

A fast, complete, point cloud based loop closure for LiDAR odometry and mapping

Jiarong Lin and Fu Zhang

Abstract—This paper presents a loop closure method to correct the long-term drift in LiDAR odometry and mapping (LOAM). Our proposed method computes the 2D histogram of keyframes, a local map patch, and uses the normalized cross-correlation of the 2D histograms as the similarity metric between the current keyframe and those in the map. We show that this method is fast, invariant to rotation, and produces reliable and accurate loop detection. The proposed method is implemented with careful engineering and integrated into the LOAM algorithm, forming a complete and practical system ready to use. To benefit the community by serving a benchmark for loop closure, the entire system is made open source on Github¹.

I. INTRODUCTION

With the capacity of estimating the 6 degrees of freedom (DOF) state, and meanwhile building the high precision maps of surrounding environments, SLAM methods using LiDAR sensors have been regarded as an accurate and reliable way for robotic perception. In the past years, LiDAR odometry and mapping (LOAM) have been successfully applied in the field of robotics, like self-driving car [1], autonomous drone [2, 3], field robots surveying and mapping [4, 5], etc. In this paper, we focus on the problem of developing a fast and complete loop closure system for laser-based SLAM systems.

Loop closure is an essential part of SLAM system, to estimate the long term accumulating drift caused by local feature matching. In a common paradigm of loop closure, the successful detection of loops plays an essential role. Loop detection is the ability of recognizing the previously visited places, by giving a measurement of similarity between any two places. For visual-SLAM methods, the loop closure considerably benefits from various largely available computer vision algorithms. For example, by utilizing the bag-of-words model [6, 7] and clustering the feature descriptors as words, the similarity between observations can be computed in the word space. This kind of method has been used in most of the state of the art visual SLAM system (e.g. [8, 9]) and have achieved great success in the past years.

Unlike the visual-SLAM, the relevant research of laser-based loop closure is rare, and it is surprisingly hard for us to find any available open soured solution which addresses this problem. We conclude these phenomenons as two main reasons: Firstly, compared to the camera sensors, the cost of LiDAR sensors are extremely expensive, preventing them in wider use. In most of the robotics navigation perception,

J. Lin and F. Zhang are with the Department of Mechanical Engineering, Hong Kong University, Hong Kong SAR., China. {jiarong.lin, fuzhang}@hku.hk

¹https://github.com/hku-mars/loam_livox

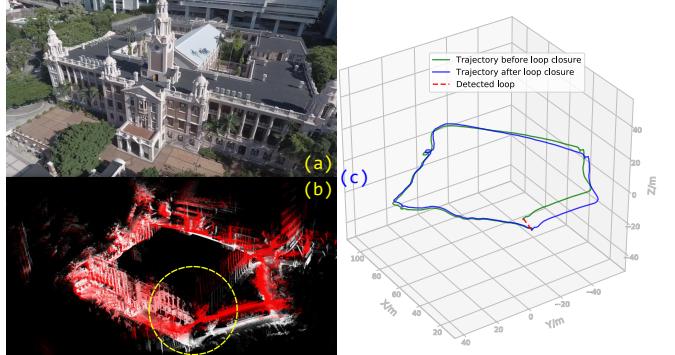


Fig. 1: An example of loop closure around the Main Building of the Hong Kong University (HKU). (a), the RGB image of the map area; (b), the red and white points are off the map before and after loop closure, respectively; (c), the red dashed line indicates the detected loop, the green, and blue solid lines are the trajectories before and after loop closure, respectively.

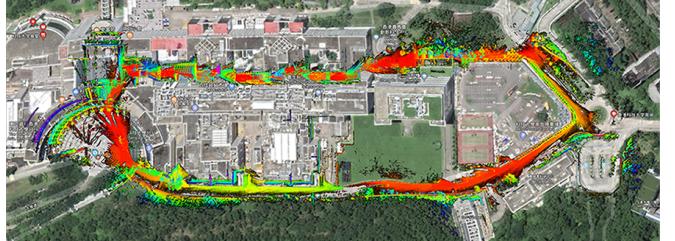


Fig. 2: The large scale loop closure of the Hong Kong University of Science and Technology (HKUST) campus. We align the point cloud map after loop closure with the satellite image. Our video is available at https://youtu.be/fOSTJ_yLhFM.

LiDARs is not always the first choice. Secondly, the problem of place recognition on point cloud is very challenging. Unlike 2D images containing rich information such as textures and colors, the available informations in point cloud are only the geometry shapes in 3D space.

In this paper, we develop a fast, complete loop closure system for LiDAR odometry and mapping (LOAM), consisting of fast loop detection, maps alignment, and pose graph optimization. We integrate the proposed loop closure method into a LOAM algorithm with Livox MID40² sensor, a high performance low cost LiDAR sensor easily available. Some of the results we obtain are shown in Fig. 1 and Fig. 2. To contribute to the development of laser-based slam methods, we will open source all the datasets and codes on Github¹.

II. RELATED WORK

Loop closure is widely found in visual-SLAM to correct its long-term drift. The commonly used pipeline mainly consists of three steps: First, local feature of a 2D images

²<https://www.livoxtech.com/mid-40-and-mid-100>

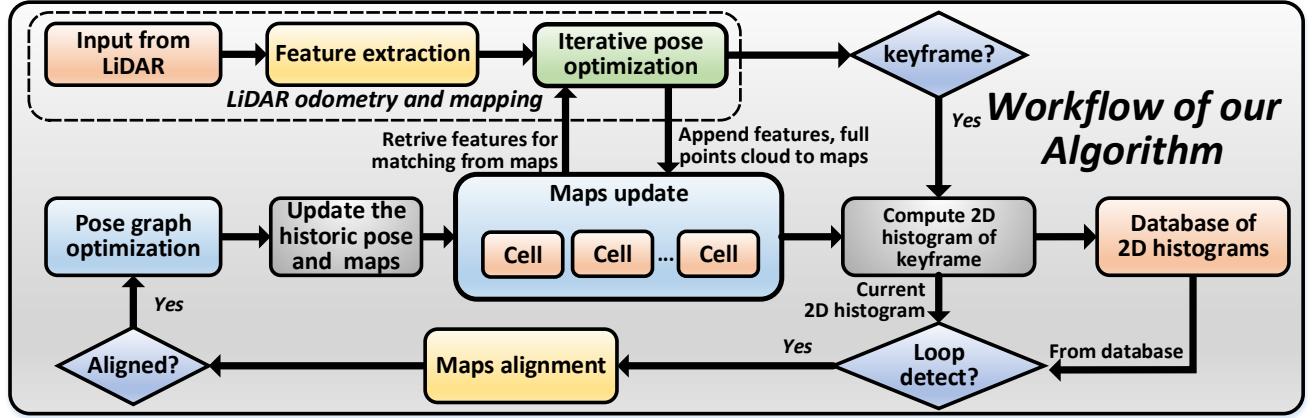


Fig. 3: The overview of our system.

is extracted by using handcrafted descriptors such as Scale-Invariant Feature Transforms (SIFT) [10], Binary Robust Independent Elementary Features (BRIEF) [11], Oriented Fast and Rotated BRIEF(ORB) [12], etc. Then, a bag-of-world model [6, 7] is used to cluster these features and build a dictionary to achieve loop detection. Finally, with the detected loop, a pose graph optimization is formulated and solved update the historic poses in the map.

Unlike the visual-SLAM, loop detection for point cloud data is still an open, challenging problem in laser-based SLAM. Bosse *et al* [13] achieve place recognition by directly extracting keypoints from the 3D point cloud and describe them with a handcrafted *3D Gestalt* descriptors. Then the keypoints vote for their nearest points and the scores are used to determine if a loop is detected. The similar method is also used in [14]. Magnusson *et al* [15] describe the appearance of 3D point cloud by utilizing the normal distribution transform (NDT), they compute the similarity of two scans from the histogram of the NDT descriptors.

Besides the hand-crafted descriptors, learning-based method has also been used in loop detection(or place recognitions) in recent years. For example, the SegMatch proposed by Dube *et al* [16] achieves place recognition by matching semantic features like buildings, tree, vehicles, etc. Angelina *et al* [17] realize place recognition by extracting the global descriptor from an end-to-end network, which is trained by combining the PointNet [18] and NetVLAD [19]. The learning-based method is usually computationally expensive, and the performances greatly rely on the dataset in the training process.

Although with these reviewed work, to our best knowledge, there is no open-sourced codes or dataset that benchmark the problem of loop closure for LOAM, which leaves some difficulties for readers on reproducing their works. To this purpose, we propose a complete loop closure system. The loop detection of our work is mainly inspired by the method of [15] and some of the adjustments are made in our scenarios. Due to the small FoV and special scanning pattern of our considered LiDAR sensor, we perform loop detection for an accumulated time of scans (i.e., the *keyframe*). To summarize, our contributions are threefold: (1) we develop

a fast loop detection method to quickly measure the similarity of two keyframes; (2) we integrate our loop closure system, consisting of the loop detection, map alignment, and pose optimization into an LiDAR odometry and mapping algorithm (LOAM) [20], leading to a complete and practical system ready to use; (3) we provide an available solution and paradigm for point cloud based loop closure by opening source our systems and datasets on Github.

III. SYSTEM OVERVIEW

The workflow of our system is shown in Fig. 3, each new frame (i.e., scan) input from LiDAR is registered to the global map by LOAM algorithm [20]. If a specified number of frames have been received (e.g., 100 frames), a keyframe created, which forms a small local map patch. The raw points, which were registered to the cells of the global map (Section IV) by LOAM, corresponding to the new keyframe are retrieved to compute its 2D histogram (see Section V). The computed 2D histogram is compared to the database, which contains 2D histograms of the global map consisting of all the past keyframes, to detect a possible loop (see Section VI). Meanwhile, the new keyframe 2D histogram is added to the database for the use of next keyframe. Once a loop is detected, the keyframe is aligned with global map and a pose graph optimization is performed to correct the drift in the global map.

IV. MAP AND CELL

In this section, we will introduce two key elements of our algorithm, the maps and cell. For conveniently, We use \mathcal{M} and \mathcal{C} denote the map and cell, respectively.

A. Cell

A cell is a small cube of a fixed size (i.e., S_x, S_y and S_z in x, y and z directions) by partitioning the 3D space. It is represented by its center location \mathcal{C}_c and created by the first point $\mathbf{P}_i = [\mathbf{P}_{i_x}, \mathbf{P}_{i_y}, \mathbf{P}_{i_z}]^T$ in it

$$\mathcal{C}_c = \begin{bmatrix} \text{round}(\mathbf{P}_{i_x}/S_x) * S_x + S_x/2 \\ \text{round}(\mathbf{P}_{i_y}/S_y) * S_y + S_y/2 \\ \text{round}(\mathbf{P}_{i_z}/S_z) * S_z + S_z/2 \end{bmatrix} \quad (1)$$

Let N denote the number of points located in a cell \mathcal{C}_c , the mean \mathcal{C}_μ and covariance \mathcal{C}_Σ of this cell is:

$$\mathcal{C}_\mu = \frac{1}{N} \left(\sum_{i=1}^N \mathbf{P}_i \right) \quad (2)$$

$$\mathcal{C}_\Sigma = \frac{1}{N-1} \left(\sum_{i=1}^N (\mathbf{P}_i - \mathcal{C}_\mu)(\mathbf{P}_i - \mathcal{C}_\mu)^T \right) \quad (3)$$

Notice that the cell is a fixed partitioning of the 3D space are is constantly populated with new points. To speed up the computation of mean 2 and covariance 3, we derive its recursive form as follows. Denote \mathbf{P}_{N+1} the new point, N is the number of existing points in a cell with mean \mathcal{C}'_μ and covariance \mathcal{C}'_Σ . The mean \mathcal{C}_μ and covariance \mathcal{C}_Σ of all the $N+1$ points in the cell are:

$$\mathcal{C}_\mu = \frac{1}{N+1} (N\mathcal{C}'_\mu + \mathbf{P}_{N+1}) \quad (4)$$

$$\begin{aligned} \mathcal{C}_\Sigma &= \frac{1}{N} \sum_{i=1}^{N+1} (\mathbf{P}_i - \mathcal{C}_\mu)(\mathbf{P}_i - \mathcal{C}_\mu)^T \\ &= \frac{1}{N} \sum_{i=1}^{N+1} (\mathbf{P}_i - \mathcal{C}'_\mu + \mathcal{C}'_\mu - \mathcal{C}_\mu)(\mathbf{P}_i - \mathcal{C}'_\mu + \mathcal{C}'_\mu - \mathcal{C}_\mu)^T \\ &= \frac{1}{N} \left[(N-1)\mathcal{C}'_\Sigma + (\mathbf{P}_{N+1} - \mathcal{C}'_\mu)(\mathbf{P}_{N+1} - \mathcal{C}'_\mu)^T \right. \\ &\quad \left. + (N+1)(\mathcal{C}'_\mu - \mathcal{C}_\mu)(\mathcal{C}'_\mu - \mathcal{C}_\mu)^T \right. \\ &\quad \left. + 2(\mathcal{C}'_\mu - \mathcal{C}_\mu)(\mathbf{P}_{N+1} - \mathcal{C}'_\mu)^T \right] \end{aligned} \quad (5)$$

Therefore, a cell \mathcal{C} is composed of its static center \mathcal{C}_c , the dynamically updated mean \mathcal{C}_μ and covariance \mathcal{C}_Σ , and the raw points collection $\{\mathbf{P}_i\}$: $\mathcal{C} = (\mathcal{C}_c, \mathcal{C}_\mu, \mathcal{C}_\Sigma, \{\mathbf{P}_i\})$.

B. Map

The map \mathcal{M} is the collection of all raw points saved in cells. More specifically, \mathcal{M} consists of a hash table \mathcal{H} and a global octree \mathcal{O} . The hash table \mathcal{H} enables to quickly find the specific cell according to its center \mathcal{C}_c . The octree \mathcal{O} enables to find out all cells located in the specific area of given range. These two are of significant importance in speeding up the maps alignments.

For any new added cell \mathcal{C} , we compute its hash index $\mathcal{H}(\mathcal{C}_c)$ using the XOR operation of hash index of its individual components: $(\mathcal{C}_{c_x}, \mathcal{C}_{c_y}, \mathcal{C}_{c_z})$. The computed hash index is then added to the hash table of the map \mathcal{H} . Since the cell is a fixed partitioning of the 3D space, its center location \mathcal{C}_c is static, requiring no update for existing entries in the hash table (the hash table is dynamically growing though).

The new added cell \mathcal{C} is also added to the Octree \mathcal{O} according to its center location, similar to the OctoMap in [21]. Algorithm 1 illustrates the procedure of incrementally creating cells and maps from new frames.

Algorithm 1: Registration of new frame

```

Input : Points  $\mathcal{P}_k$  from  $k$ -th frame, Current map  $\mathcal{M}$ ,  

         the pose  $(\mathbf{R}_k, \mathbf{T}_k)$  estimated from LOAM  

         algorithm
for each  $\mathbf{P}_l \in \mathcal{P}_k$  do  

    Transform  $\mathbf{P}_l$  to global frame by  $\mathbf{P}_i = \mathbf{R}_k \mathbf{P}_l + \mathbf{T}_k$ .  

    Compute the cell center  $\mathcal{C}_c$  from (1).  

    Compute the hash index  $\mathcal{H}(\mathcal{C}_c)$ .  

    if  $\mathcal{H}(\mathcal{C}_c) \notin \mathcal{H}$  then  

        Create new cell  $\mathcal{C}$  with center  $\mathcal{C}_c$ .  

        Insert  $\mathcal{H}(\mathcal{C}_c)$  to hash table  $\mathcal{H}$  of map  $\mathcal{M}$ .  

        Insert  $\mathcal{C}_c$  to Octree  $\mathcal{O}$  of map  $\mathcal{M}$ .  

    Add  $\mathbf{P}_i$  to  $\mathcal{C}$ .  

    Update mean  $\mathcal{C}_c$  of  $\mathcal{C}$  using (4).  

    Update covariance  $\mathcal{C}_\Sigma$  of  $\mathcal{C}$  using (5).

```

V. 2D HISTOGRAM OF ROTATION INVARIANCE

The main idea of our fast loop detection is that we use the 2D image-like histograms to roughly describe the keyframe. The 2D histogram describes the distribution of the Euler-angles of the feature direction in a keyframe.

A. The feature type and direction in a cell

As mentioned previously, each keyframe consists of a number of (e.g., 100) frames and each frame (i.e., scan) is partitioned into cells. For each cell, we determine the shape formed by its points and the associated feature direction (denoted as \mathcal{C}_d). Similar to [15], we perform eigenvalue decomposition on the covariance matrix \mathcal{C}_Σ in (3):

$$\mathcal{C}_\Sigma \mathbf{V} = \mathbf{V} \Lambda \quad (6)$$

where Λ is diagonal matrix with eigenvalues in descending order (i.e., $\lambda_1 \geq \lambda_2 \geq \lambda_3$). In practice, we only consider cells with 5 or more points to increase the robustness.

- **Cell of plane shape:** If λ_2 is significantly larger than λ_3 , we regard this cell as a plane shape and regard the plane normal as the feature direction, i.e., $\mathcal{C}_d = \mathbf{V}_3$ where \mathbf{V}_3 is the third column of the matrix \mathbf{V} .
- **Cell of line shape:** If the cell is not a plane and λ_1 is significantly larger than λ_2 , we regard this cell as a line shape and regard the line direction as the feature direction, i.e., $\mathcal{C}_d = \mathbf{V}_1$, the first column of \mathbf{V} .
- **Cell with no feature:** A cell which is neither a line nor plane shape is not considered.

B. Rotation invariance

In order to make our feature descriptors invariant to arbitrary rotation of the keyframe, we rotate each feature direction \mathcal{C}_d by multiplying it to an additional rotation matrix \mathbf{R} , and expect that most of the feature direction are lie on X -axis, and the secondary most are on Y -axis. Since plane feature is more reliable than line feature (e.g., the edge of plane feature are treated as a line feature), we use the feature

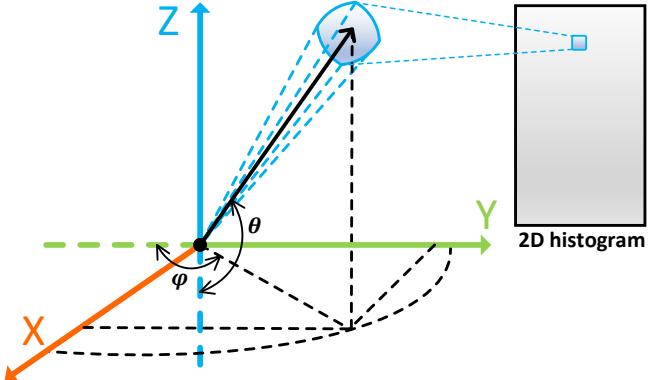


Fig. 4: The Euler angle of a feature direction and its contribution to the 2D histogram, each element of the 2D histogram is the number of feature directions with pitch θ and yaw ϕ located in the corresponding bin.

direction of plane cells to determine the rotation matrix \mathbf{R} . Similar to the previous sections, we compute the covariance Σ_d of all plane feature directions in a keyframe:

$$\Sigma_d = \sum_{i=1}^N \mathcal{C}_{i_d} \mathcal{C}_{i_d}^T \quad (7)$$

where N is the number of plane cells, \mathcal{C}_{i_d} denotes the feature direction (i.e., plane normal) of the i -th plane cell. Similarly, the eigenvalue decomposition of Σ_d is:

$$\Sigma_d \mathbf{V}_d = \mathbf{V}_d \Lambda_d \quad (8)$$

where Λ_d is a diagonal matrix with eigenvalues in descending order ($\lambda_{d_1} \geq \lambda_{d_2} \geq \lambda_{d_3}$), $\mathbf{V}_d = [\mathbf{V}_{d_1} \ \mathbf{V}_{d_2} \ \mathbf{V}_{d_3}]$ is the eigenvector matrix. Then, the rotation matrix \mathbf{R} is determined as:

$$\mathbf{R} = [\mathbf{V}_{d_1} \ \mathbf{V}_{d_2} \ \mathbf{V}_{d_1} \times \mathbf{V}_{d_2}]^T \quad (9)$$

After compute the rotation matrix \mathbf{R} , we apply the rotation transformation to all feature (both plane and line) directions.

Algorithm 2: Computing the 2D hist. of a keyframe

```

Input : Current keyframe  $\mathcal{F}$ 
Output: 2D Histogram  $\mathbf{H}_L$  of line cell
          2D Histogram  $\mathbf{H}_P$  of plane cell
Start :  $\mathbf{H}_L \leftarrow 0$ ,  $\mathbf{H}_P \leftarrow 0$ .
          Compute rotation matrix  $\mathbf{R}$  from Sect. V-B.
for each  $\mathcal{C} \in \mathcal{F}$  do
    if  $\mathcal{C}$  is a line shape then
         $\mathcal{C}_d \leftarrow \mathbf{R}\mathcal{C}_d$ 
        Compute the pitch  $\theta$  and yaw  $\phi$  angle of  $\mathcal{C}_d$ .
         $\mathbf{H}_L[\text{round}(\phi/3^\circ), \text{round}(\theta/3^\circ)] += 1$ 
    if  $\mathcal{C}$  is a plane shape then
         $\mathcal{C}_d \leftarrow \mathbf{R}\mathcal{C}_d$ 
        Compute the pitch  $\theta$  and yaw  $\phi$  angle of  $\mathcal{C}_d$ .
         $\mathbf{H}_P[\text{floor}(\phi/3^\circ), \text{floor}(\theta/3^\circ)] += 1$ 
Gaussian blur  $\mathbf{H}_P$  and  $\mathbf{H}_L$ .
```

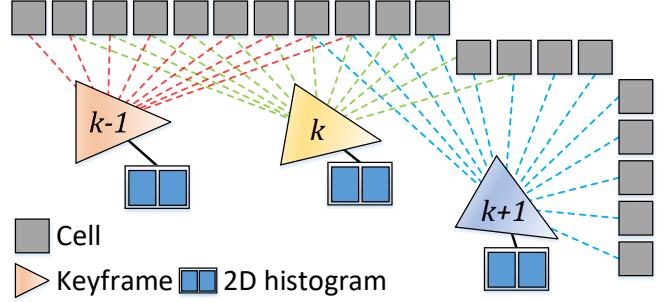


Fig. 5: A keyframe consists of n (e.g., $n = 100$) frames (not shown), which then contains many cells. Each keyframe has two 2D histograms, one for line cells and the other one for plane cells.

C. 2D histogram of keyframe

With the rotation invariant feature directions of all cells in a keyframe, we compute the 2D histogram as follows:

Firstly, for a given feature direction $\mathcal{C}_d = [\mathcal{C}_{d_x}, \mathcal{C}_{d_y}, \mathcal{C}_{d_z}]$, we choose the direction with positive X components, i.e., $\mathcal{C}_d = \text{sign}(\mathcal{C}_{d_x}) \cdot \mathcal{C}_d$. Then, the Euler angle of the feature direction is computed (see Fig. 4):

$$\theta = \sin^{-1}(\mathcal{C}_{d_z}) + 90^\circ \in [0^\circ, 180^\circ] \quad (10)$$

$$\phi = \tan^{-1}(\mathcal{C}_{d_y}/\mathcal{C}_{d_x}) + 90^\circ \in [0^\circ, 180^\circ] \quad (11)$$

The 2D-histogram we use is a 60×60 matrix (have 3° resolution on both pitch and yaw angle), the elements of this matrix denote the number of line/plane cell with its pitch θ and yaw ϕ located in the corresponding bin. For example, i -th row, j -th column element, e_{ij} , is the number of cells with the angle of its feature direction satisfied:

$$j \times 3^\circ \leq \theta < (j+1) \times 3^\circ \\ i \times 3^\circ \leq \phi < (i+1) \times 3^\circ$$

To increase the robustness of the 2D histogram to possible noise, we apply a Gaussian blur on each 2D histogram we computed.

The complete algorithm of computing the 2D histogram with rotation invariance is shown in Algorithm. 2.

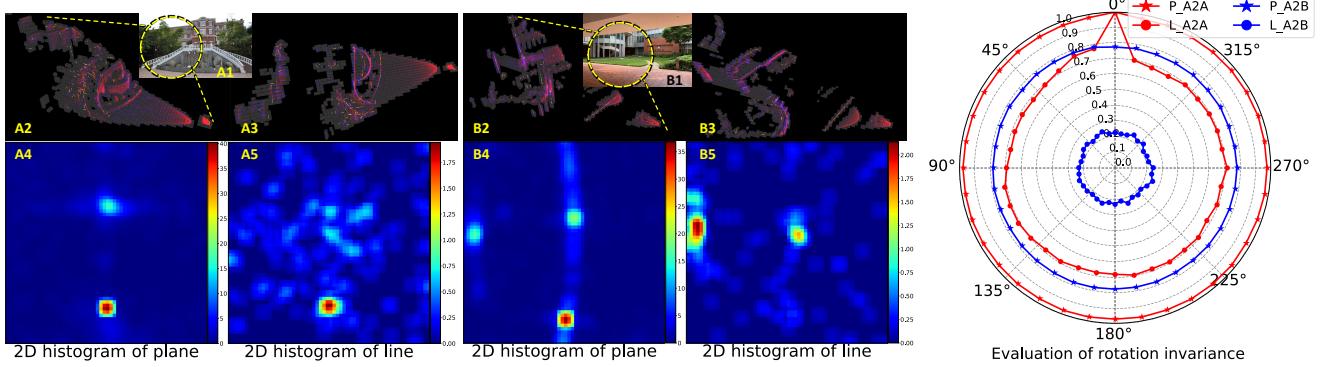
VI. FAST LOOP DETECTION

A. Procedure of loop detection

As mentioned previously, we group n frames (e.g., $n = 100$) into a keyframe \mathcal{F} . It can be viewed as a local patch of the global map \mathcal{M} , and contains all of the cells appearing in the last n frames, as shown in Fig. 5. We compute the 2D histogram of a new keyframe \mathcal{F} and its similarity (Section VI. B) with all keyframes in the past to detect a loop. The keyframe with a detected loop is then matched to the map (Section VI. C) and the map is updated with a pose graph optimization (Section VI. D).

B. Similarity of two keyframes

For each newly added keyframe, we measure its similarity to all history keyframes. In this work, we use the normalized cross-correlation of 2D histograms to compute their similarity, which has been widely used in the field of computer vision (e.g., template matching, image tracking,



(a) The visualization of two keyframes (frame A and B), 2D histograms, and contained cells. Fig. A1 is the RGB image of the first scene. Fig. A2 and Fig. A3 are the side-view and bird view of the keyframe, frame A and A ("P_A2A"), and between respectively. In Fig. A2 and Fig. A3, the red points denote the raw point cloud in the keyframe, the frame A and B ("P_A2B"), and in line feature white cubes denote the cells, the blue lines are the feature direction of plane cells, and the yellow lines are the feature direction of line cells. Fig. A4 and Fig. A5 are the 2D histogram (pixels are colored by their values) of plane and line features, respectively. The arrangement of Fig. B1~B5 is the same as distance is the similarity level while polar angle is the magnitude of random rotations.

Fig. 6: The visualization of keyframe, cells, and 2D histograms (a) and the evaluation of rotation invariance (b).

etc.). The similarity $S(\mathbf{H}_1, \mathbf{H}_2)$ of two 2D histogram $\mathbf{H}_1, \mathbf{H}_2$ is computed as:

$$S(\mathbf{H}_1, \mathbf{H}_2) = \frac{\sum_I (\mathbf{H}_1(I) - \bar{\mathbf{H}}_1)(\mathbf{H}_2(I) - \bar{\mathbf{H}}_2)}{\sqrt{\sum_I (\mathbf{H}_1(I) - \bar{\mathbf{H}}_1)^2 \sum_I (\mathbf{H}_2(I) - \bar{\mathbf{H}}_2)^2}} \quad (12)$$

where $\bar{\mathbf{H}}_k = \frac{1}{N} \sum_I \mathbf{H}_k(I)$ is the mean of \mathbf{H}_k and $I = (i, j)$ is the index of the element in \mathbf{H}_k . If the similarity $S(\mathbf{H}_1, \mathbf{H}_2)$ between two keyframes is higher than threshold (e.g., 0.90 for plane and 0.65 for line), a loop is thought to be detected.

C. Maps alignment

After the successful detecting of a loop, we perform maps alignment to compute the relative pose between two keyframes. The problem of maps alignment can be viewed as the registration between the target point cloud and source point cloud, as their work of [22].

Since we have classified the cell of linear shape and planar shape in our LOAM algorithm [20], we use the feature of edge-to-edge and planar-to-planar to iteratively solve the relative pose.

After the alignment, if the average distance of the points of edge/plane feature on is close enough to the edge/plane feature (distance less than 0.1m), we regard these two maps are aligned.

D. Pose graph optimization

As the workflow is shown in Fig. 3, once the two keyframes are aligned, we perform the pose graph optimization following the method in [23, 24]. We implement the graph optimization using the Google *ceres-solver*³. After optimizing the pose graph, we update all the cells in the entire map by recomputing the contained points, the points mean and covariance.

VII. RESULTS

A. Visualization of keyframe, cells, and 2D histograms

We visualize the two keyframes, their associated 2D histograms local maps, and contained cells in Fig. 6(a). This figure shows that the 2D histogram of the two different scenes are very distinctive.

B. Rotation invariance

We evaluate the rotation invariance of our loop detection method by computing the similarity of the two scenes in Fig. 6(a) with random rotations. For each level of rotation, we generate 50 random rotation matrix of random directions but the same magnitude, rotate one of the two scenes by the generated rotation matrix, and compute the average similarity among all the 50 rotations of the same magnitude. The similarity of keyframe A to itself, keyframe B to itself, and keyframe A to keyframe B are shown as Fig. 6(b). It can be seen that, the similarity of plane features almost hold the same under different or rotation magnitude, and the similarity of the same keyframe (with arbitrary rotation) is constantly higher than the similarity of different keyframes. For line features, although the similarity of the same keyframe slightly drops when rotation takes place, it is still significantly higher than the similarity of different keyframes.

C. Time of computation

We evaluate the time consumption of each step of our system on two platforms: the desktop PC (with i7-9700K) and onboard-computer (DJI manifold2⁴ with i7-8550U). The average running time of our algorithm run on *HKUST large scale dataset* (the first column of Fig. 7) are shown in Table. I, where we can see our proposed method is fast and suitable for real time scenarios on both platforms.

³<http://ceres-solver.org/>

⁴<https://www.dji.com/cn/manifold-2>

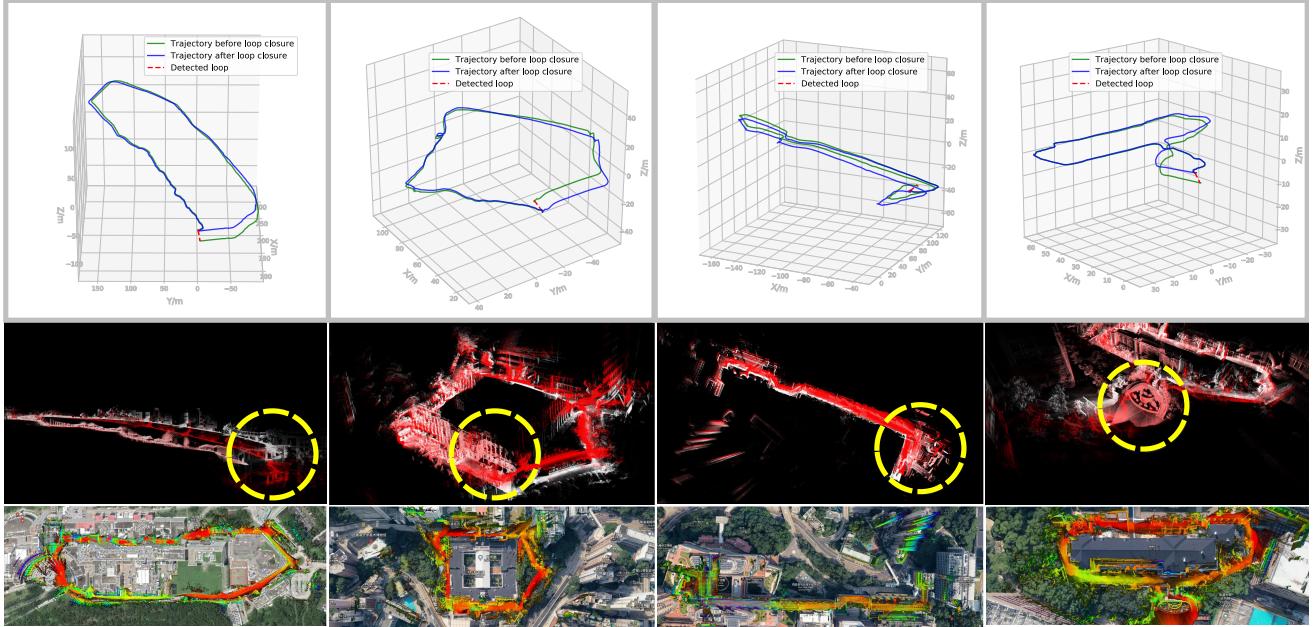


Fig. 7: We test our algorithm on four datasets, which are all sampled by Livox-MID40. The first one is a large scale dataset sampled in HKUST campus; The second one is sampled around a square building (main building of HKU); The third one is sampled indoor consisting of two long corridors in two neighboring floors. The fourth one is sampled around a rectangular building (Chong Yuet Ming Cultural Center in HKU) with many natural objects, such as trees, stairs, sculptures, etc.

	2D histogram computing	Maps alignment	Similarity of two maps
Desktop PC	1.18 ms	621 ms	13 μ s
Onboard-computer	1.48 ms	931 ms	16 μ s

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

D. Large scale loop closure results

We test our algorithm on four datasets in Fig. 7, where the first row is the comparison of trajectory before (green solid line) and after (blue solid line) loop closure, the red dashed lines indicate the detected loop. The second row of figures is the comparison of the point cloud map before (red) and after loop closure (white), where we can see the loop closure can effectively reduce the drift of LiDAR odometry and mapping (especially in the areas inside yellow circle). We align our point cloud after loop closure with Goolge maps in the third row, where we can see the alignment is very accurate, showing that the accuracy of our system is of high precision.

VIII. CONCLUSION

This paper presented a fast, complete, point cloud based loop closure for LiDAR odometry and mapping, we develop a loop detection method which can quickly evaluate the similarity of two keyframes from the 2D histogram of plane and line features in the keyframe. We test our system on four datasets and the results demonstrate that our loop closure can significantly reduce the long-term drift error. We open sourced all of our datasets and codes on Github to serve as an available solution and paradigm for point cloud based loop closure research in the future.

REFERENCES

- [1] J. Levinson, J. Askeland, J. Becker, J. Dolson, D. Held, S. Kammel, J. Z. Kolter, D. Langer, O. Pink, V. Pratt, *et al.*, “Towards fully au-

- tonomous driving: Systems and algorithms,” in *2011 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2011, pp. 163–168.
- [2] A. Bry, A. Bachrach, and N. Roy, “State estimation for aggressive flight in gps-denied environments using onboard sensing,” in *2012 IEEE International Conference on Robotics and Automation*. IEEE, 2012, pp. 1–8.
- [3] F. Gao, W. Wu, W. Gao, and S. Shen, “Flying on point clouds: Online trajectory generation and autonomous navigation for quadrotors in cluttered environments,” *Journal of Field Robotics*, vol. 36, no. 4, pp. 710–733, 2019.
- [4] A. Nüchter, K. Lingemann, J. Hertzberg, and H. Surmann, “6d slam3d mapping outdoor environments,” *Journal of Field Robotics*, vol. 24, no. 8-9, pp. 699–722, 2007.
- [5] B. Schwarz, “Lidar: Mapping the world in 3d,” *Nature Photonics*, vol. 4, no. 7, p. 429, 2010.
- [6] D. Gálvez-López and J. D. Tardos, “Bags of binary words for fast place recognition in image sequences,” *IEEE Transactions on Robotics*, vol. 28, no. 5, pp. 1188–1197, 2012.
- [7] D. Filliat, “A visual bag of words method for interactive qualitative localization and mapping,” in *Proceedings 2007 IEEE International Conference on Robotics and Automation*. IEEE, 2007, pp. 3921–3926.
- [8] R. Mur-Artal and J. D. Tardós, “Orb-slam2: An open-source slam system for monocular, stereo, and rgbd cameras,” *IEEE Transactions on Robotics*, vol. 33, no. 5, pp. 1255–1262, 2017.
- [9] T. Qin, P. Li, and S. Shen, “Vins-mono: A robust and versatile monocular visual-inertial state estimator,” *IEEE Transactions on Robotics*, vol. 34, no. 4, pp. 1004–1020, 2018.
- [10] D. G. Lowe *et al.*, “Object recognition from local scale-invariant features.” in *iccv*, vol. 99, no. 2, 1999, pp. 1150–1157.
- [11] M. Calonder, V. Lepetit, C. Strecha, and P. Fua, “Brief: Binary robust independent elementary features,” in *European conference on computer vision*. Springer, 2010, pp. 778–792.
- [12] E. Rublee, V. Rabaud, K. Konolige, and G. R. Bradski, “Orb: An efficient alternative to sift or surf.” in *ICCV*, vol. 11, no. 1. Citeseer, 2011, p. 2.
- [13] 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.
- [14] A. Gawel, T. Cieslewski, R. Dubé, M. Bosse, R. Siegwart, and J. Nieto, “Structure-based vision-laser matching,” in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2016, pp. 182–188.

- [15] M. Magnusson, H. Andreasson, A. Nuchter, and A. J. Lilienthal, “Appearance-based loop detection from 3d laser data using the normal distributions transform,” in *2009 IEEE International Conference on Robotics and Automation*. IEEE, 2009, pp. 23–28.
- [16] 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.
- [17] M. Angelina Uy and G. Hee Lee, “Pointnetvlad: Deep point cloud based retrieval for large-scale place recognition,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 4470–4479.
- [18] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, “Pointnet: Deep learning on point sets for 3d classification and segmentation,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 652–660.
- [19] R. Arandjelovic, P. Gronat, A. Torii, T. Pajdla, and J. Sivic, “Netvlad: Cnn architecture for weakly supervised place recognition,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 5297–5307.
- [20] J. Lin and F. Zhang, “Loam_livox: A fast, robust, high-precision lidar odometry and mapping package for lidars of small fov,” *arXiv preprint*, 2019.
- [21] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, “Octomap: An efficient probabilistic 3d mapping framework based on octrees,” *Autonomous robots*, vol. 34, no. 3, pp. 189–206, 2013.
- [22] K. Pulli, “Multiview registration for large data sets,” in *Second International Conference on 3-D Digital Imaging and Modeling (Cat. No. PR00062)*. IEEE, 1999, pp. 160–168.
- [23] E. Olson, J. Leonard, and S. Teller, “Fast iterative alignment of pose graphs with poor initial estimates,” in *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006*. IEEE, 2006, pp. 2262–2269.
- [24] 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.