper, by proposing a wheel-LiDAR method of odometry and mapping (WLOAM), using wheel encoder, steering encoder, LiDAR, and optional GPS for autonomous vehicles, estimates the low-drift pose at real-time and builds a high-accurate map. The odometry consists of the wheel odometry algorithm and the LiDAR odometry algorithm. The former estimates the 3-DOF ego-motion of LiDAR at a high frequency based on Ackermann steering geometry, whose resulting pose increment is applied in point clouds de-skewing and works as a fine initial guess for LiDAR odometry while the latter performs the 6-DOF scan-to-map LiDAR pose optimization at a relatively low frequency to compensate the pose error accumulated by the wheel odometry, whose core is a two-stage method with an angle-based metric for extracting features. The mapping method is based on the factor graph consisting of the LiDAR odometry factor, the loop closure factor, and the optional GPS factor, which is solved via incremental smoothing and mapping (iSAM) to produce a global map online. An auto-aligned-GPS-factor is proposed for fusing GPS measurement incrementally without explicit initialization. The proposed method was extensively evaluated on the datasets gathered from the autonomous vehicle platform and compared with related open-sourced works. The results show a lower drift rate, which reaches 0.53% in the largest test described in this paper. The implementation of the proposed method is open-sourced for communication (https://github.com/Saki-Chen/W-LOAM).

Key words: autonomous vehicles; simultaneous localization and mapping; LiDAR odometry; wheel odometer; factor graph optimization; sensor fusion

1　Introduction

Simultaneous Localization and Mapping (SLAM) has been researched for the last 30 years but remains a popular topic among the field of robotics[1]. Recently, the development of autonomous vehicle brings new sensors, new scenarios and new challenges for SLAM[2]. One of

收稿日期：2021-09-20
第一作者：陈贤钦(1994—)，男，硕士研究生，主要研究方向为多传感器融合定位与建图和激光检测。E-mail：1933510@tongji.edu.cn
通信作者：陈慧(1964—)，男，教授，博士生导师，工学博士，主要研究方向为汽车底盘电子控制系统技术和智能汽车技术。
E-mail：hui-chen@tongji.edu.cn

the most important challenge is self-localization, which is also the basic problem of SLAM, but in a larger scale with various challenges from the real world like lighting, weather, GPS signal quality, etc. As it is difficult to build a SLAM system with only a single type of sensor handling all these challenges, fusing method stands out, which takes advantages of different types of sensors. Reference [3] summarizes different sensors and their combinations like visual inertial method and LiDAR inertial method, which are among the most popular SLAM methods, to perform self-localization for autonomous vehicles. In this paper, wheel encoder, steering encoder, LiDAR and an optional GPS are picked out for building SLAM system on car-like platforms.

It is desirable to design an accurate, robust and real-time SLAM system with carefully designed architecture for autonomous vehicles. The LiDAR odometry and mapping (LOAM) method proposes a good example of that kind using dual-layer optimization to achieve high frequency LiDAR odometry and low frequency LiDAR mapping as well as correction to odometry[4]. But the frequency of odometry is limited by the frame rate of LiDAR. The LiDAR inertial odometry via smoothing and mapping (LIO-SAM) method fuses inertial measurements to produce higher output frequency of odometry, which also serves to de-skew point cloud without a linear motion assumption compared to LOAM, allowing it to outperform LOAM especially when sensor moves or rotates violently[5]. But it works in a tightly-coupled framework with feed-back between inertial module and LiDAR odometry module making it fragile when good features of point cloud are rare. However, LIO-SAM demonstrates a good practice that LiDAR odometry is enhanced with another supporting sensor predicting motion between laser scans at a high frequency. For autonomous vehicles, wheel encoder and steering encoder, which are equipped for most automobiles, are qualified to play this supporter role with fast and robust wheel odometry taking place of inertial measurement unit (IMU), despite the fact that the wheel odometry only estimates 3-DOF motion and drift quickly when turning[6].

Feature extraction lies in the core of feature-based LiDAR odometry method determining the accuracy, robustness and even computational efficiency. Reference [4-5] and [7] all follow a general procedure including classifying points as edge points or planar points, searching for neighbor points in corresponding feature map, registering points to map by minimizing point-to-line or point-to-plane distance and finally merge points into feature maps. The feature extraction in this procedure is based on geometric criterion, which are calculated within a small piece of laser scan for taking advantage of dense points along the scanning direction of LiDAR. This method works well for most planar object like wall or ground resulting smooth planar feature map, but suffers from some rough object like bush or grass producing noisy edge feature map. Thus, most methods filter out bad edge features according to the spatial distribution of neighbor points. But, actually, some of these abandoned features, for example, feature points of a greensward, can be treated as planar features after voxel grid filtering because they look planar within a bigger neighborhood rather than a small piece of laser scan. This observation inspires an idea called degenerated features, which are extracted from edge features but work as a planer feature with lower weight.

Another important issue for the system is correction of drift. One possible technique is applying loop-closure, which is widely used in SLAM. Reference [5] and [7] implement an iterative-closest-point-based (ICP) method for loop-closure, but it would easily fail when the drift goes too big. A simpler and more practical technique for autonomous vehicles is fusing GPS measurements to limit the growing of drift, which also helps to improve the performance of ICP-based loop-closure. Reference [5] adopts both of these techniques with a single factor graph, which is optimized via iSAM[8-9] efficiently. However, an extra orientation measurement is required for alignment of the GPS coordinate system, which is helpful to improve accuracy of orientation but redundant for fusing GPS

measurements. In fact, two coordinate system can be aligned using only path points measured in each coordinate system using Umeyama's algorithm [10]. Thus, it is also possible to align a pose graph, which is treat as a set of path point, to corresponding path points set of GPS measurements through optimization method, where the drift of every pose is minimized concurrently. This idea is implemented as Auto-Aligned-GPS-factor, which estimates the transformation from the local coordinate to the GPS coordinate and correct drift of poses at the same time.

In summary, this paper proposed a SLAM system consisting of three layers for performing odometry and mapping online. In the first layer, a wheel odometry method based on Ackermann steering geometry is introduced to output high frequency ego-motion in real time, which serves as a fine initial guess of accurate pose and is used for point cloud de-skewing. In the second layer, an improved two-stage feature-based method with an angle-based metric is applied to extract edge features, planar features and degenerated features from point cloud, which process features respectively in local scan scale and local map scale for more robust feature extraction. The features are then parsed to form constraints to the sensor pose in a scan-to-map manner for LiDAR odometry optimization. In the third layer, a graph-based method is applied to formulate a factor graph optimization problem with LiDAR odometry, loop closure and optional GPS measurement, which is solved via iSAM. Here, an Auto-Aligned-GPS-factor for fusing GPS measurements without explicit initialization is proposed for large scale mapping task. With the three-layer framework, odometry runs with very low delay in the first layer and the accumulated error is compensated by the LiDAR odometry optimization in the second layer. And mapping runs in the third layer with lower demand of real-time processing. The main contributions of this paper are summarized as follows:

(1) A robust and real-time front end based on Ackermann steering geometry for car-like platforms.

(2) An improved two-stage feature-based method for scan matching: an angle-based metric for parsing point cloud in first stage, and a principal-component-analysis-based (PCA) method for extracting edge features, planar features and degenerated features in second stage.

(3) Auto-Aligned-GPS-factor.

The following chapters are arranged as follows: The Method chapter presents an overview of system firstly and then makes detailed description of each module. The experiments chapter introduces the benchmark results of proposed algorithm using datasets gathered from our autonomous vehicle platform. The Conclusion chapter reviews main contributions of this work and prospects the future work.

2 Method

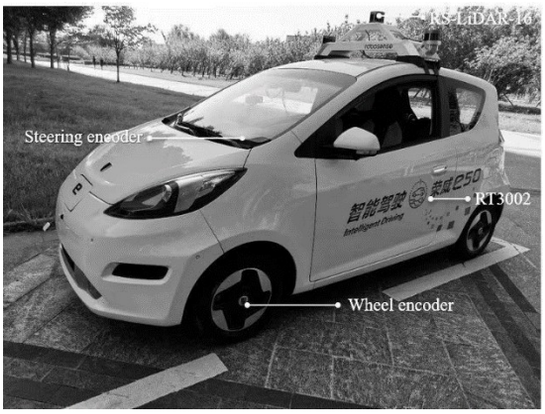
2.1 Notations

As a convention in this paper, ${}_{target}^{t}T_{source}$ represents the 6-DOF pose of frame *source* with respect to frame *target* at time t. ${}_{source}^{t}p$ represents point p with timestamp t expressed in the frame *source*. And the point can be transformed to the frame *target* by applying ${}_{target}^{t}T_{source} \cdot {}_{source}^{t}p$, which results ${}_{target}^{t}p$.

2.2 System overview

The proposed algorithm is validated using data collected from the autonomous vehicle platform shown in Fig. 1. The steering encoder and wheel encoder provide 100 Hz output though controller area network (CAN). The mounted central LiDAR provides 10 Hz 16-channel data with a field of view (FOV) by 360° ×30°, where the horizontal resolution is 0.2° and vertical resolution is 2.0°. The RT 3002 provides an optional GPS measurement for the proposed algorithm and ground truth data for validation when real-time kinematic (RTK) is available.

The overall system consists of five modules illustrated by Fig. 2, which receives point cloud data directly from LiDAR and reads encoder data and optional GPS measurement data through CAN. The system outputs optimized global trajectory and global point cloud map, which are both aligned to the GPS coordinate system if GPS measurement is available.



The steering encoder and wheel encoder for each wheel are the original parts of the vehicle, which are accessed through CAN. Three sets of RS-LiDAR-16 produced by Robosense are mounted atop the vehicle but only the central one is exploited for the proposed algorithm. An inertial and GPS measurement system, RT 3002, is mounted at the center of rear axle, which provides precise ground truth data for validation.

Fig.1　Autonomous vehicle platform and sensors setup

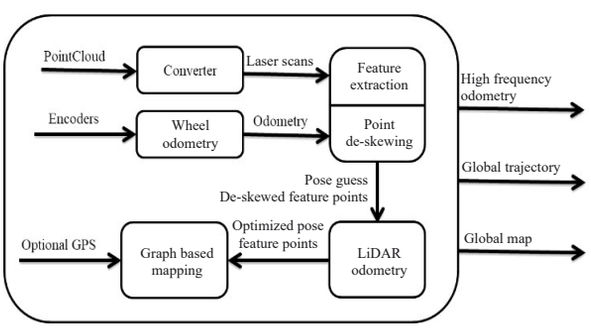
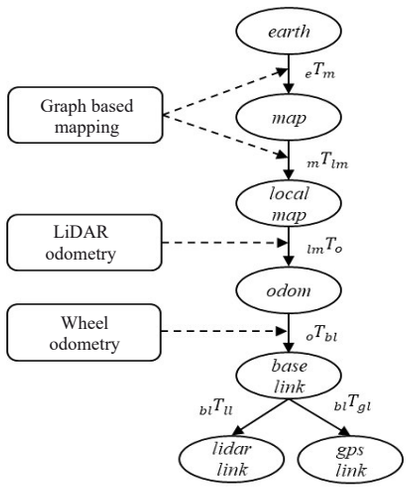


Fig. 2　Overview of WLOAM algorithm

The output of odometry is in the form of coordinate transformation tree provided by ROS[11], which makes it easy to decouple these modules. Fig. 3 illustrates the coordinate transformation tree and modules responsible for updating these transformations.

2.3　Wheel odometry

The kinematic model for performing wheel odometry is derived from Ackermann steering geometry shown in Fig. 4. Given turning radius R and velocity v_{wheel} at any single wheel, the linear velocity v at the center of rear axle is proportional to v_{wheel}, and the angular velocity $w=v\cdot c$, where c means curvature and $c\cdot R=1$. But the wheel encoder outputs count directly, which is associated with the rolling distance of wheel. Hence there is no need to



In Fig. 3, *earth* means the local coordinate system of GPS which can be ENU or UTM; *map* means the coordinate system for global map and global trajectory whose origin is located at the start point; *local map* means the coordinate system for LiDAR odometry, which drifts slowly as the system travels over a long distance; *odom* means the coordinate system for wheel odometry, which drift fast but has a low delay; *base link* is attached to the center of rear axle of vehicle shown by Fig.4; *lidar link* and *gps link* are attached to corresponding sensors. The arrows between circles are defined as transformations between coordinate systems, where ${}_eT_m$, ${}_mT_{lm}$, ${}_{lm}T_o$ and ${}_oT_{bl}$ are estimated by their corresponding modules while ${}_{bl}T_{ll}$ and ${}_{bl}T_{gl}$ are directly calibrated as constant. Other transformations not shown in Fig. 3 can be derived from combination like ${}_oT_{ll}={}_oT_{bl}\cdot{}_{bl}T_{ll}$.

Fig.3　Coordinate transformation tree

estimate velocity of wheel and compute distance, which introduce extra timing problem[12]. Instead, v is replaced by $\mathrm{d}s$, which means small increment of distance, w is replaced by $\mathrm{d}\theta$, which means small increment of yaw angle, so that $\mathrm{d}s$ and $\mathrm{d}\theta$ can be computed using Eqs. (1)–(5):

$$\mathrm{d}s=\frac{\mathrm{d}s_{fl}}{\sqrt{\left(1-\frac{D_f}{2}\cdot c\right)^2+(L\cdot c)^2}} \tag{1}$$

$$\mathrm{d}s=\frac{\mathrm{d}s_{fr}}{\sqrt{\left(1+\frac{D_f}{2}\cdot c\right)^2+(L\cdot c)^2}} \tag{2}$$

$$\mathrm{d}s=\frac{\mathrm{d}s_{rl}}{1-\frac{D_r}{2}\cdot c} \tag{3}$$

$$\mathrm{d}s=\frac{\mathrm{d}s_{rr}}{1+\frac{D_r}{2}\cdot c} \tag{4}$$

$$\mathrm{d}\theta=\mathrm{d}s\cdot c \tag{5}$$

where: *fl* denotes front-left wheel, *fr* denotes front-right wheel, *rl* denotes rear-left wheel, and *rr* denotes rear-right wheel.

Assuming that there is no longitudinal slip of wheel, the $\mathrm{d}s_x$, where $x=fl, fr, rl, rr$, can be calculated using the increment of count by any wheel encoder and the average value is used as final $\mathrm{d}s$.

Assuming that there is no lateral slip of wheel, the curvature c depends on the steering angle, which can be calculated using Eq. (6):

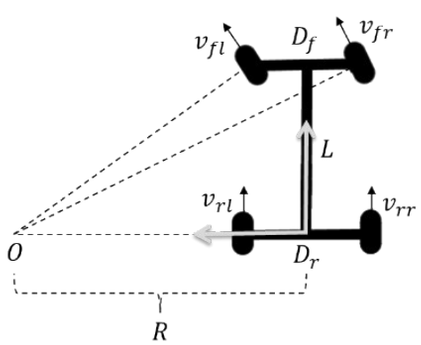
$$c=\frac{1}{L}\cdot\tan\frac{steering\ angle}{\eta} \tag{6}$$

where η denotes transmission ratio of steering system.

Finally, the 3-DOF pose of the vehicle can be simply estimated as:

$$\begin{bmatrix}x_{k+1}\\ y_{k+1}\\ \theta_{k+1}\end{bmatrix}=\begin{bmatrix}x_k\\ y_k\\ \theta_k\end{bmatrix}+\begin{bmatrix}\mathrm{d}s\cdot\cos\theta_k\\ \mathrm{d}s\cdot\sin\theta_k\\ \mathrm{d}\theta\end{bmatrix} \tag{7}$$

where: x and y denote the position of the center of rear axle, and θ denotes the yaw angle of the vehicle. When converting the 3-DOF pose to 6-DOF transformation ${}_oT_{bl}$, the remaining three degrees of freedom are simply assumed to be zero.



This is a simple model for car-like platforms, where R denotes turning radius, L denotes wheel base, D_f denotes front wheel track and D_r denotes rear wheel track.

Fig. 4 Ackermann steering geometry

2.4 Converter

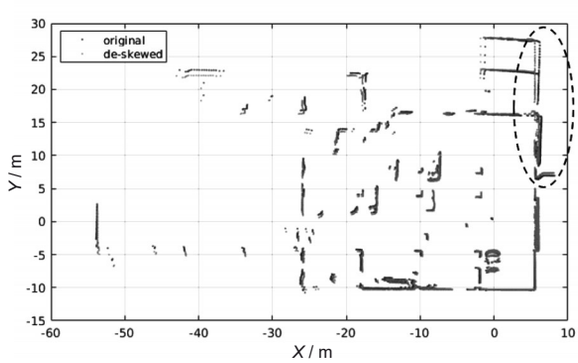
Most LiDAR sensors perform beam steering to scan the environment[13], which changes the direction of beam continuously to obtain a series of distance measurements. This mechanism is exploited to unify the format of point cloud produced by different LiDAR. The points are sorted by the time of measuring, which requires the point data to provide timestamp field for every point. Besides, many LiDAR also provide data field called ring indicating which scanner producing the point. Thus, timestamp and ring are required for the proposed algorithm to recover the scanning sequence of points for each laser scanner. A sequence is called a scan in this paper. For example, there are sixteen scans for the RS-LiDAR-16 while there are six scans for the Livox Avia produced by DJI Inc. Corp. This conversion allows the algorithm to adapt different type of LiDAR as long as timestamp and ring are available.

2.5 Point cloud de-skewing

The motion of LiDAR causes distortion of point cloud, which should be recovered firstly. Point cloud is de-skewed point by point using:

$${}^{t_0}_{ll}p_k={}^{t_0}_{o}T^{-1}_{ll}\cdot{}^{t_k}_{o}T_{ll}\cdot{}^{t_k}_{ll}p_k \tag{8}$$

where: ${}^{t_k}_{ll}p_k$ means the kth point in one frame of point cloud with timestamp t_k; ${}^{t_k}_{o}T_{ll}$ is interpolated from the transformations estimated by wheel odometry assuming a linear motion model between updating of wheel odometry. Compared with LOAM, which assumes constant angular and linear velocity during one period of scanning[4], the proposed method of de-skewing is more accurate because it exploits motion information directly measured by encoders. Fig. 5 compares the original point cloud and the de-skewed one.



This is sample data for an in-door parking lot. The sensor platform is turning left at the corner. For clarity, the points on ground and ceiling is removed. The top-right de-skewed points are slightly different from the original ones. The reason for this is that the points are back-propagated to about 0.1 s ago to remove the distortion caused by the motion of sensor.

Fig.5 De-skewing of point cloud

2.6 Feature extraction

Exactly, this module does the first stage work,

which extracts features in local scan scale. The points are classified by a geometric criterion called *corner angle* in Eq. (9), which is calculated using points within a narrow neighborhood illustrated in Fig. 6.

Firstly, a scan is divided to segments according to the angle or difference of distance between neighbor points. Segments with too few points are ignored for filtering out noisy points. And then the first and last points of remained segments are label as edge points, but the occluded points are removed for the reason described in reference [4]. Finally, points within segment are classified as edge point or planar point according to *corner angle* calculated using Eq. (9), where edge point is point with *corner angle* greater than the threshold, $angle_{corner}$, and planar point is point with *corner angle* smaller than another threshold, $angle_{planar}$. These two parameters can be chosen according to experiments. In this paper, they are simply set to 56°.

$$corner\ angle = \frac{1}{m}\left|\sum_{k=1}^{m}\left(\tan^{-1}\frac{r_{l_k}-r_m}{\theta_{l_k}\cdot r_m}+\tan^{-1}\frac{r_{r_k}-r_m}{\theta_{r_k}\cdot r_m}\right)\right| \tag{9}$$

Last but not least, the extraction is done using not-de-skewed point cloud, because the *corner angle* is calculated using neighbor points obtained within a very short duration of time, where the sensor can be treated as still, while the process of de-skewing may introduce extra noise to these local points resulting a worse quality of feature extraction.

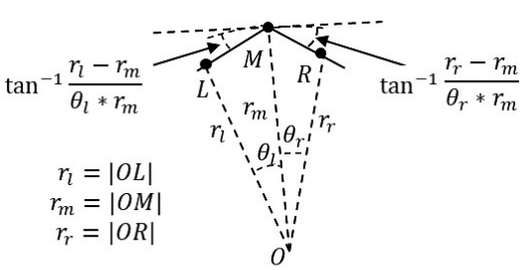


Fig.6 Extracting *corner angle* from laser measurements

2.7 LiDAR odometry

LiDAR odometry is performed in a scan-to-map manner which is widely used in previous work[4-5,7,14].

As shown by Fig. 7, the incoming laser frame is registered to the local map and then merged to it. To limit the scale of local map, it is arranged in local map segment, which is created by merging all registered feature points gathered along a fixed-length trajectory into two voxel maps, M_{edge} and M_{planar}, which is corresponding to two types of feature points p_{edge} and p_{planer} described in chapter 2.6. Similar to reference [5], the local map is then built using a sliding window approach with several latest local map segments.

With local map, the point cloud registration is to minimize the registration error by optimizing the pose of incoming frame ${}_{lm}T_{ll}$ using Eq. (10), which can be solved by Levenberg-Marquardt algorithm[15]:

$${}_{lm}\hat{T}_{ll} = \arg\min_{{}_{lm}T_{ll}}\left(\sum\left\|d_{edge}\right\|_{\Sigma_{edge}}+\sum\left\|d_{planar}\right\|_{\Sigma_{planar}}\right) \tag{10}$$

where: $\|\cdot\|_{\Sigma}$ means Mahalanobis distance parameterized by covariance matrix Σ, and

$$d_{edge} = ({}_{lm}T_{ll}\cdot p_{edge} - c)\times n \tag{11}$$

$$d_{planar} = ({}_{lm}T_{ll}\cdot p_{planer} - c)\times n \tag{12}$$

where: d_{edge} denotes point-to-line distance; d_{planar} denotes point-to-plane distance; c denotes the center of feature; n denotes the direction vector of feature, which indicates the direction of an edge line or the direction orthogonal to a plane.

For nonlinear optimization, the initial guess of ${}_{lm}T_{ll}$ is given by

$${}_{lm}T_{ll} = {}_{lm}T_{o}\cdot {}_{o}T_{ll} \tag{13}$$

where: ${}_{o}T_{ll}$ is given by wheel odometry; ${}_{lm}T_{o}$ is the most important state maintained by this module, which estimates the accumulated pose error of wheel odometry. It is updated using optimized LiDAR pose ${}_{lm}\hat{T}_{ll}$ as following:

$${}_{lm}T_{ll} = {}_{lm}\hat{T}_{ll}\cdot {}_{o}T_{ll}^{-1} \tag{14}$$

In summary, the optimization part of LiDAR odometry algorithm is just an iteration using Eq. (10) and Eq. (14) along with updating local map shown by Fig. 7.

An important detail is the second stage of feature extraction in local map scale. The resulting features are parameterized by a triplet $\{c, n, \Sigma\}$, which is calculated with several nearest neighbor points using PCA algorithm. PCA is also adopted by previous

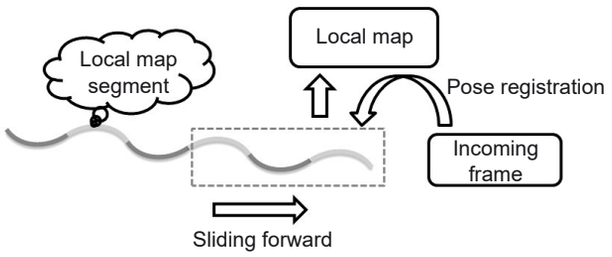


Fig.7 Sliding window updating strategy for local map

works for extracting vector n, among which reference [16] researches it in detail. But in fact, the sample covariance given by PCA is useful for weighting importance of every single feature in optimization. Assuming eigen values $\{\lambda_1, \lambda_2, \lambda_3\}$ and eigen vectors $\{n_1, n_2, n_3\}$ are extracted from m points using PCA, where $\lambda_1 < \lambda_2 < \lambda_3$, and c is the average position of these points, the feature is planar feature parameterized as:

$$\{c, n_1, \Sigma_{planar}\} \tag{15}$$

where
$$\Sigma_{planar} = \max\left(\frac{\lambda_1}{m-1}, \sigma_{prior}^2\right) \tag{16}$$

if
$$\sqrt{\frac{\lambda_1}{m-1}} < \sigma_{planar} \tag{17}$$

and
$$3\sqrt{\frac{\lambda_2}{m-1}} > \Delta_{planar} \tag{18}$$

where: σ_{prior} denotes the noise of LiDAR measurement; σ_{planar} denotes threshold for planar feature; Δ_{planar} denotes the single cell size of local voxel map M_{planar}.

Eq. (17) and Eq. (18) sets criteria for filtering out planar feature with requirements to the distribution of neighbor points. Eq. 16 estimates the uncertainty of feature directly from points data. And similarly, the feature is edge feature parameterized as:

$$\{c, n_3, \Sigma_{edge}\} \tag{19}$$

where
$$\Sigma_{edge} = \max\left(\frac{\lambda_1 + \lambda_2}{m-1}, \sigma_{prior}^2\right) \tag{20}$$

if
$$\sqrt{\frac{\lambda_1 + \lambda_2}{m-1}} < \sigma_{edge} \tag{21}$$

But in fact, edge feature is not as stable as planar feature due to noisy points in the real data, which may be points of vegetation. Thus, before classified as an edge feature, the feature must not be a degenerated feature. The feature is degenerated feature parameterized as:

$$\{c, n_1, \Sigma_{de}\}$$

where
$$\Sigma_{de} = \max\left(\sigma_{planar}^2 + \lambda_1, \sigma_{prior}^2\right) \tag{22}$$

If it meets Eq. (17), and

$$3\sqrt{\frac{\lambda_2}{m-1}} > \Delta_{edge} \tag{23}$$

where Δ_{edge} denotes the single cell size of local voxel map M_{edge}.

The form of degenerated feature is very similar to planar feature but with punishment of uncertainty in Eq. (22) for reducing its weight for optimization. It does not matter to discard this kind of feature in most time, but it helps when the environment is too noisy to extract other types of feature. If the degenerated feature is exploited, Eq. (10) is rewritten as:

$$\begin{aligned} {}_{lm}\hat{T}_{ll} = \arg\min_{{}_{lm}T_{ll}} \Big(& \sum \| d_{edge} \|_{\Sigma_{edge}} + \\ & \sum \| d_{planar} \|_{\Sigma_{planar}} + \sum \| d_{de} \|_{\Sigma_{de}} \Big) \end{aligned} \tag{24}$$

where d_{de} is point-to-plane distance analogous to Eq. (12).

2.8 Graph based mapping

Although LiDAR odometry is much more accurate than wheel odometry, it suffers from drift after travelling a long distance because of the nature of odometry, which accumulates error. Extra measurements are required for eliminating drift for creating global map with internal consistency. Loop closure is a solution for the trajectory with loop. And fusing GPS measurements is another practical way when the GPS signal is available. Both approaches are integrated to correct drift using a unified factor graph[17] illustrated by Fig. 8. It is solved incrementally via iSAM[8-9] algorithm with the open-sourced implementation GTSAM[18], which makes the mapping module able to run online efficiently.

The global map is arranged as a series of poses associated with submap generated from registered frames, which is a kind of topological map[19]. To reduce the scale of the graph, new pose is added every five meters assuming the drift of LiDAR odometry is small within a short distance. Four types of factor are introduced for optimizing the poses as shown by Fig. 7. Prior factor just set the coordinate origin to the start point of the trajectory. LiDAR

odometry factor and loop closure factor are implemented as described in reference [5]. The detail of them is omitted for the sake of brevity.

For fusing GPS measurement, an unknown transformation ${}_{gps}T_{map}$ is assumed, which transforms point from mapping coordinate to GPS coordinate. The GPS factor evaluates the difference between GPS measurement and its prediction using ${}_{gps}T_{map}$ and the poses of submap as:

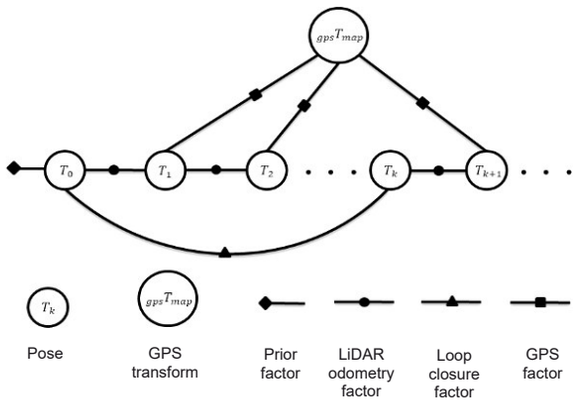
$$f\left(T_k, {}_{gps}T_{map}\right)= \left\| translation\left({}_{gps}T_{map}\cdot T_k\right)-t_{gps}\right\|_{\Sigma_{gps}} \tag{25}$$

where: T_k is the pose of submap; t_{gps} is GPS measurement expressed in cartesian coordinate like ENU system; Σ_{gps} is covariance matrix expressing the uncertainty of GPS measurement. The graph structure for GPS factor is similar to that in reference [20] but specialized for GPS measurement.

The GPS factor requires no extra process to transform GPS measurements to the local, but estimates the transformation automatically as well as correcting the drift in a unified optimization process, which simplifies the GPS fusion and is compatible to iSAM for it can fuse GPS measurement incrementally. This makes the system robust to the GPS signal quality, because it only requires a few GPS measurements to correct pose instead of continuous GPS measurements. Once GPS fails, the measurement is just dropped out till the signal recovers. The system may drift during the GPS-denied time, but the error can be corrected when GPS is available or a loop closure is detected. Only one limit is that it requires at least three GPS factors to form a closed constraint (see Fig. 8).

3 Experiments

The proposed algorithm is tested on the laptop with i5-8265U CPU under Ubuntu 18.04. Tab. 1 shows the working frequency of each module. Because wheel odometry is triggered by the wheel encoder, its frequency is proportional to the vehicle speed and limited by the updating rate of encoder. The feature extraction, point de-skewing and LiDAR



Factors are visualized as lines with different markers. Estimated variables are visualized as circle with variable names.

Fig.8 Factor graph for fusing measurements

Odometry process 10 Hz point cloud data in one pipeline and drop out data when system is busy with no harm to performance. The graph based mapping record map data in a low frequency in most time and optimizes the global map only when loop closure is detected or GPS is available. The workload is well balanced to meet a real-time performance of odometry and online processing of global mapping. When the GPS signal is available, the local map is well located in the GPS coordinate system.

Tab.1 System update frequency

Modules	Frequency/Hz
Wheel odometry	10–50
Feature extraction and point de-skewing	10
LiDAR odometry	8–10
Graph based mapping	1

For further evaluation of the performance of accuracy, the proposed method was compared to several previous works using their open-sourced implementation, which are listed in Tab. 2. And three datasets were collected from the autonomous vehicle platform for benchmark, which are listed in Tab. 3. For the sensor configuration of LIO-SAM, the IMU and GPS is both provided by RT 3002, which is an inertial and navigation system produced by OxTS company. As GPS is used as an optional correction both for LIO-SAM and proposed method, it is filtered by signal quality and is assumed a noise with 1 m standard deviation for both methods.

The accuracy of trajectory was evaluated by comparing it with the ground truth trajectory obtained

Tab.2 Sensor configuration and loop closure of compared SLAM Algorithms

Algorithm	Sensors configuration	Loop closure
A−LOAM[4]	LiDAR	No
LeGO−LOAM[7]	LiDAR	Yes
LIO−SAM[5]	IMU+LiDAR+optional GPS	Yes
Proposed WLOAM	Wheel+LiDAR+optional GPS	Yes

Tab.3 Details of datasets

Dataset	Trajectory length/m	Environment	Average speed/(km/h)
Single Loop	2 883	Building and vegetation	14.8
West Campus	2 276	Almost building	13.3
South campus	8 088	Building and vegetation	15.8

from RT 3002, which was expected to be of high accuracy with 2 cm standard deviation in the RTK mode. But due to the signal quality, the accuracy would degenerate occasionally and the device would indicate no RTK mode available. Thus, only the data in RTK mode was exploited. Noting that the GPS measurement is of worse accuracy in altitude, the accuracy is also evaluated by projecting trajectory to ground plane and calculating horizontal position error.

Absolute pose error (APE) and relative pose error (RPE) are widely used for evaluating precision of SLAM algorithm[21]. For the APE, the trajectory was firstly aligned to ground truth trajectory using Umeyama's algorithm, and then calculating APE using Eq. (26) and APE 2D using Eq. (27):

$$\text{average}\left(\text{APE}_{1:m}\right)=\frac{1}{m}\sum_{i=1}^{m}\left\|\text{trans}\left(Q_i^{-1}SP_i\right)\right\| \tag{26}$$

$$\text{average}\left(\text{APE 2D}_{1:m}\right)=\frac{1}{m}\sum_{i=1}^{m}\left\|\text{trans}\left(\left\langle Q_i\right\rangle^{-1}\left\langle SP_i\right\rangle\right)\right\| \tag{27}$$

where: Q_i denotes the ground truth pose; P_i denotes the estimated pose; S denotes transformation for aligning trajectory; $trans(\cdot)$ means extracting translation part of a pose; $\langle\cdot\rangle$ means extracting 2D pose from 3D pose by projection.

For calculating the RPE, a pair of poses Δ meters apart along the trajectory are selected from ground truth trajectory and estimated trajectory respectively. And it is calculated using Eq. (28):

$$\text{average}\left(\text{RPE}_{1:m}^{\Delta}\right)=\frac{1}{m}\sum_{i=1}^{m}\left\|\text{trans}\left(\left(Q_i^{-1}Q_{i+\Delta}\right)^{-1}P_i^{-1}P_{i+\Delta}\right)\right\| \tag{28}$$

The drift rate with respect to trajectory length Δ is defined as:

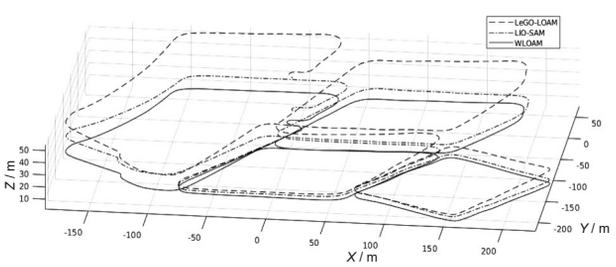
$$drift_{\Delta}=\frac{average\left(\text{RPE}_{1:m}^{\Delta}\right)}{\Delta}\times 100\% \tag{29}$$

Just as described in reference [22], a series of trajectory length Δ_i are selected and further an overall RPE is obtained using Eq. (30) and an overall drift rate is obtained using Eq. (31):

$$\text{RPE}_{overall}=\frac{1}{\sum_i m_i}\sum_i m_i\cdot average\left(\text{RPE}_{1:m_i}^{\Delta_i}\right) \tag{30}$$

$$drift_{overall}=\frac{1}{\sum_i m_i}\sum_i m_i\cdot drift_{\Delta_i} \tag{31}$$

Fig. 9 shows the odometry trajectory without the correction of loop closure and GPS measurement. LeGO-LOAM and LIO-SAM suffer from more z-directional drift than the proposed method. Tab. 4 shows the overall drift rate in the pure odometry mode for each method. Note that A-LOAM is dropped out for this comparison for it is possible for it to performance loop closure in an implicit way and it is not possible to turn it off.



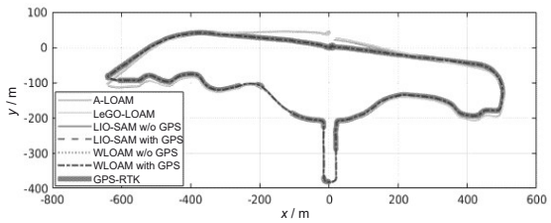
In pure odometry mode, the drift in the z direction is significant for LeGO-LOAM and LIO-SAM.

Fig.9 Comparison of odometry

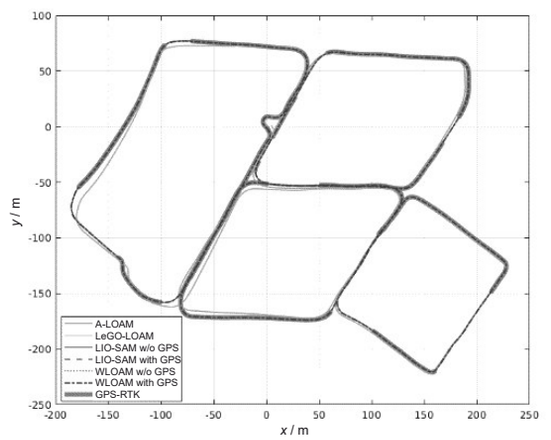
Tab.4 Drift rate of odometry

mode	drift/%
LeGO-LOAM	2.79
LIO-SAM	0.99
WLAOM	0.76

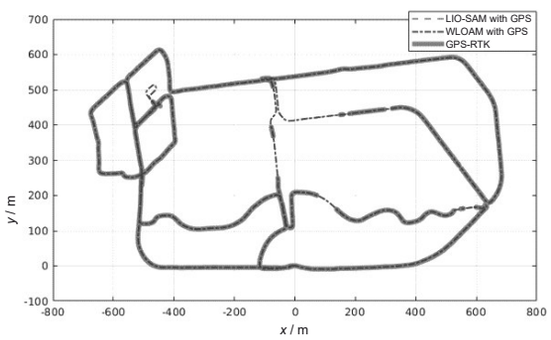
Fig. 10 shows the estimated trajectories in comparison with the ground truth. Most trajectories match the ground truth well except for A-LOAM and LeGO-LOAM. A-LOAM deviates significantly from the ground truth trajectory in Fig. 10a and Fig. 10b for its implicit method failed to perform loop closure. LeGO-LOAM fails to detect loop closure in Fig. 10a. This indicates that the accuracy can be



a Single loop



b West campus



c South campus

The GPS-RTK is not always available due to the signal quality, thus the trajectory is the disconnected line segments in Fig. 10. Most trajectories match the GPS-RTK trajectory well excepting the A-LOAM and LeGO-LOAM.

Fig. 10　Comparison of trajectories

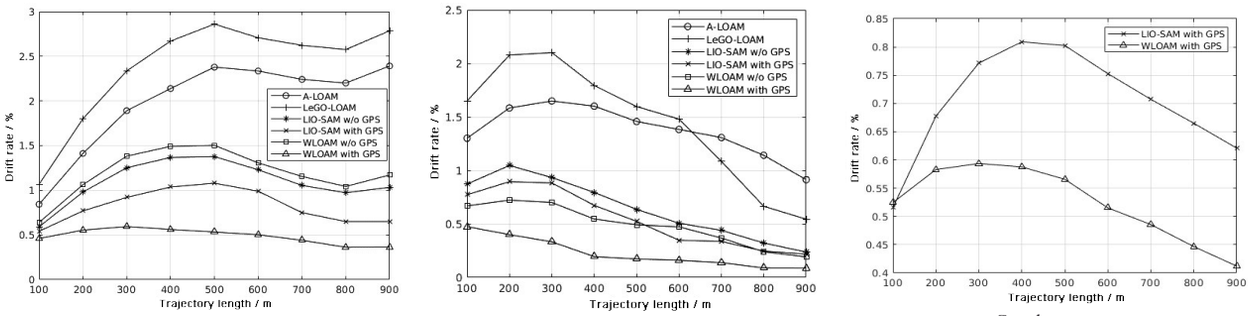
greatly improved if loop closure is performed successfully. Otherwise the accuracy is undermined by the drift. For dataset South Campus shown in Fig. 10c, only methods with GPS succeed and is plotted on the figure while other methods accumulate too much error and fail to find loop closure. This indicates that the combination of odometry and loop closure works well only when the loop is not too long, because the loop closure method is position-based in this paper, where too large position error leads to failure. But fusing GPS deals with this problem by limiting the drift.

Fig. 11 shows the drift rate in different trajectory lengths, where the proposed method with GPS outperforms others and has a similar performance with LIO-SAM if GPS is removed. Tab. 5 shows the absolute accuracy by APE or APE 2D and relative accuracy by overall drift rate, where the best and second best is shown in bold character. Note that the APE 2D is much smaller than APE when the GPS is removed but the difference gets small when GPS is fused. This is because the accuracy of altitude measurement by GPS is relatively low due to the nature of GPS measurements, which makes up the majority of difference between estimated trajectory and ground truth. However, when GPS is fused, the estimated trajectory is forcedly aligned to the GPS measurements eliminating the difference.

Fig. 12 shows the mapping result of South Campus dataset using proposed method. GPS correction is fused so that the point cloud map is automatically aligned to the satellite map using the estimated transformation from the local coordinate to the GPS coordinate.

4　Conclusions

This paper proposeds a SLAM system WLAOM exploiting different sensors for real-time odometry and online mapping. An improved feature-based LiDAR odometry method is enhanced by a kinetic-model-based wheel odometry method to produce low-drift pose estimation with low delay. A graph-based method is introduced for fusing loop closure and GPS measurement, where an Auto-Aaligned-GPS factor is modeled to correct pose and estimate alignment parameters. This makes it possible for the system to map a broad area covering several kilometers even when the GPS signal is not always available. But GPS measurement contains more uncertainty in altitude, which is ignored now and may cause bad estimation of roll and pitch when the signal quality is poor. Thus, a future work is to fuse the information about the roll and pitch angle. It may come from inertial measurement which contains information about the gravity. In addition, because



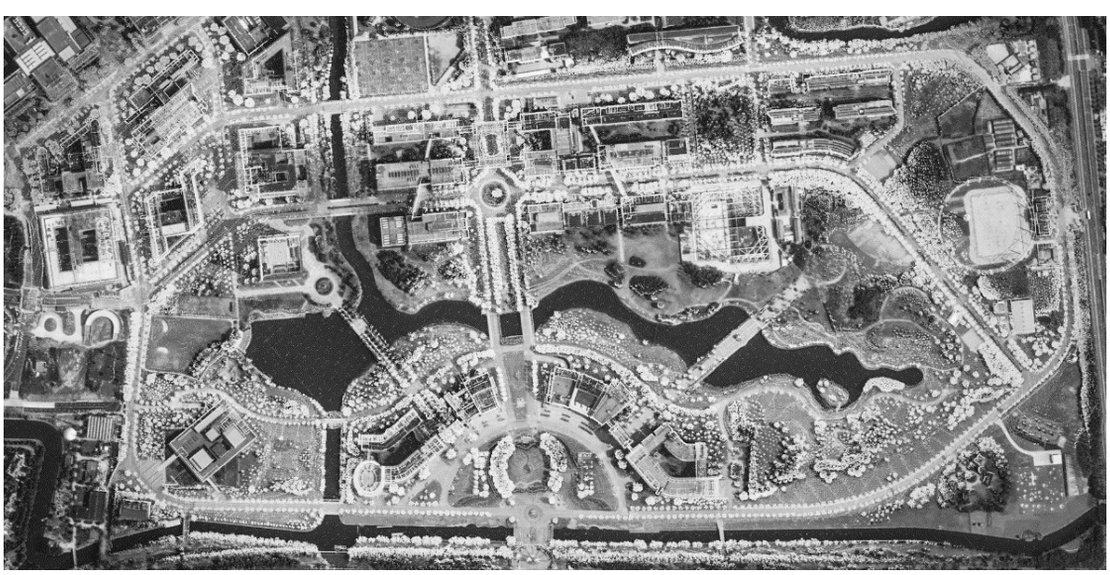
a Single loop b West campus c South campus

The proposed method and LIO-SAM achieve much better performance than the other methods on drift rate. For all datasets, the proposed method with GPS performs best. In a and b, without GPS, LIO-SAM and the proposed method achieve a closed performance. In c, all methods without GPS failed. Only the proposed method and LIO-SAM with GPS survive and the proposed method has a lower drift rate.

Fig. 11 Comparison of drift rate.

Tab.5 Accuracy of trajectory

Algorithm	Single loop			West campus			South campus		
	APE/m	APE 2D/m	drift/%	APE/m	APE 2D/m	drift/%	APE/m	APE 2D/m	drift/%
A—LOAM	16.00	14.60	1.92	6.19	4.03	1.38	—	—	—
LeGO—LOAM	16.30	14.45	2.30	1.80	0.33	1.49	—	—	—
LIO—SAM w/o GPS	4.55	1.65	1.08	0.78	0.23	0.68	—	—	—
LIO—SAM with GPS	**1.11**	1.05	**0.82**	**0.46**	**0.11**	0.57	0.85	0.78	0.70
WLOAM w/o GPS	4.94	**0.85**	1.18	0.64	0.19	**0.51**	—	—	—
WLOAM with GPS	**0.32**	**0.30**	**0.49**	**0.14**	**0.09**	**0.24**	**0.27**	**0.26**	**0.53**



The point cloud is visualized as brighter points.

Fig. 12 Global point cloud map of the South campus aligned with satellite map

the system is well de-coupled as separated modules, it is possible to transplant the system to other platforms like drone by only replacing the Wheel Oodometry module with the inertial system. And Moreover, as described in chapter 2.4, a general LiDAR data format requirement makes the system compatible with different type of LiDAR, but the evaluation work is left for the future.

References:

[1] CADENA C, CARLONE L, CARRILLO H, *et al*. Past, present, and future of simultaneous localization and mapping: toward the robust-perception age [J]. IEEE Transactions on Robotics, 2016, 32(6):1309. DOI: 10.1109/TRO.2016.2624754.

[2] BRESSON G, ALSAYED Z, YU L, *et al*. Simultaneous localization and mapping: a survey of current trends in autonomous driving [J]. IEEE Transactions on Intelligent Vehicles, 2017, 2(3):194. DOI: 10.1109/TIV.2017.2749181.

[3] MOHAMED S A S, HAGHBAYAN M H, WESTERLUND T, *et al*. A survey on odometry for autonomous navigation systems [J]. IEEE Access, 2019, 7: 97466. DOI: 10.1109/ACCESS.2019.2929133.

[4] ZHANG J, SINGH S. Low-drift and real-time lidar odometry and mapping [J]. Autonomous Robots, 2017, 41(2): 401. DOI: 10.1007/s10514-016-9548-2.

[5] SHAN T, ENGLOT B, MEYERS D, *et al*. LIO-SAM: tightly-coupled lidar inertial odometry via smoothing and mapping [C]// 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). Las Vegas: IEEE, 2020: 5135.

[6] BRUNKER A, WOHLGEMUTH T, FREY M, *et al*. Odometry 2.0: a slip-adaptive EIF-based four-wheel-odometry model for parking[J]. IEEE Transactions on Intelligent Vehicles, 2019, 4(1): 114. DOI: 10.1109/TIV.2018.2886675.

[7] SHAN T, ENGLOT B. LeGO-LOAM: Lightweight and ground-optimized lidar odometry and mapping on variable terrain [C]// 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). Madrid: IEEE, 2018: 4758.

[8] KAESS M, RANGANATHAN A, DELLAERT F. iSAM: Incremental smoothing and mapping[J]. IEEE Transactions on Robotics, 2008, 24(6): 1365. DOI: 10.1109/TRO.2008.2006706.

[9] KAESS M, JOHANNSSON H, ROBERTS R, *et al*. iSAM2: Incremental smoothing and mapping using the bayes tree[J]. The International Journal of Robotics Research, 2012, 31(2): 216. DOI: 10.1177/0278364911430419.

[10] UMEYAMA S. Least-squares estimation of transformation parameters between two point patterns[J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 1991, 13(4): 376. DOI: 10.1109/34.88573.

[11] QUIGLEY M, GERKEY B, CONLEY K, *et al*. ROS: An open-source robot operating system [C/OL]. (2009-01-01) [2021-08-25]. https://www. researchgate. net/publication/303138182_ROS_An_open-source_Robot_Operating_System.

[12] 郭媛媛. 全自动泊车系统的位姿估计技术研究[D]. 上海:同济大学, 2016.
GUO Yuanyuan. Research on pose estimation technology of fully automatic parking system [D]. Shanghai: Tongji University, 2016.

[13] RAJ T, HASHIM F H, HUDDIN A B, *et al*. A survey on LiDAR scanning mechanisms [J]. Electronics, 2020, 9(5): 741. DOI: 10.3390/electronics9050741.

[14] XU W, ZHANG F. FAST-LIO: A fast, robust lidar-inertial odometry package by tightly-coupled iterated Kalman filter[J]. IEEE Robotics and Automation Letters, 2021, 6(2): 3317. DOI: 10.1109/LRA.2021.3064227.

[15] HARTLEY R, ZISSERMAN A. Multiple view geometry in computer vision [M]. 2nd ed. Cambridge: Cambridge University Press, 2003.

[16] ZHANG S, XIAO L, NIE Y, *et al*. Lidar odometry and mapping based on two-stage feature extraction [C]// 39th Chinese Control Conference (CCC 2020). Shenyang: Chinese Association of Automation, 2020: 3966.

[17] DELLAERT F, KAESS M. Factor graphs for robot perception [J]. Foundations and Trends in Robotics, 2017, 6(1/2): 1. DOI: 10.1561/2300000043.

[18] DELLAERT F. Factor graphs and GTSAM: A hands-on introduction[J]. Georgia Institute of Technology, 2012. [2021-06-25]. https://www. semanticscholar. org/paper/Factor-Graphs-and-GTSAM% 3A-A-Hands-on-Introduction-Dellaert/b94fbf48299d78cd586c057e700763ec09b88f80.

[19] YANG A, LUO Y, CHEN L, *et al*. Survey of 3D map in SLAM: Localization and navigation[C]// FEI M, MA S, LI X, et al. Advanced Computational Methods in Life System Modeling and Simulation. Singapore: Springer Singapore, 2017: 410. DOI: 10.1007/978-981-10-6370-1_41.

[20] DING W, HOU S, GAO H, *et al*. LiDAR inertial odometry aided robust LiDAR localization system in changing city scenes [C]// 2020 IEEE International Conference on Robotics and Automation (ICRA). Paris: IEEE, 2020: 4322.

[21] STURM J, ENGELHARD N, ENDRES F, *et al*. A benchmark for the evaluation of RGB-D SLAM systems[C]// 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems. Vilamoura-Algarve, Portugal: IEEE, 2012.

[22] GEIGER A, LENZ P, URTASUN R. Are we ready for autonomous driving? The KITTI vision benchmark suite [C]// 2012 IEEE Conference on Computer Vision and Pattern Recognition. Providence: IEEE, 2012.