LiPMatch: LiDAR Point Cloud Plane based Loop-Closure

Jianwen Jiang[1], Jikai Wang[1], Peng Wang[2], Peng Bao[1], and Zonghai Chen[1]

***Abstract*—This paper presents a point clouds based loop-closure method to correct long-term drifts in Light Detection and Ranging based Simultaneous Localization and Mapping systems. In the method, we formulate each keyframe as a fully-connected graph with nodes representing planes. To detect loop-closures, the proposed method employs geometric restrictions to define a similarity metric to match current keyframe and those in the map. After similarity assessment, the candidate keyframes which comply with the geometric restrictions are further checked out successively by normal constraints of planes, and validated by an improved Iterative Closest Point method. The latter also provides relative pose transformation estimation between the current keyframe and the matched keyframe in the global reference frame. Experimental results demonstrate that the proposed method is able to fulfill fast and reliable loop-closure. To benefit the community by serving a benchmark for loop-closure, the entire system is made open source on GitHub[3].**

***Index Terms*—SLAM, Mapping, Localization, Range Sensing.**

I. Introduction

LOOP-closure represents one of the key challenges towards accurate Simultaneous Localization and Mapping (SLAM) solutions. As drift is inevitable when performing state estimation without global positioning information [1], [2], [3], reliable loop-closure becomes crucial for many robotic platforms. In recent years, Light Detection And Ranging (LiDAR) based SLAM [4], [5], [6], [7] have been extensively developed. The LiDAR sensor is capable of capturing geometric features stably in a considerably fine resolution, thus not suffering as much as computer vision when illuminations and viewpoints vary. However, there are still existing challenges in the LiDAR based loop-closure paradigm, since LiDAR point clouds only contain geometry information of the three-dimensional (3D) space, which makes it difficult for loop-closure detection due to the lack of textures and colors.

Manuscript received: March 29, 2020; Revised: May 20, 2020; Accepted: August 10, 2020.

This paper was recommended for publication by Editor Sven Behnke upon evaluation of the Reviewers' comments. This work was supported by the National Natural Science Foundation of China (Grant No. 91848111) and the Young Scientists Fund of the National Natural Science Foundation of China (Grant No. 61703387). (*Corresponding author: Zonghai Chen.*)

[1]Jianwen Jiang, Jikai Wang, Peng Bao, and Zonghai Chen are with Department of Automation, University of Science and Technology of China, Hefei 230027, PR China {jjwen, wangjk, baopeng}@mail.ustc.edu.cn, chenzh@ustc.edu.cn

[2]Peng Wang is with Department of Automatic Control and Systems Engineering, The University of Sheffield, Sheffield S1 3JD, United Kingdom peng.wang@sheffield.ac.uk

[3]https://github.com/jiangjianwen/LiPMatch

Digital Object Identifier (DOI): see top of this page.

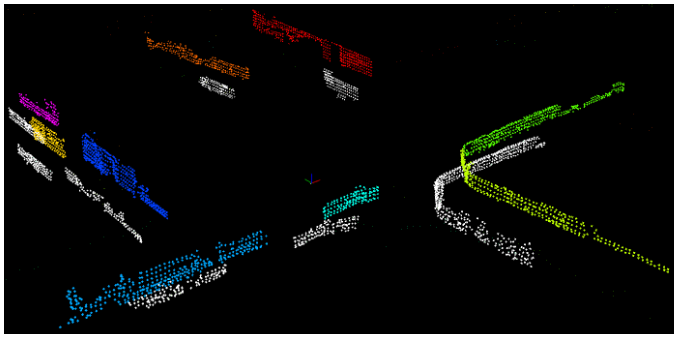
Fig. 1: An illustration of the presented loop-closure detection framework. The planes of the current keyframe are shown below (in white), and the corresponding planes of the keyframe in the map are shown above. Colors are used to indicate the point cloud plane segmentation.

In this paper, in order to alleviate the existing challenges in LiDAR based loop-closure, we develop a fast, reliable and complete loop-closure for multi-channel LiDAR-based SLAM, consisting of fast loop-closure detection and loop-closure correction, in which planes and their geometric relationships are exploited. Specifically, we propose a compact plane-based fully-connected graph representation of keyframes. Loop-closure detection is therefore considered as a graph matching problem: the graph representing the current keyframe is compared with those keyframes that are already integrated into the global map. To solve this problem, we exploit geometric characteristics of planes and their relative geometric relationships. For gaining in robustness, we introduce planes' normal constraints. To perform loop-closure correction, an improved Iterative Closest Point (ICP) which integrates plane constraints is proposed. The outputs of ICP can also benefit the graph matching results validation. The main advantages of our proposed method for loop-closure are: 1) it is fast and reliable. 2) It is capable of providing accurate loop-closure correction.

To summarize, our contributions comprise of four aspects:

(1) we develop a fast, reliable LiDAR-based loop-closure detection method for multi-channel LiDAR-based SLAM, in which the similarity of two keyframes is efficiently evaluated;

(2) we apply the geometric constraints between the matched planes to improve the ICP method in the process of loop-closure correction;

(3) we integrate our loop-closure detection and loop-closure correction methods into LOAM, setting up a complete and practical LiDAR-based SLAM system;

(4) we provide an affordable solution for LiDAR-based loop-closure by making our system open source on GitHub.

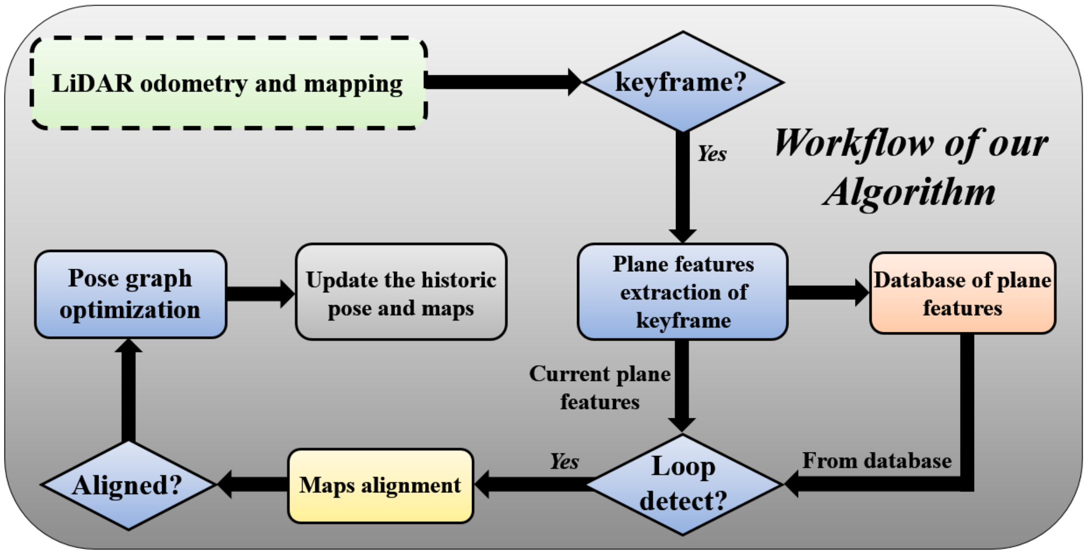
LiDAR odometry and mapping
keyframe?
Yes
Workflow of our Algorithm
Pose graph optimization
Update the historic pose and maps
Plane features extraction of keyframe
Database of plane features
Current plane features
From database
Yes
Aligned?
Maps alignment
Loop detect?

Fig. 2: The overview of our system.

The remainder of this paper is organized as follows. In Section II, we discuss the related work. Loop-closure detection is presented with details in Section III, Loop-closure correction using geometric features is given in Section IV. Experimental results demonstrating the effectiveness of our method for loop-closure are shown in Section V. Finally, Section VI concludes the paper.

II. Related Work

Detecting loop-closures from 3D data remains an open problem in SLAM. The problem has been addressed with different approaches, and we have identified three main trends.

Bosse and Zlot [8] extract keypoints directly from the point clouds and describe them with a 3D Gestalt descriptor. Keypoints then vote for their nearest neighbors in a vote matrix which is finally thresholded for detecting loop-closure.

Using global descriptors of the local point clouds for loop-closure is also proposed [9], which splits the cloud into overlapping grids and computes shape properties of each cell and combines them into a matrix of surface shape histograms. Similar to other works, these descriptors are compared for recognizing places.

While local keypoint features often lack descriptive power, global descriptors still struggle with invariance. Therefore, 3D objects are exploited for the loop-closure task. Somewhat analogous, seminal works on segment-based loop-closure detection are presented in recent years [10], [11], [12], [13]. For example, the SegMatch [10] achieves loop-closure by matching semantic features like buildings, trees, and vehicles, etc. A semi-handcrafted learning method is proposed in [14] for LiDAR point clouds using LocNets, which regards the loop-closure detection problem as a similarity modeling problem.

All methods mentioned above detect loop-closures by extracting and matching descriptors which are usually time consuming. Other works have been proposed to use geometric restrictions between objects for the loop-closure task, to mitigate the computational burden. Fernandez-Moral et al. [15], for instance, propose to perform loop-closure detection by detecting planes in 3D environments. The planes are accumulated in a graph and an interpretation tree is used to match sub-graphs. A final geometric consistency test is conducted over the planes in the matched sub-graphs. Their method is applied to small and indoor environments using RGB-D cameras. Our method is inspired by the paradigm in [15] which is modified to suit for outdoor scenes and multi-channel LiDAR.

After loop-closure detection, a point cloud registration method is used to correct the loop-closure. One classical method to cope with point cloud registration is ICP [16]. In recent years, ICP has developed many variants such as point-to-plane, point-to-line, and plane-to-plane [17], [18], [19]. Being an iterative method, ICP would easily get trapped into local minima with a poor initialization. General solutions such as [20] proceed according to the following procedures. First, feature points are extracted from point clouds. Second, feature matching is applied to determine the correspondences, and finally, the initialization is achieved by calculating the transformation between correspondences.

III. Loop-closure Detection

In this section, we describe our system for loop-closure detection from 3D point clouds. The proposed system is depicted in Fig. 2 and is mainly composed of two modules: plane extraction and parametrization, and graph matching. We give details in regard to each module of our system as follows.

A. Plane extraction and parametrization

Inspired by [10], for each incoming point cloud, we transform it into a global reference frame using the output of LOAM odometry, and accumulate each LiDAR point cloud into the current keyframe. A new keyframe will be created when the vehicle moves a certain distance. For each keyframe, the accumulated point clouds are segmented into a set of point clusters using the Euclidean clustering method [21]. However, the method requires the ground plane to be removed. We have coped with it by the method in [22]. We drop out those clusters comprise of less than 100 points and plane extraction is next applied on the remained clusters. We employ the region growing approach [10] provided by PCL [21] to extract planes.

A plane is represented by its normal $\mathbf{n}$ and the distance d from the plane to the original point of the global reference frame. In this way, a 3D point $\mathbf{p}$ which lies on the plane satisfies the equation

$$\mathbf{n} \cdot \mathbf{p} + d = 0. \tag{1}$$

Each plane P is described by a set of geometric features:

- $\mathbf{n}$ the normal vector,
- $\mathbf{c}$ the centroid,
- d the distance to the global reference frame,
- λ_0, λ_1, and λ_2 the eigenvalues reflecting the spatial distribution of point clouds,
- $\mathbf{l}$ the polygon contour points defining the convex hull of the plane,
- a the area of the plane.

In this paper, $\mathbf{n}$ and d are provided by PCL, and $\mathbf{c}$ is calculated by averaging out point coordinates on the plane. The calculation of λ_0, λ_1, and λ_2 with $\lambda_0 \geq \lambda_1 \geq \lambda_2$ is the same to our previous work [1]. The convex hull $\mathbf{l}$ is efficiently computed by the Jarvis March algorithm [23] from points on the plane, and a is calculated from the polygon contour points $\mathbf{l}$ [15].

B. Graph matching

Different from [15] that extracts planes for each single frame and represents the whole map as a graph, we extract planes for keyframes and represent each keyframe as a fully-connected graph of planes. The reasons are: 1) The method in [15] is applied to indoor environment, which is more likely to contain more planes than outdoor environments. 2) The method in [15] aims at place recognition in a map, whereas our method focuses on loop-closure detection in large scale outdoor environments. In this scenario, using keyframes would benefit the match efficiency. Besides, our method uses an adaptation parameter rather than distance difference to measure the similarity. In the graph, each node is formulated by geometric features of the corresponding plane.

In order to match two graphs, G_c, generated from the current keyframe, and G_p, generated from one previous keyframe, we employ geometric restrictions represented as two sets of unary and binary constraints. The unary constraints are used to check the correspondence of two single planes by comparing their geometric features. The binary constraints serve to validate whether two pairs of planes within G_c and G_p have similar relative spatial position, e.g. the distance between the centroids of the plane pairs within G_c is similar to the centroid distance between the plane pairs within G_p.

Algorithm 1 details the graph matching process. First, unary constraints are verified to get candidate matches between G_c and G_p. Second, the binary constraints are checked with the already matched planes. If all the constraints are satisfied, a match between planes of the two keyframes is accepted and the recursive process continues with updated arguments. The algorithm finishes when all the candidate matches have been explored and then returns a list of matched planes $\mathbf{L}_{FM}$.

In spite of the large amount of potential plane matches for this problem, most of them fail to satisfy unary constraints. As only simple operations such as 3D vector and scalar comparisons are performed, the evaluation of these restrictions requires very little computation. After unary and binary assessments, we choose the keyframe with the largest number of matched planes as the candidate keyframe. We further validate the candidate keyframe according to normals of planes and the fitness score of ICP.

1) Unary constraints: The unary constraints presented here are designed to reject incorrect matches of two planes. They are relatively weak constraints, meaning that a fairly relaxing threshold is set to avoid rejecting correct matches. In other words, a unary constraint should validate that two planes are distinct when their geometric characteristics have significant difference, but they lack information to confirm that these two observations belong to the same plane. This is because different planes can have the same characteristics such as normals.

Different from [15] that uses geometry and color information from RGB-D sensors, LiDAR point clouds only contain geometry information. Several unary constraints have been used here, which perform direct comparisons of the planes areas and spatial distribution of the contained points. If the planes are not satisfied with the following equations, the two planes are not matched.

$$R^{-1} < \frac{a^c}{a^p} < R, \tag{2}$$

$$R^{-1} < \frac{\lambda_{10}^c}{\lambda_{10}^p} < R, \tag{3}$$

$$R^{-1} < \frac{\lambda_{21}^c}{\lambda_{21}^p} < R, \tag{4}$$

where $\lambda_{10} = \lambda_1/\lambda_0$, $\lambda_{21} = \lambda_2/\lambda_1$, the superscript c and p mark the current graph G_c and the previous graph G_p, and R is a threshold parameter set for filtering out mismatches. The advantages of using the parameter R include: 1) It reduces the number of parameters compared with using distance difference as thresholds. 2) It increases the robustness of our method when scene scale changes.

2) Binary constraints: As for binary constraints, stemming from [15], we apply binary constraints to impose geometric restrictions on the relative positions of two pairs of planes. These constraints take account for providing robustness in the graph matching process by enforcing the consistency of the matched scenes. Three binary constraints

$$R^{-1} < \frac{\arccos(\mathbf{n}_i^c \cdot \mathbf{n}_j^c)}{\arccos(\mathbf{n}_{i'}^p \cdot \mathbf{n}_{j'}^p)} < R, \tag{5}$$

$$R^{-1} < \frac{\|\mathbf{c}_i^c - \mathbf{c}_j^c\|}{\|\mathbf{c}_{i'}^p - \mathbf{c}_{j'}^p\|} < R, \tag{6}$$

$$R^{-1} < \frac{\mathbf{n}_j^c \cdot (\mathbf{c}_i^c - \mathbf{c}_j^c)}{\mathbf{n}_{j'}^p \cdot (\mathbf{c}_{i'}^p - \mathbf{c}_{j'}^p)} < R, \tag{7}$$

are imposed to each pair of planes in a matched graph, where $\mathbf{n}_i^c$ and $\mathbf{n}_j^c$ are the normals of a pair of planes from graph G_c, and similarly $\mathbf{n}_{i'}^p$ and $\mathbf{n}_{j'}^p$ are the normals of a pair of planes from graph G_p, $\mathbf{c}_i^c$ and $\mathbf{c}_j^c$ are the centroids of a pair of planes from the graph G_c, and similarly $\mathbf{c}_{i'}^p$ and $\mathbf{c}_{j'}^p$ are the centroids of a pair of planes from the graph G_p.

3) Normal constraints of planes: After unary and binary constraint assessments, there is still chance that the detected loop-closure is incorrect. We observe that the failure tend to happen when all the matched planes have the same normal. To improve the robustness of our method, we further explore normal information. Specifically, for each matched planes in the current keyframe, if the maximum value of the angles between two normals is less than a threshold, as is shown in equation (8), the matched planes in the current keyframe are considered not distinguishable enough to detect loop-closures.

$$\max_{i,j}(\mathbf{n}_i^c \cdot \mathbf{n}_j^c) < th_\theta. \tag{8}$$

In addition, we use equation (9) to get rid of the cases where the difference between two keyframes' areas is significant.

$$R_a^{-1} < \frac{A^c}{A^p} < R_a, \tag{9}$$

where A^c is the sum of matched planes' area in the current keyframe, and A^p is that in the matched history keyframe.

When passing through all constraints introduced above, we determine whether loop-closure is detected according to the number of matched planes, n_{match}. We choose the keyframe with the largest number of matched planes as the candidate loop-closure. Generally, there is only one candidate keyframe. When there are multiple candidate keyframes, we choose the one with the lowest fitness score.

IV. Loop-closure correction

Once a loop-closure is detected, loop-closure correction is performed to compute relative pose between the two keyframes. The problem of loop-closure correction can be viewed as the registration between the current point cloud and the history point cloud. Since our ICP method is based on LOAM, we use the edge-to-edge and plane-to-plane features to iteratively calculate the relative pose.

Compared with the ICP in odometry block of LOAM, the relative pose transformation of two keyframes is larger and it is more likely to trap into a local minimum. In this paper, we exploit geometric features to alleviate this problem. The geometric features used here are the matched planes' normals. We firstly use the corresponding normals to achieve the initial relative pose transformation of the two point clouds. The initial rotation value is achieved as follows:

$$\begin{cases} \mathbf{M}_{cov} = \sum_{i=1}^N \mathbf{n}_i^c \times \mathbf{n}_i^p, \\ \mathbf{M}_v \mathbf{\Sigma}_{\text{svd}} \mathbf{M}_u = \text{SVD}(\mathbf{M}_{cov}), \\ \mathbf{M}_R = \mathbf{M}_v \times \mathbf{M}_u^{\text{T}}, \end{cases} \tag{10}$$

Algorithm 1: Employ geometric restrictions to search recursively for the best match between two graphs of planes G_c and G_p.

```
Input: Current graph $G_c$, list of planes of $G_c$ $\mathbf{L}_c$,
       previous graph $G_p$, list of planes of $G_p$ $\mathbf{L}_p$ and
       list of matched planes $\mathbf{L}_M$
Output: Final list of matched planes $\mathbf{L}_{FM}$
1  $\mathbf{L}_{FM}$ = MatchGraphs($\mathbf{L}_c$, $\mathbf{L}_p$, $\mathbf{L}_M$)
2  $\mathbf{L}_{FM}$ = $\mathbf{L}_M$
3  for each plane $P_c \in L_c$ do
4      for each plane $P_p \in L_p$ do
5          if EvalUnaryConstraints($P_c$, $P_p$) == F then
6              continue;
7          for each $P_c'$, $P_p' \in L_M$ do
8              if $P_c$, $P_c' \in G_c$ and $P_p$, $P_p' \in G_p$ then
9                  if EvalBinaryConstraints($P_c$, $P_c'$, $P_p$, $P_p'$)
                   == F then
10                     continue;
11         new_$\mathbf{L}_c$ = $\mathbf{L}_c$ - $P_c$
12         new_$\mathbf{L}_p$ = $\mathbf{L}_p$ - $P_p$
13         new_$\mathbf{L}_M$ = $\mathbf{L}_M \cup P_c$, $P_p$
14         result = MatchGraphs(new_$\mathbf{L}_c$, new_$\mathbf{L}_p$,
            new_$\mathbf{L}_M$)
15         if SizeOf(result) > SizeOf($\mathbf{L}_{FM}$) then
16             $\mathbf{L}_{FM}$ = result;
17 return $\mathbf{L}_{FM}$
```

where $\mathbf{M}_{cov}$ is the normal covariance matrix, $\mathbf{M}_v$ and $\mathbf{M}_u$ are the Singular Value Decomposition (SVD) results, and $\mathbf{M}_R$ is the determined rotation matrix.

Then, we add geometric constraints to the optimization problem, which helps to reduce the probability of falling into a local extremum. We use the Levenberg-Marquardt method [24] to solve the optimization problem. Our objective function to minimize ICP matching error given as

$$d^k = D^k + \sum_{i=1}^N \sum_{j=1}^M \mathbf{n}_i \cdot (\mathbf{T}^k \mathbf{p}_{ij} - \mathbf{c}_i), \tag{11}$$

where D^k represents the error function provided by LOAM at the k_{th} iteration, $\mathbf{T}^k$ represents the relative pose transformation estimation at the k_{th} iteration, $\mathbf{p}_{ij}$ represents the j_{th} point in the i_{th} matched plane from the history keyframe, $\mathbf{c}_i$ and $\mathbf{n}_i$ represent the centroid and normal of the i_{th} matched plane from the current keyframe.

After the correction, if the average distance of the corresponding points between two keyframes is less than 0.1 m, we regard that these two keyframes are aligned. If two keyframes are not aligned, we do not regard the two keyframes as loop-closure.

Once the two keyframes are aligned, we perform the pose graph optimization following the method in [25]. When it is finished, we update all the point clouds and plane parameters in the global reference frame.

V. Experiments and analysis

A. Datasets

We evaluate our method on the KITTI odometry benchmark [26], where we use point clouds from a vertical Velodyne HDL-64E S2 mounted on the roof of a car. Sequences 00 and 05 of the KITTI dataset and data from [27] are processed. Sequence 00 lasts 3.7 km (470 s) and suits our case as it contains one large loop where the vehicle revisits the previous scenarios for a stretch of 500 m. This portion with multiple traversals will therefore be used in the loop-closure detection experiment. Sequence 05 lasts 2.2 km (287 s) and is used for presenting the online operation of the framework. Data from [27] are used to demonstrate that our method can apply to various datasets.

B. Baselines

Scan Context [28] is a loop-closure detection method based on global descriptors. The method extracts global descriptors and performs loop-closure detection for each frame. The performance of Scan Context depends on the number of candidates from the KD tree, which has been taken as a criterion to divide Scan Context into Scan Context-50 and Scan Context-10. Overall, Scan Context-50 reveals better performance than Scan Context-10 due to more candidates are used to search for loop-closures, and consequently takes more time than Scan Context-10.

The batch version SegMatch [10] performs loop-closure detection every other distance based on the local map, which is similar to ours. The incremental version SegMatch [11] performs loop-closure detection for each frame and is based on incremental local maps. In order to improve the real-time performance, the incremental version SegMatch designs an incremental calculation method of local map maintenance, normal vector calculation, and local map segmentation. The loop-closure detection method of incremental version SegMatch is similar to that of the batch version.

C. Loop-closure detection performance

For segmentation, the maximum Euclidean distance between two points such that they are considered to belong to the same cluster is set to 0.8 m. We only consider segments that contain more than 100 points for the sake of efficiency. For plane extraction, the number of neighbours is set to 20 points, and the smoothness threshold is set to 15 degrees.

As is shown in Fig. 3, we have computed the ROC curves with different unary constraints parameter ranging from 1.3 to 6.0, while the binary constraints parameter remains unchanged. It is worth to mention that when R becomes large, the unary constraints are greatly relaxed, until they no longer impose any restrictions on rejecting mismatches.

Fig. 3 shows that the true positive rate reaches the maximum value fastly and remains almost unchanged while the unary constraints parameter increases. Overall, the true positive rate is relatively stable which means the unary constraints have little effect on the true positive rate. Especially, when the unary constraints parameter is relatively large, the unary constraints almost do not affect the true positive rate.

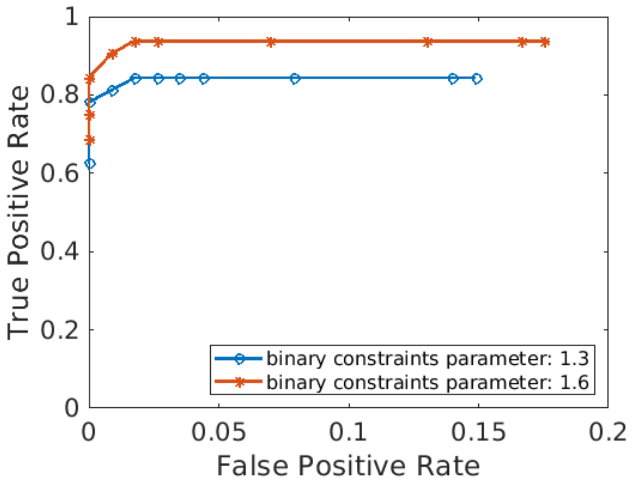

Fig. 3: The ROC curves produced by setting the binary constraints parameter to 1.3 and 1.6, respectively. For each curve, the marks (stars and circles) from left to right correspond to the unary constraints parameter increasing from 1.3 to 6.0.

Besides, the false positive rate gradually reaches the maximum value while the unary constraints parameter increases. The false positive rate's maximum value is relatively small, which means when we remove the unary constraints, we can get almost similar results. But the unary constraints help to filter out some obvious mismatching to reduce the number of candidate matches for binary constraints, which can benefit the computational efficiency. TABLE I shows the difference of graph matching time with and without unary constraints. Fig. 4 shows the number of matched planes after unary and binary assessments.

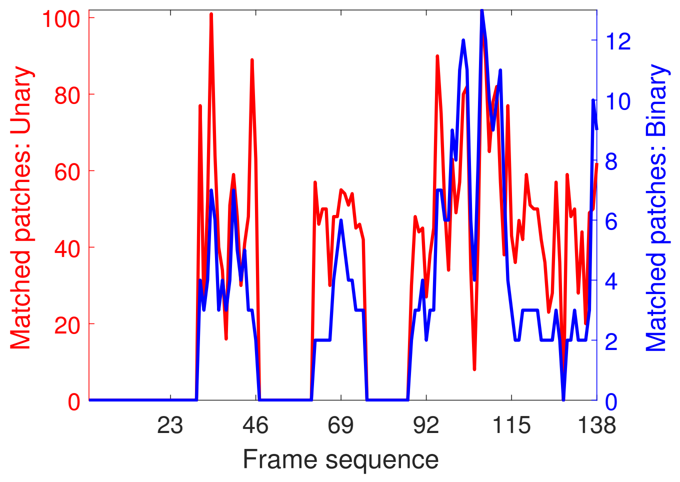

Fig. 4: The number of matched planes after unary and binary constraints.

We compare our method with SegMatch[1] and Scan Context[2] on a laptop with an Intel i7-7700 CPU at 2.80 GHz and 16 GB memory. Fig. 5 and Fig. 6 demonstrate the robustness of

[1]https://github.com/ethz-asl/segmap

[2]https://github.com/irapkaist/scancontext

TABLE I: The graph matching time with and without unary constraints.

with unary constraints	without unary constraints
0.259 ms	3.6 ms

these loop-closure detection methods. From Fig. 5 and Fig. 6, we can conclude that our method's robustness performance is better than Scan Context-10 [28] and equivalent to incremental version SegMatch [11] whose performace is better than that of the batch version, while slightly lower than Scan Context-50 [28].

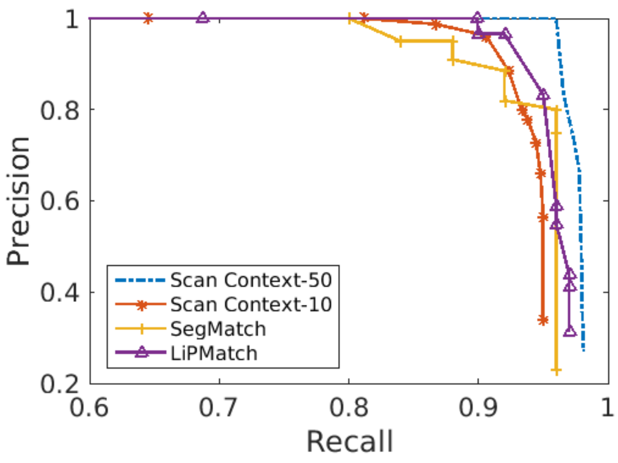

Fig. 5: The Precision/Recall curves of our method and the state-of-the-art methods in KITTI 00.

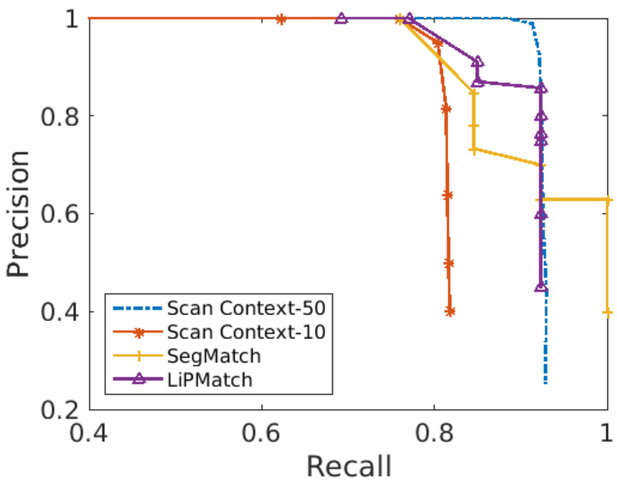

Fig. 6: The Precision/Recall curves of our method and the state-of-the-art methods in KITTI 05.

We evaluate the computational cost of each step of our method on two platforms: a laptop (Intel i7-7700 CPU at 2.80 GHz and 16 GB memory), and the Nvidia Jetson TX2 which is equipped with an ARM Cortex-A57 CPU. The average running time of our method while processing KITTI dataset is shown in TABLE II.

TABLE II: The time table of our system run on two platforms.

	plane extraction	graph matching	correction
Desktop PC	204 ms	0.259 ms	2080 ms
Jetson TX2	440 ms	0.471 ms	2640 ms

It is worth to mention that the execution time of correction module is affected by many factors including the point cloud registration method used, the size of local map, the point cloud density determined by voxel filtering and desired precision etc. For example, the correction module of Scan Context applies frame to frame point cloud registration, but the correction module of our method and SegMatch applies local map to local map point cloud registration. Therefore, for fair comparison, we do not include the time cost by this module. Then the overall execution time of Scan Context-50 including searching loop and calculating descriptors is 394.8 ms, which is greater than ours time 204.3 ms. The total time of Scan Context-10 is 211.4 ms, which is also greater than ours time 204.3 ms.

We divide the execution time of SegMatch into two parts of loop-closure detection and loop-closure correction. The loop-closure detection time of incremental version SegMatch including voxel filtering, normal estimation, segmentation, recognition and others [11] is 90.2 ms, which is indeed less than ours time 204.3 ms. The loop-closure correction time of SegMatch is 427.4 ms, which is also less than our loop-closure correction time 2080 ms. However, our method shows better robustness as shown in Fig. 5. It is worth to mention that there are not enough planes in one region on KITTI sequence 05 and our method can not detect loop-closure in this region. Therefore, compared with SegMatch, our method's Recall can not reach 1.0 on KITTI sequence 05. Nevertheless, the strategy developed in the incremental version SegMatch is quite inspiring. We will refer increment SegMatch to further improve our segmentation and plane extraction part in the future work.

Overall, our method's robustness performance is comparable to the state-of-the-art methods. Our method's efficiency performance is better than all methods mentioned above except incremental version SegMatch.

The result of applying our method on KITTI sequence 00 is illustrated in Fig. 7. For these two sequences, the vehicle trajectory is created using LOAM and we can tell there exists obvious pose drift. We can see that, in city scenarios, our method has correctly detected most of the loop-closures.

To further validate and evaluate our method, we implemented our algorithm for loop detection in data from [27]. Fig. 10 shows the results. Note that though we can observe two 'crossroads' in the middle of Fig. 10, they actually correspond to a bridge (the horizontal trajectory) goes over two roads (the two vertical trajectories). Therefore, there is no loop detected.

D. Loop-closure correction performance

We also show the performance of our loop-closure correction method in Fig. 9. We apply the fitness score as metric to evaluate the loop-closure correction performance of our

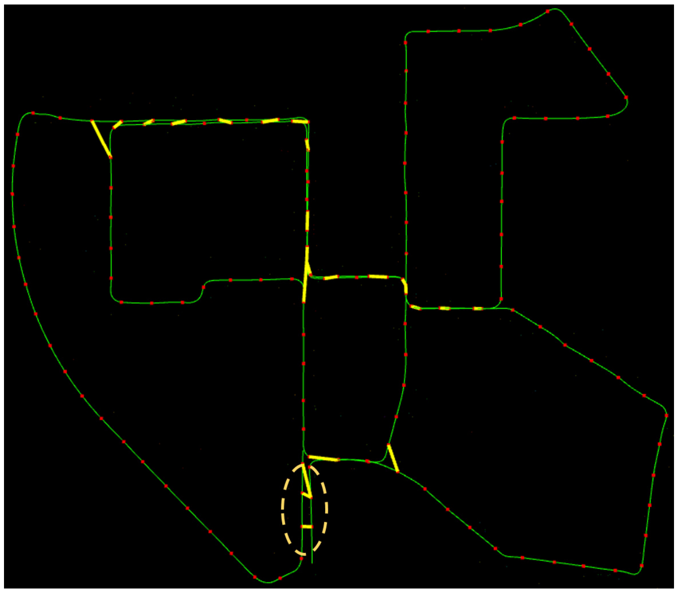
Fig. 7: Illustration of loop-closure detection with LiPMatch: the figures show loop-closures detected in real time during sequence 00 of the KITTI dataset. The trajectory is before pose graph optimization. The red dots represent locations where plane extraction and loop-closure detection were performed and the yellow lines indicate the detected loops.

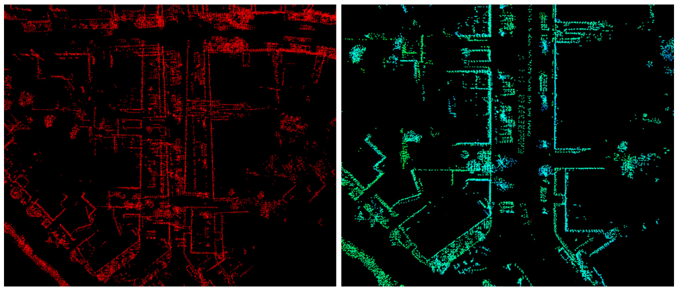
Fig. 8: The left figure shows the local map in dotted box in Fig. 7. before pose graph optimization and the right figure shows the map after pose graph optimization. From the right figure, we can find the streets and walls obviously, but not from the left figure.

method and LOAM. We can tell that our method obtain less fitness score, which demonstrates that our method can achieve less alignment errors compared with LOAM.

E. Discussion of viewpoint changes

The unary and binary constraints that are affected by viewpoint changes are given in (2), (6), and (7). We calculate the results of the related unary and binary constraints between all successfully matched planes. The average values of $\frac{a^c}{a^p}$, $\frac{\|\mathbf{c}_i^c-\mathbf{c}_j^c\|}{\|\mathbf{c}_{i'}^p-\mathbf{c}_{j'}^p\|}$, and $\frac{n_j^c\cdot(c_i^c-c_j^c)}{n_{j'}^p\cdot(c_{i'}^p-c_{j'}^p)}$ are respectively 1.16, 1.04, and 1.06. The parameter R is set to 1.3 at a minimum in our experiments which is greater than all the average values. Therefore, the setting of parameter R has taken account of potential occlusions caused by viewpoint changes. Besides, each keyframe represents a local map in our method, which implies that multiple frames are contained in the local map. As a result, the local map contains more planes compared with a single frame. This actually helps to alleviate the potential loop-close failures caused by occlusions.

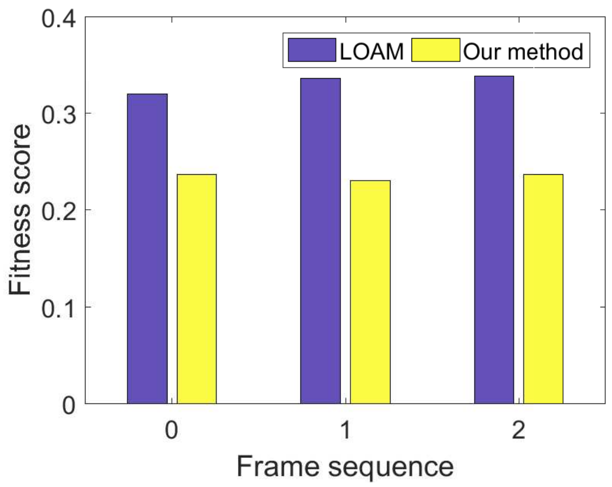

Fig. 9: The three frame sequences correspond to the three loop-closure detected in dotted box in Fig. 7. The blue bars represent the fitness score of LOAM among loop-closure correction and the yellow bars represent the fitness score of our method.

In some cases, two planes respectively from two local maps corresponding to the same object are not matched, which is caused by viewpoint changes. In this paper, we refer the case to 'missing matches'. In most city scenarios, local maps contain abundant planes. Thus, if it is not the case that all or most of the correct matched planes are missing, our method still works well. Because a few missing matches have no influence on the relative positions of other correct matched planes.

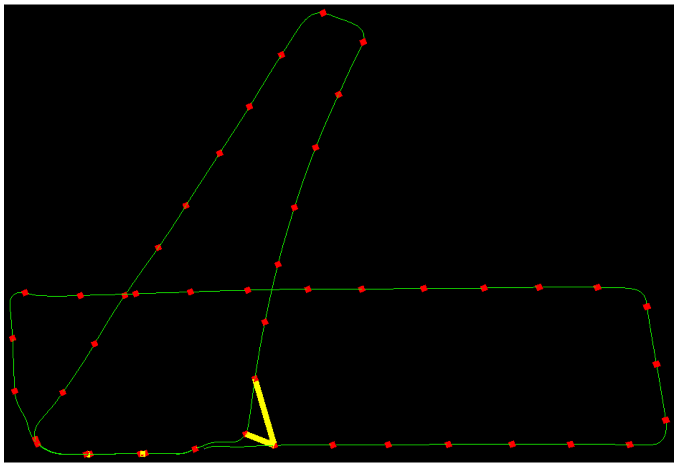
Fig. 10: Illustration of loop-closure detection with LiPMatch on data from [27].

In general, in the scenes with enough planes (most city scenarios), our method is capable of detecting loop closures. However, we also observe that viewpoint change could still lead to deficiency of our method. The reason is twofold. Firstly, it happens that some local maps contain few planes, and secondly, there exist massive planes that can not be completely contained in one local map, which means there might be only a single plane in the local map. Essentially, the performance of our method depends on the number and quality of features.

This makes binary constraints much difficult to be satisfied, hence leading to the deficiency of the loop-closure detection results. One solution is extracting some other segments (features) from the local map to ensure that even if some segments are affected by occlusions, the remaining segments can still help to detect the loop-closure. These 'features' really vary from scenario to scenario. We would not argue against the existence of a universal feature, but we are working towards incorporating other semantic features such as vehicles into our framework, to make it more robust in the future.

VI. Conclusion

We present a new 3D LiDAR point cloud plane based loop-closure detection and loop-closure correction methods. We build a plane graph for each keyframe and employ the geometric characteristics of the planes and their relative positions to detect loop-closures. We also exploit matched planes between keyframes to improve the ICP's robustness. When the loop-closures are successfully detected, we optimize the pose graph and get an optimized trajectory and map, to improve the SLAM accuracy.

We evaluate our method on the KITTI odometry benchmarks. We have demonstrated that our method can achieve comparable or better results when compared with the state-of-the-art methods in city scenarios. In the future, we will integrate non-structural, semantic information into our loop-closure detection to generalize our method applicable to more scenes.

Acknowledgment

This work is supported by the National Natural Science Foundation of China (Grant No. 91848111) and the Young Scientists Fund of the National Natural Science Foundation of China (Grant No. 61703387).

References

[1] J. Jiang, J. Wang, P. Wang, and Z. Chen, "Pou-slam: Scan-to-model matching based on 3d voxels," *Applied Sciences*, vol. 9, no. 19, pp. 4147 – 4155, 2019.

[2] P. Wang, Z. Chen, Q. Zhang, and J. Sun, "A loop closure improvement method of gmapping for low cost and resolution laser scanner," *IFAC-PapersOnLine*, vol. 49, no. 12, pp. 168–173, 2016.

[3] J. Wang, P. Wang, D. Dai, M. Xu, and Z. Chen, "Regression forest based rgb-d visual relocalization using coarse-to-fine strategy," *IEEE Robotics and Automation Letters*, vol. 5, no. 3, pp. 4431–4438, 2020.

[4] Q. Zhang, P. Wang, and Z. Chen, "An improved particle filter for mobile robot localization based on particle swarm optimization," *Expert Systems with Applications*, 2019.

[5] P. Wang, P. Xu, P. Bonnifait, and J. Jiang, "Box particle filtering for slam with bounded errors," in *2018 15th International Conference on Control, Automation, Robotics and Vision (ICARCV)*. IEEE, 2018, pp. 1032–1038.

[6] X. Chen, A. Milioto, E. Palazzolo, P. Gigure, J. Behley, and C. Stachniss, "Semantically assisted loop closure in slam using ndt histograms," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2019, pp. 1–8.

[7] J. Wang, P. Wang, and Z. Chen, "A novel qualitative motion model based probabilistic indoor global localization method," *Information Sciences*, vol. 429, pp. 284–295, 2018.

[8] M. Bosse and R. Zlot, "Place recognition using keypoint voting in large 3d lidar datasets," in *2013 IEEE International Conference on Robotics and Automation*. IEEE, 2013, pp. 2677–2684.

[9] M. Magnusson, H. Andreasson, A. Nüchter, and A. J. Lilienthal, "Automatic appearance-based loop detection from three-dimensional laser data using the normal distributions transform," *Journal of Field Robotics*, vol. 26, no. 11-12, pp. 892–914, 2009.

[10] R. Dubé, D. Dugas, E. Stumm, J. Nieto, R. Siegwart, and C. Cadena, "Segmatch: Segment based place recognition in 3d point clouds," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2017, pp. 5266–5272.

[11] R. Dubé, M. G. Gollub, H. Sommer, I. Gilitschenski, R. Siegwart, C. Cadena, and J. Nieto, "Incremental segment-based localization in 3D point clouds," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 1832–1839, 2018.

[12] R. Dubé, A. Cramariuc, D. Dugas, J. Nieto, R. Siegwart, and C. Cadena, "Segmap: 3d segment mapping using data-driven descriptors," *arXiv preprint arXiv:1804.09557*, 2018.

[13] R. Dubé, A. Gawel, H. Sommer, J. Nieto, R. Siegwart, and C. Cadena, "An online multi-robot slam system for 3d lidars," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017, pp. 1004–1011.

[14] H. Yin, L. Tang, X. Ding, Y. Wang, and R. Xiong, "Locnet: Global localization in 3d point clouds for mobile vehicles," in *2018 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2018, pp. 728–733.

[15] E. Fernández-Moral, W. Mayol-Cuevas, V. Arévalo, and J. Gonzalez-Jimenez, "Fast place recognition with plane-based maps," in *2013 IEEE International Conference on Robotics and Automation*. IEEE, 2013, pp. 2719–2724.

[16] G. C. Sharp, W. L. Sang, and D. K. Wehe, "Icp registration using invariant features," *IEEE Transactions on Pattern Analysis Machine Intelligence*, vol. 24, no. 1, pp. 90–102, 2002.

[17] Y. Chen and G. G. Medioni, "Object modeling by registration of multiple range images." *Image Vision Comput.*, vol. 10, no. 3, pp. 145–155, 1992.

[18] A. Censi, "An icp variant using a point-to-line metric," in *2008 IEEE International Conference on Robotics and Automation*. Ieee, 2008, pp. 19–25.

[19] A. Segal, D. Haehnel, and S. Thrun, "Generalized-icp." in *Robotics: science and systems*, vol. 2, no. 4. Seattle, WA, 2009, p. 435.

[20] Y. Pan, B. Yang, F. Liang, and Z. Dong, "Iterative global similarity points: A robust coarse-to-fine integration solution for pairwise 3d point cloud registration," in *2018 International Conference on 3D Vision (3DV)*. IEEE, 2018, pp. 180–189.

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

[22] T. Chen, B. Dai, R. Wang, and D. Liu, "Gaussian-process-based real-time ground segmentation for autonomous land vehicles," *Journal of Intelligent Robotic Systems*, vol. 76, no. 3-4, pp. 563–582.

[23] A. Day, "Planar convex hull algorithms in theory and practice," in *Computer graphics forum*, vol. 7, no. 3. Wiley Online Library, 1988, pp. 177–193.

[24] R. Hartley and A. Zisserman, *Multiple view geometry in computer vision*. Cambridge university press, 2003.

[25] G. Grisetti, R. Kummerle, C. Stachniss, and W. Burgard, "A tutorial on graph-based slam," *IEEE Intelligent Transportation Systems Magazine*, vol. 2, no. 4, pp. 31–43, 2010.

[26] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," in *2012 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2012, pp. 3354–3361.

[27] F. Moosmann and C. Stiller, "Velodyne slam," in *2011 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2011, pp. 393–398.

[28] G. Kim and A. Kim, "Scan context: Egocentric spatial descriptor for place recognition within 3d point cloud map," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 4802–4809.