



A tightly coupled monocular visual lidar odometry with loop closure

Lingbo Meng¹ · Chao Ye¹ · Weiyang Lin¹

Received: 12 May 2021 / Accepted: 30 November 2021 / Published online: 5 January 2022
© The Author(s), under exclusive licence to Springer-Verlag GmbH Germany, part of Springer Nature 2022

Abstract

Simultaneous localization and mapping (SLAM) is a fundamental requirement for mobile robots like self-driving cars. Vision-based methods have advantages in sensor cost and loop closure detection, but are sensitive to illumination change and texture deficiency. Lidar-based SLAM systems perform better in accuracy, field-of-view and robustness to environmental changes, but may easily fail in structure-less scenarios. To compensate for the deficiencies of standalone sensors and provide more efficient SLAM functions, in this paper we propose a tightly coupled monocular visual lidar odometry, which tightly fuses the measurements of a monocular camera and a 3D lidar in a joint optimization. The system starts with a data preprocessing module, which outputs 3D visual and laser features through feature extraction and data association. Furthermore, the tightly coupled visual lidar odometry tightly fuses the visual and laser features in a unified optimization framework to estimate the transformation between consecutive scans. Finally, we combine visual and vicinity loop detection to construct loop constraints and optimize a 6-DOF global pose graph to achieve global-consistent pose estimation and environment mapping. The performance of our system is verified in the public KITTI dataset, and the experimental results demonstrate that the proposed method can run in real time with the 64-line lidar data and achieve better in accuracy, runtime and mapping against other state-of-the-art lidar-based and fusion-based methods.

Keywords Pose estimation · Simultaneous localization and mapping (SLAM) · Loop closure · Tightly coupled · Visual lidar odometry

1 Introduction

With the popularization of intelligent mobile robots like self-driving cars and unmanned aerial vehicles, the need of providing a high-accuracy and lightweight SLAM system for these platforms has been rising. Though many efforts have been devoted to improve the performance of SLAM over the past few decades, there are still some challenges in integrating multi-modality sensors for better localization and mapping performance.

Thanks to its small size, low cost and easy-to-use hardware, monocular cameras have long been popular with SLAM researchers and robot manufacturers. However, monocular cameras cannot recover the metric scale of envi-

ronment, and thus the pose estimation and mapping results of pure monocular SLAM systems have no definite scale information, which limits the further application of this kind of sensor in real-world applications. To remedy that, stereocameras estimate the depth of each pixel in the image by calculating the parallax between two cameras, and then recover the environmental information with accurate scale. However, stereocameras also have shortcomings such as complex calibration, limited depth range depending on the baseline length, and large amount of parallax computation.

By transmitting and receiving laser beams, lidars can obtain accurate structure information of the environment, and thus perform well in computational complexity, sensing distance, measurement accuracy and insensitivity to light change. Nevertheless, a point cloud collected by a lidar is usually very sparse in the vertical direction and does not contain any color information, which makes it difficult to provide enough information for place recognition and loop detection. Therefore, most of the current lidar-based SLAM frameworks [7,23,30] only have the odometry and mapping part, and can hardly perform loop closure or achieve global

✉ Weiyang Lin
wylin@hit.edu.cn

Lingbo Meng
lingbo_meng@outlook.com

Chao Ye
yechao@hit.edu.cn

¹ Harbin Institute of Technology, Harbin, China

consistent mapping. Besides, lidar-based methods will also suffer from the constraint degradation in structure-less scenarios such as tunnel and open plane areas, which finally lead to failure of the whole system.

In the research of odometry methods based on fusion of vision and laser, traditional loose-coupling-based methods, such as [29], usually process visual and laser data separately to obtain one low-precision but high-frequency visual odometry and the other higher-precision but low-frequency laser odometry, and then directly append the pose estimation of visual odometry to the result of laser odometry. This kind of fusion method is straightforward and shows good performance in accuracy. However, since visual constraints and laser constraints are constructed and optimized separately in two odometries, the complementarity of visual and laser constraints has not been fully utilized. In contrast, recent literature [21] of visual inertial odometry (VIO) reveals that the tight coupling scheme which combines visual and IMU measurement in a common optimization framework shows great advantages in positioning accuracy compared with the loose coupling scheme [17,19], where IMU integration is treated merely as high-frequency state propagation for the visual odometry part. Therefore, the tight coupling scheme has a good prospect in the research of multi-sensor fusion odometry.

To combine the best of vision and laser data, we present a tightly coupled, optimization-based monocular visual lidar odometry method, as shown in Fig. 1. Compared with other fusion algorithms [24,26,29,31], the main difference is that our system tightly fuses the measurements of a monocular camera and 3D lidar in a joint-optimization. The main contributions of our work are as follows:

- A tightly coupled monocular visual lidar odometry algorithm. By jointly optimizing the visual and lidar measurements in a common optimization framework, we can achieve state-of-the-art 6-DOF pose estimation in real time.
- A Reliable data preprocessing method is proposed to associate visual and lidar data, and provide depth estimation for 2D visual features.
- Visual loop detection and vicinity loop detection are combined in a separate thread, and loop constraints are constructed through ICP registration.
- Loop and odometry constraints are added to the 6-DOF global pose graph to achieve globally consistent pose estimation and mapping.

The rest of this paper is organized as follows. In Sect. 2, we discuss the related literature in this field. In Sect. 3, we describe the preliminaries we made for this paper, and give an overview of the whole system pipeline. The data preprocessing method is introduced in Sect. 4. The proposed tightly

coupled visual lidar odometry method and loop closure module are presented in Sects. 5 and 6, respectively. Experimental evaluations of the proposed method are presented in Sect. 7 and conclusion is finally made in Sect. 8.

2 Related work

Both visual and laser SLAM methods have their own advantages and disadvantages. Vision-based methods, such as ORB-SLAM2 [20], LSD-SLAM [8], DS-PTAM [6] and LDSO [10] perform well in sensor cost and loop closure detection, but they are sensitive to illumination change and have a limited field-of-view (FOV).

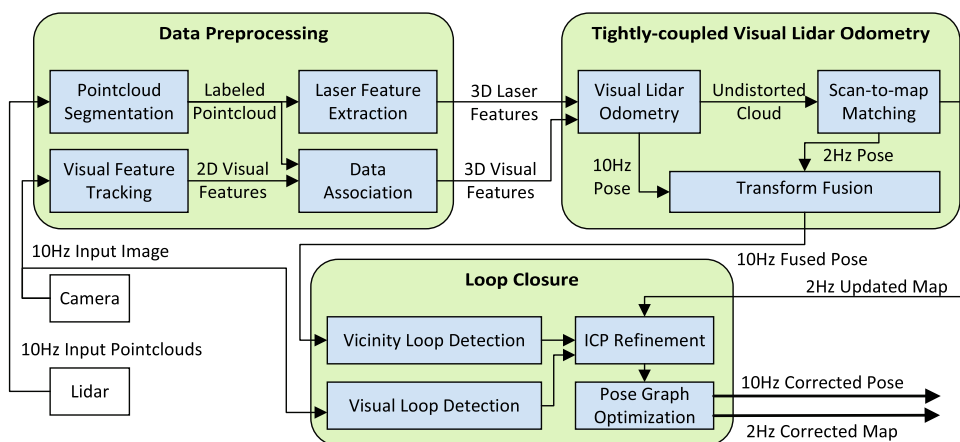
In order to improve the robustness and accuracy of the monocular visual SLAM system, it is a good approach to introduce a low-cost IMU [14] to construct a visual inertial odometry system, such as what [15,21,27] does. This is because the integral measurements of IMU can provide an absolute scale information for a monocular system, and can improve the performance of motion tracking through tight or loose coupling with visual constraints. Nevertheless, the overall localization accuracy of the vision-based SLAM methods is still lower than that of the lidar-based methods.

In addition, since most visual SLAM methods assume that the environment is static, visual features on moving objects will lead to a decrease in accuracy and robustness of the entire visual localization system. To solve this problem, learning-based methods [4,18], such as Bayesian framework or convolutional neural networks (CNNs), can be utilized to realize the detection and removal of dynamic regions in the image, which shows a significant improvement compared with the original SLAM method. Nevertheless, such methods inevitably lead to a significant increase in computing time, which are often prohibitive for mobile robot platforms.

Lidar-based SLAM systems have gained better popularity for their high precision in range detection, a wide horizontal FOV of 360 degrees and robustness to environmental changes. These algorithms, can provide both state-of-the-art pose estimation and elaborate 3D point cloud map of the traversed environment. Nevertheless, pure lidar-based methods only extract structural features of the environment and may easily fail in structure-less scenarios like tunnels and hallways.

One of the most popular 3D lidar odometry method is LOAM [30]. The LOAM divides the intricate SLAM problem into two concurrent algorithms, the lidar odometry which performs feature extraction and pose estimation at a high frequency and the lidar mapping module that further uses more extracted features for matching and mapping at a slower frequency for higher precision. In this way, it can achieve both low drift of motion estimation and acceptable runtime.

Fig. 1 Block diagram of the proposed tightly coupled monocular visual lidar odometry with loop closure



Based on LOAM [30], Shan et al. proposed a lightweight method, LeGO-LOAM [23], which is specially optimized for the ground scene and lightweight enough for running on an embedded-system like Jetson TX2. This algorithm adopts point cloud segmentation to filter out the noise points in the lidar point cloud, and divides the solution of 6-DOF pose transformation into two sequential 3-DOF optimization, which makes this algorithm can run on low-performance computers while maintaining the accuracy similar to LOAM.

Besides, different from the common scan-to-scan matching method for pointcloud registration, Deschaud et al. proposed a scan-to-model matching framework, IMLS-SLAM [7]. This method uses the implicit moving least squares (IMLS) surface model [5] to express several previous frame of point clouds, and registers the current scan of point cloud to the surface model rather than another scan for pose estimation.

Although lidar-based SLAM methods generally perform better than the vision-based method in positioning accuracy, field-of-view and reliability, they can only make use of geometric information of the point cloud to select structural features, which may lead to degeneracy of constraints in symmetrical scenarios, such as a symmetrical tunnel or an open highway without enough vertical landmarks. In contrast, vision-based methods mainly utilize the visual feature points, which is naturally complementary to the laser features.

In order to exploit the benefits of different sensor modalities, SLAM methods based on fusion of multiform sensors have become popular.

For data fusion of vision and lidar, Zhang et al. extended LOAM [30] to VLOAM [29], which loosely couples a monocular camera and a lidar, which starts with a visual odometry for providing motion prior and utilizes a lidar matching to further refine the motion estimation. In addition, Zhang et al. further presented a multilayer laser-visual-inertial odometry and mapping method [31], which is able to

handle sensor degradation by automatically bypassing failure modules.

Besides, through incorporating a tightly coupled stereo visual inertial odometry and a lidar mapping module, Shao et al. proposed a stereo visual inertial lidar SLAM method [24] that demonstrated improved accuracy compared to pure lidar-based methods. Likewise, Wang et al. proposed a method [26] to combine a tightly coupled monocular visual inertial odometry and a lidar scan-matching module.

However, for all of these methods, the laser and vision data are loosely coupled with each other and sensor fusion still shows potential for further improvement with tight coupling, which can make better use of the complementarity between different sensors.

3 Preliminaries and system overview

3.1 Preliminaries

In this paper, we assume that the intrinsic parameters of the monocular camera and the extrinsic parameters between it and the lidar are calibrated. Time synchronization is carried out between the camera and the lidar, where the timestamp of an image corresponds exactly to the middle moment of a point cloud, as shown in Fig. 2. Compared to the relatively long receiving time of a point cloud, an image can be considered to be captured instantaneously at a particular moment, so we use the image timestamp as the unified timestamp of each pair of laser and image data. In addition, we also assume that the linear and angular velocities of the sensor platform are relatively smooth in a short time period.

In our system, we define three coordinate systems—the lidar coordinate system $\{L\}$, the camera coordinate system $\{C\}$ and the world coordinate system $\{W\}$. The first lidar coordinate system after initialization is set as the world coordinate system. Since the sensor platform moves constantly, we denote the lidar coordinate system and the camera coor-

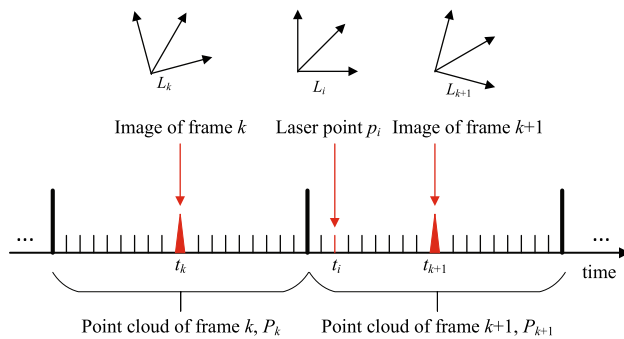


Fig. 2 Schematic diagram of time synchronization and coordinate transformation between the lidar and the camera

ordinate system at time t_i as L_i and C_i , respectively. The transformation from coordinate system L_{k+1} to L_k is represented as $\mathbf{T}_{L_{k+1}}^{L_k}$ and a feature point in L_k is denoted as $\mathbf{X}_i^{L_k}$.

3.2 System overview

The structure of the proposed tightly coupled monocular visual lidar odometry is shown in Fig. 1. The system receives time-synchronized input from a 3D lidar and a monocular camera, and output 6-DOF pose estimation together with a 3D point cloud map of its traversed environment.

The whole system is divided into three modules. Firstly, in the data preprocessing module, 2D visual features are extracted and tracked from the input image, and the raw point cloud is segmented to acquire a labeled point cloud where unreliable and unnecessary points are excluded. The 2D visual features and the labeled point cloud are further associated in the data association submodule where the depth is restored to acquire 3D visual features. The 3D laser features are extracted at the laser feature extraction submodule in the meantime. Then, the tightly coupled visual lidar odometry module tightly fuses 3D laser features and 3D visual features to estimate transformation between consecutive scans at high frequency and refine it further through a scan-to-map matching. These two pose estimations at different frequencies are further fused to acquire the high-frequency and high-precision pose estimation. Finally, the loop closure module takes in the odometry pose estimations and the input images to detect vicinity or visual loop, uses ICP method to construct loop constraints and performs global pose graph optimization to eliminate the drift.

4 Data preprocessing

This section presents preprocessing steps for both visual and lidar measurements. For visual measurements, we track features between consecutive frames and extract new features

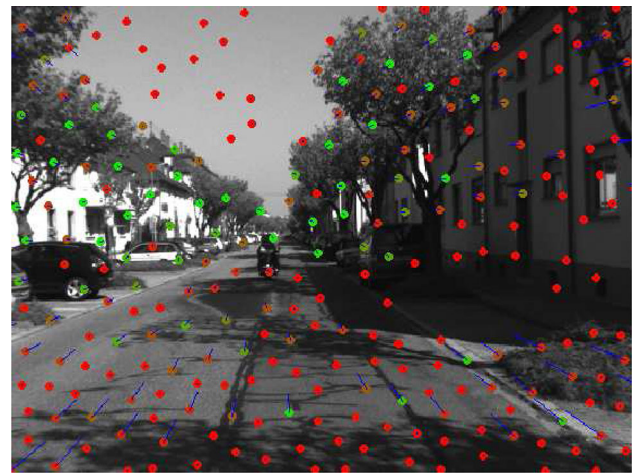


Fig. 3 The extraction and tracking of visual features. According to the number of continuous tracking between two adjacent frames from less to more, the color of each tracked visual feature point is represented from green to red. The blue line connected to each feature point uses its direction and magnitude to represent the velocity vector of that feature point

in the latest frame. For lidar measurements, we segment the raw input to acquire a labeled point cloud and extract 3D laser features. Then, the labeled point cloud and the tracked 2D visual features are associated with each other to obtain 3D visual features with depth.

4.1 Visual feature tracking

For each new frame of image, we take advantage of the LK sparse optical flow algorithm [16] to track existing visual features from the previous frame to the current frame. At the same time, new corner features are extracted using the Shi-Tomasi method [25] from the current frame to maintain a constant feature number (300). A minimal distance between two neighboring visual features is set to keep a uniform feature distribution. Besides, we utilize the RANSAC method and the fundamental matrix to reject potential outliers. We also use the CLAHE method [32] to improve the contrast of images, which can alleviate the challenge of light variation and make feature extraction and tracking easier.

The extraction and tracking of visual features are shown as Fig. 3. According to the number of continuous tracking between two adjacent frames from less to more, the color of each tracked visual feature point is represented from green to red. The blue line connected to each feature point uses its direction and magnitude to represent the velocity vector of that feature point.

4.2 Point cloud segmentation

When the sensor platform moves in a noisy environment with small objects like shrubs and tree leaves, feature extracting methods based on local surface smoothness [30] can generate a large quantity of unreliable laser features. This is because a point cloud of multi-line lidar is relatively sparse in the vertical direction, and it is difficult for the features of small objects to be continuously observed in two consecutive scans.

In order to perform fast and reliable laser feature extraction and also provide label information for the data association part, we segment the raw point cloud into labeled object clusters. Let $P_k = \{p_1, \dots, p_n\}$ be a raw point cloud received at time t_k , where p_i is a point from P_k . Firstly, we project P_k onto a range image, where each valid point p_i in P_k has a corresponding 2D pixel. We use d_i to represent the Euclidean distance from p_i to the sensor origin. Then, a fast ground plane segmentation method for ground vehicles [12] is conducted to extract ground points and label them in the range image. Thirdly, we apply an image-based method [2] to the range image to segment non-ground points into separated object clusters. Each cluster will be assigned a unique label in the range image and the ground points also form one cluster. Finally, since small objects with fewer points are less reliable for feature extraction, here we only keep clusters that have more than 50 points or extend at least three laser lines.

After this, only ground points and reliable object points that may represent large objects like walls and tree trunks are preserved as a labeled point cloud for further processing. A visualization of a point cloud after segmentation is shown in Fig. 4. The white points belong to the ground point set, and the different object sets are distributed from green to red according to the size of each set. The light blue point sets are outliers that are eliminated. It can be seen that the plant point set in the yellow box is removed as outliers after segmentation.

4.3 Laser feature extraction

In order to reduce the amount of data and avoid unreliable clusters, here we only extract laser features from the labeled point cloud. The feature extraction method is similar to [30], but we use distance values instead of original 3D coordinates to reduce calculation. Let p_i be a point in a row of the range image, we define a term to evaluate the curvature of this point,

$$c = \frac{1}{d_i \cdot |S|} \left| \sum_{j \in S, j \neq i} (d_j - d_i) \right| \tag{1}$$

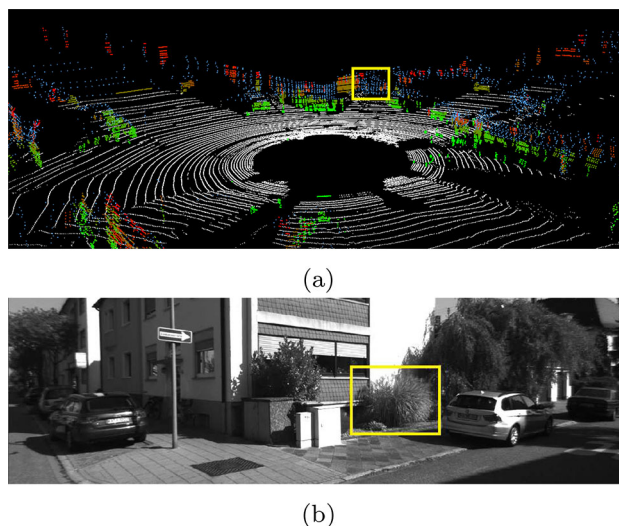


Fig. 4 A labeled point cloud after segmentation. In **a**, the ground points are labeled as white. The reliable object clusters are distributed from green to red according to the size of each cluster. The blue points are outliers. **b** is the visual image from the same viewpoint, and we can see the plant in yellow box is labeled as outliers

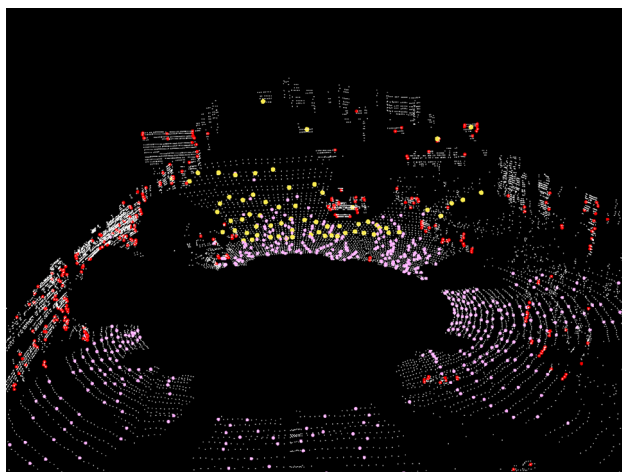
where S is a neighbor set of consecutive points around p_i at the same row with $|S|/2$ points on either side, and d_i represents the distance from point p_i to the lidar origin

To ensure an even distribution of features in all directions, we separate the range image horizontally into several equal subregions. Then, in each row of a subregion, we sort the points according to their curvature values and set a threshold c_{th} to distinguish edge features with c larger than c_{th} and planar features with c smaller than c_{th} . The set of all edge and planar features are set as F_{ls} and F_{lf} , respectively. Then, from each row of a subregion, we extract two edge features with the maximum c as F_s and four planar features with the minimum c as F_f . The extracted laser feature point sets F_s , F_f , F_{ls} and F_{lf} are shown in Fig. 5a and b.

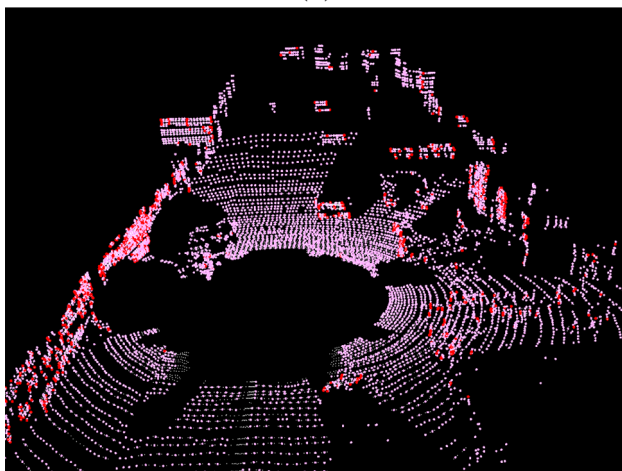
4.4 Data association

Since the monocular camera cannot provide scale information, here we estimate the depth of visual features by data association between an image and its corresponding labeled point cloud.

A schematic diagram of data association is illustrated in Fig. 6. Firstly, we transform the labeled point cloud from the lidar coordinate system to the camera coordinate system, and then project it onto the normalized plane of the camera as N_l . The visual features are also projected to the normalized plane as N_v . After that, for each visual feature $f \in N_v$ we perform the following steps.



(a)



(b)

Fig. 5 The 3D visual features F_v (yellow) from the data association and the extracted laser features F_s (red) and F_l (pink) are shown together in **a**. The laser features F_{ls} (red) and F_{lf} (pink) are illustrated in **b**

4.4.1 Neighborhood selection

To extract the local plane parameters around visual feature f , we firstly select its neighborhood N_n from the normalized lidar points. For the sake of algorithm efficiency, the commonly used KD-tree [22] is not adopted here since it would take too much time to construct the tree. Instead, we set a rectangle region around each visual feature on the normalized plane to select its neighborhood, as shown in Fig. 7a. The size of the rectangle is chosen to ensure that it contains at least two laser lines so that the singularity of plane estimation can be avoided.

4.4.2 Foreground selection

Considering that visual features prefer to lie on edges or corners of the environment, the neighborhood of a visual feature

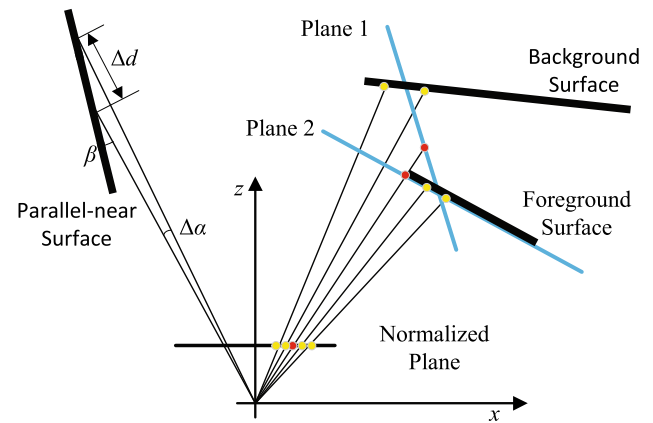


Fig. 6 A schematic diagram of data association. The labeled point cloud (yellow points) and the 2D visual features (red points) are projected to the normalized plane for neighborhood selection. Plane 1 is acquired by fitting with the original neighborhood set, which is deviated by the laser points on the background surface. By using the label information, laser points located on the foreground surface are selected and used to estimate the local tangent plane (Plane 2) of the foreground surface. The 3D visual feature with depth restored is acquired through calculating the intersection between the fitted plane (Plane 2) and the ray corresponding to f

usually contains lidar points on both foreground and background surface. Therefore, directly using the neighbor points to fit the plane would probably lead to wrongly estimated depth. To deal with this, we group the neighborhood N_n into separated sets according to each point's label assigned in the point cloud segmentation part, and select the nearest set to the sensor as the foreground set N_f .

4.4.3 Depth estimation

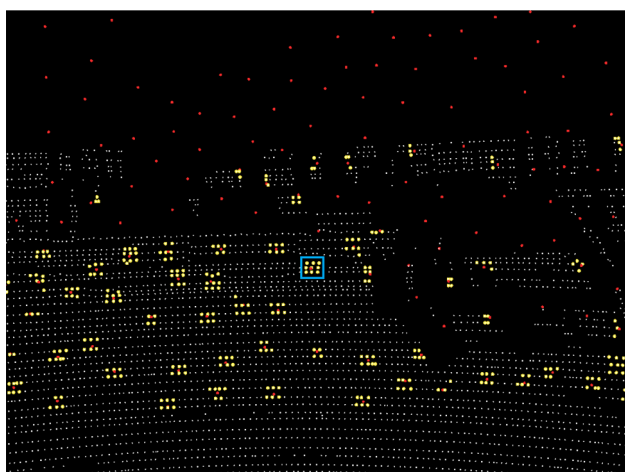
From the points in N_f , we choose three points that constitute a triangle with the maximum area. If the area is smaller than a given threshold, we would abandon this estimation to avoid wrongly estimated depth. Then, we fit the plane with these three points and get the plane parameters. Finally, we restore the depth of f through calculating the intersection between the fitted plane and the ray of sight corresponding to f ,

$$\mathbf{X}_{3d}^C = \left| \frac{d}{\mathbf{n}^T \mathbf{X}_{2d}^C} \right| \mathbf{X}_{2d}^C \quad (2)$$

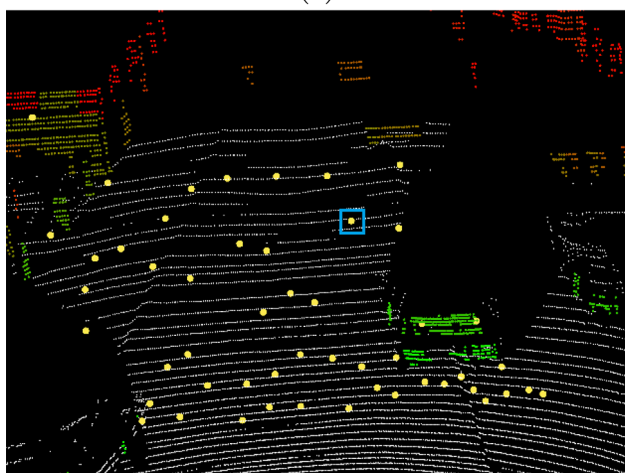
where \mathbf{X}_{2d}^C is the normalized visual feature with $z = 1$, \mathbf{n} and d are parameters of the fitted plane $\mathbf{n}^T \mathbf{X} + d = 0$, \mathbf{X}_{3d}^C is the estimated 3D visual feature with depth restored.

4.4.4 Validation

To be accepted as a valid estimation, the angle between the tangent of the fitted local plane and the ray of the visual feature should be smaller than a threshold (10°). This is because



(a)



(b)

Fig. 7 Choosing a neighborhood for a visual feature and restoring its depth by data association. In **a**, we set a rectangle region (blue) for a visual feature to determine its neighborhood. In **b**, we acquire a 3D visual feature (inside the blue rectangle) through data association between the original 2D visual feature and its neighbor lidar points

the noise of depth estimation is too large when calculating the intersection point of the plane and the ray, as shown in Fig. 6. Besides, the distance from the estimated 3D point to the sensor origin should be less than another threshold (50m), since the point cloud tends to be very sparse at a distance and may lead to failure of plane estimation.

By using the proposed method of data association, we can restore the depth information of visual features and get the set of 3D visual features F_v in the lidar coordinate system after back-projection, as illustrated in Fig. 7b.

Algorithm 1: Visual Lidar Odometry

Input: $\bar{F}_{ls}^k, \bar{F}_{lf}^k, \bar{F}_v^k$ and $T_{L_k}^{L_{k-1}}$ from the last frame; $F_s^{k+1}, F_f^{k+1}, F_v^{k+1}, F_{ls}^{k+1}$ and F_{lf}^{k+1} from the current frame;

Output: $\bar{F}_{ls}^{k+1}, \bar{F}_{lf}^{k+1}$ and $\bar{F}_v^{k+1}; T_{L_{k+1}}^{L_k};$
 $T_{L_{k+1}}^{L_k} \leftarrow T_{L_k}^{L_{k-1}};$

for a given number of iterations **do**

Use $T_{L_{k+1}}^{L_k}$ to transform F_s^{k+1}, F_f^{k+1} and F_v^{k+1} to the lidar coordinate system at time t_k based on (3) and (4), and get $\bar{F}_s^{k+1}, \bar{F}_f^{k+1}$, and \bar{F}_v^{k+1} ;

for each edge feature in \bar{F}_s^{k+1} **do**

Find an edge line as its correspondence in \bar{F}_{ls}^k , then calculate the point-to-line distance as its Euclidean residual based on (5) and add it to the residual set E ;

end

for each planar feature in \bar{F}_f^{k+1} **do**

Find a planar patch as its correspondence in \bar{F}_{lf}^k , then calculate the point-to-plane distance as its Euclidean residual based on (6) and add it to the residual set E ;

end

for each visual feature in \bar{F}_v^{k+1} **do**

Find a visual feature with the same ID as its correspondence in \bar{F}_v^k , then calculate the point-to-point distance as its Euclidean residual based on (7) and add it to the residual set E ;

end

Compute the Jacobian matrix of each residual in E with respect to $T_{L_{k+1}}^{L_k}$;

Update $T_{L_{k+1}}^{L_k}$ as a nonlinear optimization problem using the L-M method;

if the optimization converges **then**

break;

end

end

$T_{L_{k+1}}^{L_k} \leftarrow T_{L_k}^{L_{k-1}};$

Use $T_{L_{k+1}}^{L_k}$ to transform $F_{ls}^{k+1}, F_{lf}^{k+1}$ and F_v^{k+1} to the lidar coordinate system at time t_{k+1} based on (3) and (4), and get $\bar{F}_{ls}^{k+1}, \bar{F}_{lf}^{k+1}$ and \bar{F}_v^{k+1} ;

5 Tightly coupled visual lidar odometry

In this section, we proceed with a tightly coupled visual lidar odometry for high-accuracy and reliable pose estimation. Our approach starts with a visual lidar odometry to tightly fuse 3D visual features and 3D laser features in a uniform optimization framework, and then refines the estimated pose and registers the point cloud to a map through a scan-to-map matching.

5.1 Visual lidar odometry

Lidar-based methods [30] can only make use of geometric information of the environment to select laser features, which may lead to degeneracy of constraints [28] in symmetrical scenarios. As shown in Fig. 8a, in a symmetrical tunnel, if we only constrain the planar features (pink squares) and the edge

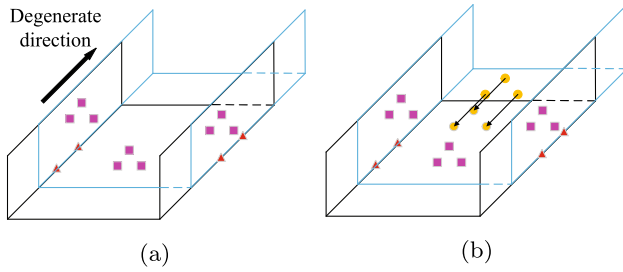


Fig. 8 Degeneracy of constraints in a symmetrical tunnel. In **a**, we only constrain the planar features and the edge features to their correspondences, which lead to a degeneracy in the tunnel direction. In **b**, the introduction of 3D visual features provides additional point-to-point constraints

features (red triangles) of current frame to their corresponding planes and edges in last frame, the motion along the tunnel direction cannot be estimated since there are no effective constraints in this direction. Considering that visual features are based on texture information, they are naturally complementary with laser features. Here, we introduce the 3D visual features (yellow circles) we acquired in data association to add additional point-to-point constraints, as illustrated in Fig. 8b.

Since sensor data is received continuously when the platform is moving, as shown in Fig. 2, we define p_i as a feature point received at time t_i in lidar coordinate system L_i . Let t_k be the time of previous frame k , t_{k+1} be the time of current frame $k+1$ and $\mathbf{T}_{L_{k+1}}^{L_k}$ be the motion estimation of lidar from t_k to t_{k+1} . Here we can utilize linear interpolation to acquire the motion estimation from the time of previous frame to the time when point p_i is received,

$$\mathbf{T}_{L_i}^{L_k} = \left(\frac{t_i - t_k}{t_{k+1} - t_k} \right) \otimes \mathbf{T}_{L_{k+1}}^{L_k} \quad (3)$$

where we have $\mathbf{T}_{L_i}^{L_k} \in [0.5 \otimes \mathbf{T}_{L_{k+1}}^{L_k}, 1.5 \otimes \mathbf{T}_{L_{k+1}}^{L_k}]$ for laser points and $\mathbf{T}_{L_i}^{L_k} = \mathbf{T}_{L_{k+1}}^{L_k}$ for visual features since they are exactly captured at time t_{k+1} . \otimes represents a linear interpolation of a transformation matrix at a given proportion.

Then, we can transform each point in the set of laser features F_s^{k+1} , F_f^{k+1} and visual features F_v^{k+1} from the lidar coordinate system when it is received to the lidar coordinate system at time t_k ,

$$\mathbf{X}_i^{L_k} = \mathbf{T}_{L_i}^{L_k} \mathbf{X}_i^{L_i} \quad (4)$$

where $\mathbf{X}_i^{L_i}$ is the coordinates of point p_i in L_i and $\mathbf{X}_i^{L_k}$ is the coordinates of p_i in L_k . The transformed sets are named \tilde{F}_s^{k+1} , \tilde{F}_f^{k+1} and \tilde{F}_v^{k+1} , respectively.

Using the previous motion estimation $\mathbf{T}_{L_k}^{L_{k-1}}$, the feature sets $F_{l_s}^k$, $F_{l_f}^k$ of previous frame are also transformed to the

same coordinate system L_k as $\tilde{F}_{l_s}^k$ and $\tilde{F}_{l_f}^k$. For each edge point in \tilde{F}_s^{k+1} we can find an edge line as its correspondence in $\tilde{F}_{l_s}^k$, and for each planar point in $\tilde{F}_{l_f}^k$ we can find a planar patch as its correspondence in $\tilde{F}_{l_f}^k$. The detailed procedures of finding correspondences of laser features can be found in [4]. The Euclidean distance of an edge point to its edge line and the Euclidean distance of a planar point to its planar patch can be written as,

$$E_s = f(\mathbf{X}_i^{L_i}, \mathbf{T}_{L_{k+1}}^{L_k}), i \in F_s^{k+1} \quad (5)$$

$$E_f = f(\mathbf{X}_i^{L_i}, \mathbf{T}_{L_{k+1}}^{L_k}), i \in F_f^{k+1} \quad (6)$$

For each 3D visual feature in \tilde{F}_v^{k+1} , we can also find its correspondence in F_v^k by matching feature ID that we assigned in feature tracking. The distance of a visual feature to its correspondence can be written as,

$$E_v = f(\mathbf{X}_i^{L_i}, \mathbf{T}_{L_{k+1}}^{L_k}), i \in F_v^{k+1} \quad (7)$$

Combining (5), (6) and (7) for all laser and visual features, we minimize the sum of residuals in a common cost function,

$$\arg \min_{\mathbf{T}_{L_{k+1}}^{L_k}} \left\{ \sum_{F_s^{k+1}} \omega_s E_s^2 + \sum_{F_f^{k+1}} \omega_f E_f^2 + \sum_{F_v^{k+1}} \omega_v E_v^2 \right\} \quad (8)$$

where ω_s , ω_f and ω_v are the weights of each kind of features.

By computing the Jacobian matrix with respect to $\mathbf{T}_{L_{k+1}}^{L_k}$, we can minimize (8) to zero by solving it as a nonlinear optimization problem with the Levenberg–Marquardt method. The pseudocode of this tightly coupled visual lidar odometry is illustrated in Algorithm 1.

5.2 Scan-to-map matching

Scan-to-map matching registers the distortion-removed point cloud $\tilde{F}_{l_s}^{k+1}$ and $\tilde{F}_{l_f}^{k+1}$ to its surrounding map $\bar{M}_{l_s}^k$ and $\bar{M}_{l_f}^k$ at a lower frequency (2Hz), where the motion estimation from visual lidar odometry is further refined and the map is updated with the newcome point cloud.

The whole environment map is stored distributedly in each keyframe as,

$$M^k = \left\{ \{\tilde{F}_{l_s}^1, \tilde{F}_{l_f}^1\}, \dots, \{\tilde{F}_{l_s}^k, \tilde{F}_{l_f}^k\} \right\} \quad (9)$$

The surrounding map of current frame $k+1$ is obtained by,

$$\bar{M}_{l_s}^k = \bigcup_{1 \leq m \leq k} \{\tilde{F}_{l_s}^m\}, \text{ s.t. } |\mathbf{t}_{L_{k+1}}^{L_m}| < \mathbf{t} \quad (10)$$

$$\bar{M}_{l_f}^k = \bigcup_{1 \leq m \leq k} \{\tilde{F}_{l_f}^m\}, \text{ s.t. } |\mathbf{t}_{L_{k+1}}^{L_m}| < \mathbf{t} \quad (11)$$

where $t_{L_{k+1}}^{L_m}$ is the relative translation from L_{k+1} to L_m

Similarly to visual lidar odometry, we use the Levenberg–Marquardt method to solve this optimization problem by minimizing all the residuals. Then, in transform fusion, the low-frequency but high-accuracy motion estimation from scan-to-map matching is fused with the high-frequency output of visual lidar odometry to get the 10Hz fused pose.

6 Loop closure

In this section, we detect visual loop using bag-of-words and detect vicinity loop based on the pose estimation of the odometry. When a loop is detected, we construct the loop constraint between the current frame and its loop frame employing ICP method. The loop constraint is added to a 6-DOF global pose graph which is optimized for global consistency.

6.1 Keyframe selection

We maintained a keyframe database for loop detection and global optimization. A keyframe consists of the lidar pose, the laser features and the image features. A new keyframe is selected and added to the keyframe database when the translation between the current frame and the previous keyframe exceeds a threshold. The transformation between the new keyframe and the previous keyframe is added as an odometry constraint to the global pose graph.

6.2 Loop detection

The visual loop detection is performed utilizing the DboW2 [9], which is based on Bag-of-Words place recognition with BRIEF descriptors [3]. When a new keyframe is added, we extract 500 FAST corner features from the image and use BRIEF descriptors to describe these features. The descriptors are used as a word to query the visual vocabulary. If a loop candidate is detected, we perform the temporal consistency check by limiting the time interval between the current keyframe and its loop keyframe to be greater than a given threshold t_{th} . Besides, we restrain the translation between the current keyframe and its loop keyframe to be smaller than a radius R_1 , which is called geometrical consistency check.

Nevertheless, the visual loop detection is susceptible to illumination change and limited by the camera’s field-of-view. Considering the lidar has a horizontal FOV of 360° and can work in any light situation, we implement a vicinity detection which is purely based on the pose estimation of the odometry. We construct a 3D KD-tree using the translations of all keyframes in the world coordinate system, and search for the closest neighbor for the current keyframe. If the relative distance is smaller than radius R_2 , $R_2 < R_1$, we will treat it as a loop. Likewise, we do not consider keyframes

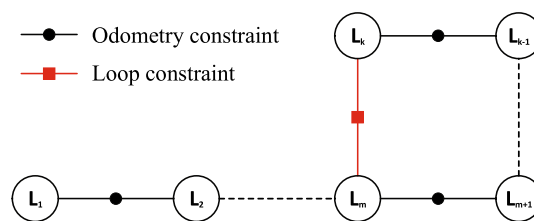


Fig. 9 The global pose graph consists of odometry constraint factor and loop constraint factor

whose time interval from the current keyframe is less than t_{th} .

6.3 Global pose graph optimization

For each loop candidate, the loop constraint is calculated by registering the laser features of current frame to the surrounding map of the loop history frame using ICP method [1]. If the ICP registration succeed, we finally treat this loop as a true loop and add this loop constraint to the global pose graph. The global pose graph contains all the keyframe lidar poses as variables, and is constrained by odometry factors and loop factors as illustrated in Fig. 9. The residuals of these two kinds of factors can be uniformly written as,

$$e_{ij} = \ln \left(\Delta \mathbf{T}_{ij} (\mathbf{T}_i^w)^{-1} \mathbf{T}_j^w \right)^\vee \tag{12}$$

where \mathbf{T}_i^w and \mathbf{T}_j^w are two keyframe poses in the world coordinate system, $\Delta \mathbf{T}_{ij}$ is a odometry or loop constraint factor which stands for a measurement of relative transformation between keyframe i and j . The residual e_{ij} is represented in the form of Lie algebra.

The whole global pose graph is optimized by,

$$\arg \min_T \left\{ \sum_{(i,j) \in O} \|e_{ij}\|^2 + \sum_{(i,j) \in L} h \left(\|e_{ij}\|^2 \right) \right\} \tag{13}$$

where O is the set of all odometry constraints, L is the set of all loop constraints and T is the set of keyframe poses to be optimized. $h(\cdot)$ is Huber function to reduce the influence of potential incorrect loop constraints.

To ensure real-time performance, we utilize iSAM2 [13] to incrementally optimize the global pose graph. After each loop optimization, the global environment point cloud map is also updated using the updated keyframe poses.

7 Experiments

We evaluate our method utilizing the KITTI datasets [11]. As shown in Fig. 10, the KITTI datasets are recorded on a



Fig. 10 The sensor suite platform of KITTI. The KITTI datasets are recorded on a passenger vehicle which is equipped with a pair of color cameras, two greyscale cameras, a Velodyne HDL-64E laser scanner and an OXTS GPS/IMU inertial navigation system which is used for providing ground-truth poses. This picture is taken from <http://www.cvlibs.net/datasets/kitti/>

passenger vehicle which is equipped with a pair of color cameras, two greyscale cameras, a Velodyne HDL-64E laser scanner and an OXTS GPS/IMU inertial navigation system.

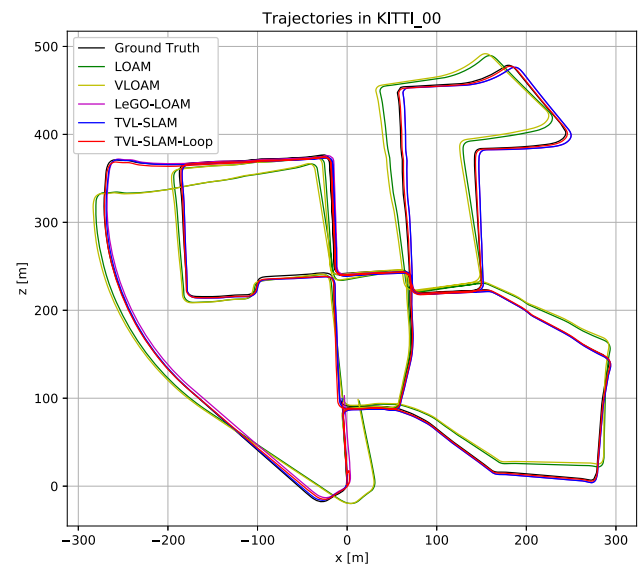
The extrinsic parameters between all sensors and the intrinsic parameters of the cameras and the lidar are already calibrated and provided by KITTI. For time synchronization, when the lidar rotates to face the front of the vehicle, the camera shutter will be triggered mechanically using a reed contact. Therefore, the timestamp of each frame of picture corresponds exactly to the middle moment of a lidar rotation, which is used as the unified timestamp of one frame of data. In addition, due to the high frequency of the GPS/IMU system (100Hz), KITTI selects the GPS/IMU data closest to the timestamp of each picture as the corresponding ground-truth.

The sensor suite that we employ in this paper includes the Velodyne HDL-64E lidar, a monocular greyscale camera and the inertial navigation system which is used for providing ground-truth poses.

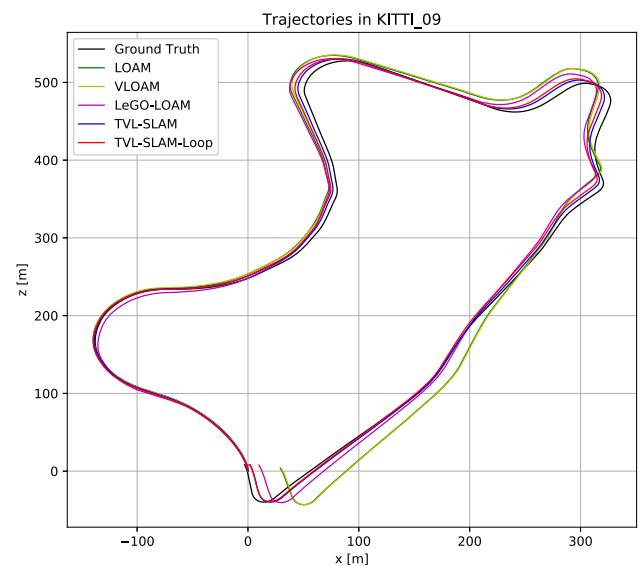
Experiments are conducted to compare our method with the state-of-the-art laser-based methods LOAM [30] and LeGO-LOAM [23] and the fusion-based method VLOAM [29] on a laptop with an i5-10210U CPU. All algorithms are implemented in the C++ language and run on the Robot Operating System (ROS) under Ubuntu environment.

7.1 Accuracy analysis

Since LOAM and VLOAM cannot run in real time when processing the KITTI dataset, we set it running at 50% of the real-time speed. TVL-SLAM and TVL-SLAM-Loop are our



(a)



(b)

Fig. 11 Trajectories from different methods in sequence KITTI_00 and KITTI_09

methods with the loop closure module disabled and enabled, respectively. Both LeGO-LOAM and our methods run at the real-time speed. The relative pose error (RPE), which is computed by comparing the final pose with the initial pose of the whole trajectory, is used to evaluate the performance of pose estimation. The rotational and translational errors are listed in Table 1 for different methods using sequence KITTI_00 and KITTI_09. The horizontal trajectories of sequence KITTI_00 and KITTI_09 are shown in Fig. 11. The absolute pose error (APE), which gives the root-mean-square error between the ground truth and the pose estimation, is illustrated in Fig. 12.

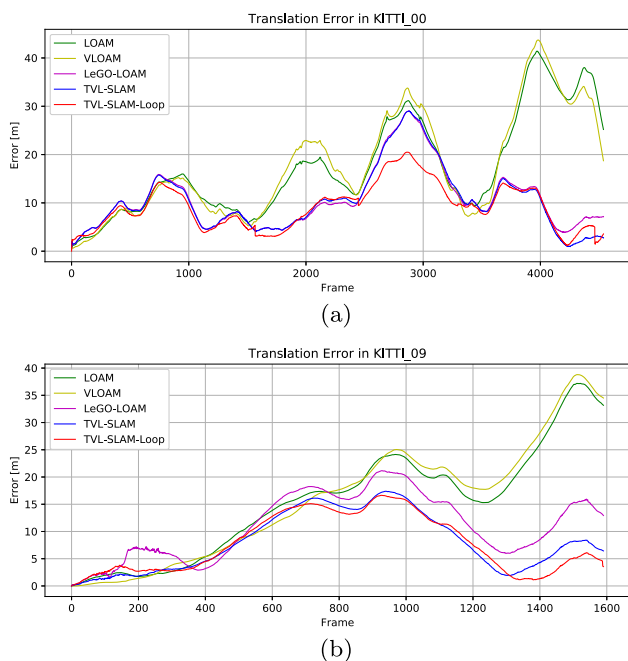


Fig. 12 Absolute pose error w.r.t. the ground truth in sequence KITT00 and KITT09

From the results in Table 1, we see that our method TVL-SLAM outperforms LOAM, VLOAM and LeGO-LOAM in all sequences. LOAM and VLOAM has the largest error since it does not have the step of cloud segmentation and thus needs to deal with a large number of unreliable or unnecessary feature points. Compared with LeGO-LOAM, our method reduces the translation error by 17%–43% and the rotational error by 19%–25%, which is mainly due to the introduction of visual feature points.

In sequence KITT00, when the loop closure module is enabled our method TVL-SLAM-Loop can further dramatically reduce the translational and rotational error to around 0.3 m and 0.005 degrees, respectively.

7.2 Running time analysis

The mean runtime for each module in LOAM, VLOAM, LeGO-LOAM and TVL-SLAM is shown in Table 2. Since the cloud segmentation procedure of our method can eliminate a large number of unreliable and unnecessary laser

features, the average runtime of the odometry and mapping are reduced to only 13.5% and 21.6% of LOAM, and 10.1% and 48.2% of VLOAM, respectively.

Compared with LeGO-LOAM, which only need to extract laser features, the introduction of visual features in our method increases the runtime of data preprocessing and odometry by 19.3ms and 3.8ms, respectively. Nevertheless, the tightly coupled visual lidar odometry outputs more accurate pose estimation, which makes it easier for the mapping module to refine the pose and reduces the runtime of mapping to 67% of LeGO-LOAM. Despite the introduction of visual features, the total runtime of our method still takes the least of the three methods. The total runtime of LOAM exceeds four times of the lidar cycle (100ms), which makes it impossible to run in real time. But by setting the mapping part running at 2Hz, our method can run in real time with a 64-line 3D lidar.

7.3 Mapping result analysis

Since the mapping quality of VLOAM is almost the same as that of LOAM, here we only compare our method with LOAM and LeGO-LOAM.

The mapping results of KITT00 dataset using LOAM, LeGO-LOAM and our method TVL-SLAM-Loop are shown in Fig. 13. We can find that our method can build a smoother and global-consistent point cloud map than LOAM and LeGO-LOAM, which has more points that do not coincide at the same location. Figure 14 shows mapping results at three loop locations before and after performing loop closure using our method. We see that our method can accurately detect the loops, complete the optimization of the global pose graph and establish a globally consistent map.

8 Conclusion

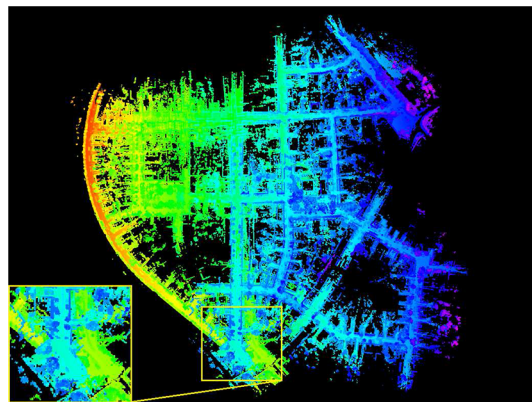
A novel tightly coupled visual lidar odometry method was presented in this paper. By combining visual feature tracking, data association, cloud segmentation and laser feature extraction, the system can acquire depth-recovered 3D visual features and fuse them with 3D laser features in a joint-optimization framework. The loop closure module, which

Table 1 Relative translational and rotational errors w.r.t. the ground truth

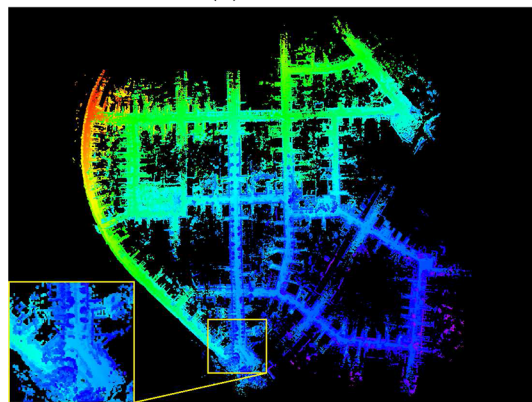
Errors	Sequence	LOAM (0.5×)	VLOAM (0.5×)	LeGO-LOAM	TVL-SLAM	TVL-SLAM-Loop
Total	KITT00	22.1582	16.4135	6.4323	5.3518	0.3071
Tra.(m)	KITT09	29.1038	30.2503	13.4184	6.5164	3.7063
Total	KITT00	7.3637	5.4982	1.4455	1.1688	0.0047
Rot.(°)	KITT09	25.3295	24.5960	4.2194	3.1533	2.5973

Table 2 Mean runtime analyses (ms)

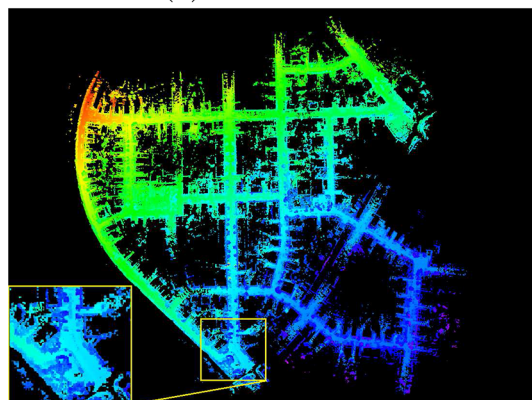
Procedures	LOAM (0.5×)	VLOAM (0.5×)	LeGO-LOAM	TVL-SLAM
Preprocessing	N/A	N/A	26.9	46.2
Odometry	95.6	127.8	8.8	12.9
Mapping	341.1	152.8	110.2	73.7
Total	437.6	280.6	145.9	132.8



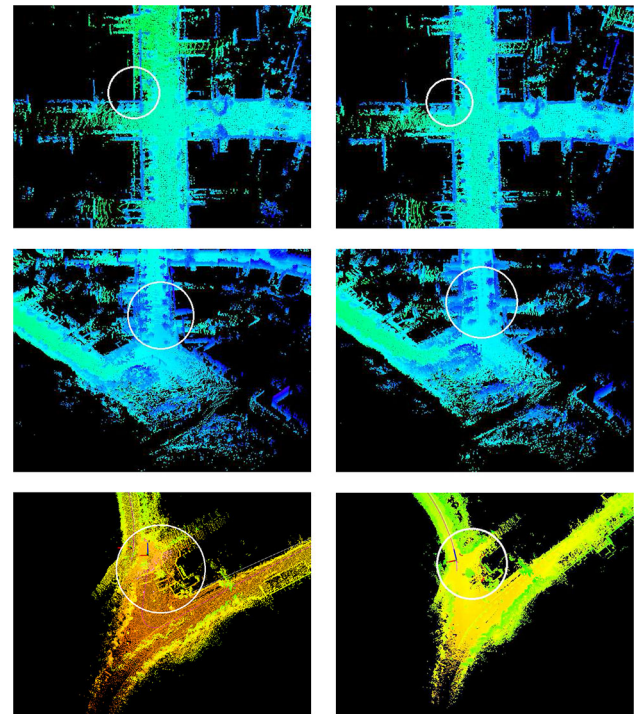
(a) LOAM



(b) LeGO-LOAM



(c) TVL-SLAM-Loop

Fig. 13 Mapping results of LOAM, LeGO-LOAM and TVL-SLAM-Loop

(a) Before loop closure (b) After loop closure

Fig. 14 Mapping results before and after loop closure. In **a** and **b**, we use white circles to point out the obvious inconsistent places before and after the loop closure

utilizes visual and vicinity loop detection together, further constructs loop constraints to remove drift accumulation. Despite that more time is spent in data preprocessing due to the introduction of visual features, the whole system can still run in real-time with the 64-line lidar and provide high-accuracy pose estimation together with a global-consistent point cloud map of the traversed environment. Experiments on public dataset show that our method outperforms the state-of-the-art lidar-based and fusion-based methods in accuracy, runtime and mapping results.

Author Contributions All authors contributed to the study conception and design. All authors commented on previous versions of the manuscript, and approved the final manuscript.

Funding This work was supported in part by the National Natural Science Foundation of China (61973099).

Declarations

Conflicts of interest The authors declare that there is no conflict of interests regarding the publication of this paper.

Ethics approval The authors declare that there are no ethics issues regarding the publication of this paper.

References

1. Besl PJ, McKay ND (1992) Method for registration of 3-d shapes. In: Sensor fusion IV: control paradigms and data structures, vol 1611. International Society for Optics and Photonics, pp 586–606
2. Bogoslavskyi I, Stachniss C (2016) Fast range image-based segmentation of sparse 3d laser scans for online operation. In: 2016 IEEE/RSJ international conference on intelligent robots and systems (IROS). IEEE, pp 163–169
3. Calonder M, Lepetit V, Strecha C, Fua P (2010) Brief: binary robust independent elementary features. In: European conference on computer vision. Springer, pp 778–792
4. Cheng J, Wang C, Meng MQH (2019) Robust visual localization in dynamic environments based on sparse motion removal. *IEEE Trans Autom Sci Eng* 1–12
5. Curless B, Levoy M (1996) A volumetric method for building complex models from range images. In: Proceedings of the 23rd annual conference on Computer graphics and interactive techniques, pp 303–312
6. De Croce M, Pire T, Bergero F (2019) Ds-ptam: distributed stereo parallel tracking and mapping slam system. *J Intell Robot Syst* 95(2):365–377
7. Deschaud JE (2018) Imls-slam: scan-to-model matching based on 3d data. In: 2018 IEEE international conference on robotics and automation (ICRA). IEEE, pp 2480–2485
8. Engel J, Stückler J, Cremers D (2015) Large-scale direct slam with stereo cameras. In: 2015 IEEE/RSJ international conference on intelligent robots and systems (IROS). IEEE, pp 1935–1942
9. Gálvez-López D, Tardos JD (2012) Bags of binary words for fast place recognition in image sequences. *IEEE Trans Robotics* 28(5):1188–1197
10. Gao X, Wang R, Demmel N, Cremers D (2018) Ldso: direct sparse odometry with loop closure, pp 2198–2204
11. Geiger A, Lenz P, Urtasun R (2012) Are we ready for autonomous driving? the kitti vision benchmark suite. In: 2012 IEEE conference on computer vision and pattern recognition. IEEE, pp 3354–3361
12. Himmelsbach M, Hundelshausen FV, Wuensche HJ (2010) Fast segmentation of 3d point clouds for ground vehicles. In: 2010 IEEE intelligent vehicles symposium. IEEE, pp 560–565
13. Kaess M, Johannsson H, Roberts R, Ila V, Leonard J, Dellaert F (2011) isam2: incremental smoothing and mapping with fluid relinearization and incremental variable reordering. In: 2011 IEEE international conference on robotics and automation. IEEE, pp 3281–3288
14. Kim D, Shin S, Kweon IS (2018) On-line initialization and extrinsic calibration of an inertial navigation system with a relative preintegration method on manifold. *IEEE Trans Autom Sci Eng* 15(3):1272–1285
15. Li G, Yu L, Fei S (2020) A binocular msckf-based visual inertial odometry system using lk optical flow. *J Intell Robot Syst* 100(3):1179–1194
16. Lucas BD, Kanade T et al (1981) An iterative image registration technique with an application to stereo vision
17. Lynen S, Achtelik MW, Weiss S, Chli M, Siegwart R (2013) A robust and modular multi-sensor fusion approach applied to mav navigation. In: 2013 IEEE/RSJ international conference on intelligent robots and systems. IEEE, pp 3923–3929
18. McCormac J, Handa A, Davison AJ, Leutenegger S (2017) Semanticfusion: dense 3d semantic mapping with convolutional neural networks, pp 4628–4635
19. Mourikis AI, Roumeliotis SI (2007) A multi-state constraint kalman filter for vision-aided inertial navigation. In: Proceedings 2007 IEEE international conference on robotics and automation. IEEE, pp 3565–3572
20. Mur-Artal R, Tardós JD (2017) Orb-slam2: an open-source slam system for monocular, stereo, and rgb-d cameras. *IEEE Trans Robotics* 33(5):1255–1262
21. Qin T, Li P, Shen S (2018) Vins-mono: a robust and versatile monocular visual-inertial state estimator. *IEEE Trans Robotics* 34(4):1004–1020
22. Ramasubramanian V, Paliwal KK (1992) Fast k-dimensional tree algorithms for nearest neighbor search with application to vector quantization encoding. *IEEE Trans Sig Process* 40(3):518–531
23. Shan T, Englot B (2018) Lego-loam: lightweight and ground-optimized lidar odometry and mapping on variable terrain. In: 2018 IEEE/RSJ international conference on intelligent robots and systems (IROS). IEEE, pp 4758–4765
24. Shao W, Vijayarangan S, Li C, Kantor G (2019) Stereo visual inertial lidar simultaneous localization and mapping. [arXiv preprint arXiv:1902.10741](https://arxiv.org/abs/1902.10741)
25. Shi J et al (1994) Good features to track. In: 1994 Proceedings of IEEE conference on computer vision and pattern recognition. IEEE, pp 593–600
26. Wang Z, Zhang J, Chen S, Yuan C, Zhang J, Zhang J (2019) Robust high accuracy visual-inertial-laser slam system. In: 2019 IEEE/RSJ international conference on intelligent robots and systems (IROS). IEEE, pp 6636–6641
27. Yang Z, Shen S (2017) Monocular visual-inertial state estimation with online initialization and camera-imu extrinsic calibration. *IEEE Trans Autom Sci Eng* 14(1):39–51
28. Zhang J, Kaess M, Singh S (2016) On degeneracy of optimization-based state estimation problems. In: 2016 IEEE international conference on robotics and automation (ICRA). IEEE, pp 809–816
29. Zhang J, Singh S (2015) Visual-lidar odometry and mapping: low-drift, robust, and fast. In: 2015 IEEE international conference on robotics and automation (ICRA). IEEE, pp 2174–2181
30. Zhang J, Singh S (2017) Low-drift and real-time lidar odometry and mapping. *Autonom Robots* 41(2):401–416
31. Zhang J, Singh S (2018) Laser-visual-inertial odometry and mapping with high robustness and low drift. *J Field Robotics* 35(8):1242–1264
32. Zuiderveld KJ (1994) Contrast limited adaptive histogram equalization. *Graphics gems*, pp 474–485

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.