BoW3D: Bag of Words for Real-Time Loop Closing in 3D LiDAR SLAM

Yunge Cui ⬮, Xieyuanli Chen ⬮, Yinlong Zhang ⬮, Jiahua Dong ⬮, Qingxiao Wu, and Feng Zhu ⬮

Abstract—Loop closing is a fundamental part of simultaneous localization and mapping (SLAM) for autonomous mobile systems. In the field of visual SLAM, bag of words (BoW) has achieved great success in loop closure. The BoW features for loop searching can also be used in the subsequent 6-DoF loop correction. However, for 3D LiDAR SLAM, the state-of-the-art methods may fail to effectively recognize the loop in real time, and usually cannot correct the full 6-DoF loop pose. To address this limitation, we present a novel <u>B</u>ag <u>o</u>f <u>W</u>ords for real-time loop closing in <u>3D</u> LiDAR SLAM, called BoW3D. Our method not only efficiently recognizes the revisited loop places, but also corrects the full 6-DoF loop pose in real time. BoW3D builds the bag of words based on the 3D LiDAR feature LinK3D, which is efficient, pose-invariant and can be used for accurate point-to-point matching. We furthermore embed our proposed method into 3D LiDAR odometry system to evaluate loop closing performance. We test our method on public dataset, and compare it against other state-of-the-art algorithms. BoW3D shows better performance in terms of F_1 max and extended precision scores on most scenes. It is noticeable that BoW3D takes an average of 48 ms to recognize and correct the loops on KITTI 00 (includes 4K+ 64-ray LiDAR scans), when executed on a notebook with an Intel Core i7 @2.2 GHz processor.

Index Terms—Bag of words, LinK3D feature, loop correction, place recognition, real-time.

I. INTRODUCTION

O NE of the basic requirements for long-term simultaneous localization and mapping (SLAM) is the fast and robust loop closing. After a long-term operation, standard frame-to-frame point registration algorithms may fail to align the current observation to the revisited places due to the large drift in the pose estimation. When the revisited places can be robustly recognized, loop closure can provide correct data association to eliminate the drifts resulting in more globally consistent estimation. The same methods used for loop detection can be also used for robot relocalization in presence of the tracking failure case. The basic technique for place recognition is to build a database from the images (for visual SLAM) or point clouds (for LiDAR SLAM) collected online by the robot so that the most similar one can be retrieved when a new sensor observation is acquired. In presence of the same scene detected, a loop closure will be performed. Admittedly, the distance-based association can also be used for loop closing with drift in a certain range. However, when applied to long-term large-scale cases, the pose drift accumulates to a large extent that distance-based association fails to find correct correspondences between the current sensor frame and the historical ones. Therefore, a fast and robust loop closing module is essential for SLAM system.

In the field of visual SLAM, many algorithms [1], [2], [3], [4], [5], [6], [7], [8] use image retrieval techniques. They typically match images by comparing their 2D features via bag-of-words (BoW) [9]. BoW achieves very efficient image retrieval, and have promoted the progress of visual SLAM. Unfortunately, in the field of 3D LiDAR SLAM, the irregularity, sparsity and disorder of LiDAR point cloud raise challenges to 3D feature extraction and representation [10], [11]. Building the BoW for 3D features, applying it to real-time loop closing, and correcting the full 6-DoF loop pose in 3D LiDAR SLAM, are still unsolved.

In this letter, we propose a novel loop closing method that builds the bag of words for 3D features extracted from LiDAR point clouds. We use LinK3D [10] as the 3D feature and build the BoW for 3D LiDAR point clouds named BoW3D. To achieve fast retrieval, the hash table is used as the basic structure of the database. The proposed retrieval and update algorithms can be performed online. After finding the candidate frame, we calculate the full 6-DoF loop pose using the accurate point-to-point LinK3D matching results with RANSAC [12] and SVD [13]. Moreover, we compare our BoW3D with state-of-the-art methods on public dataset, and embed it to the LiDAR odometry A-LOAM [14] to evaluate its performance in practice. The experimental results show that our method achieves significant improvements, and has superior real-time performance.

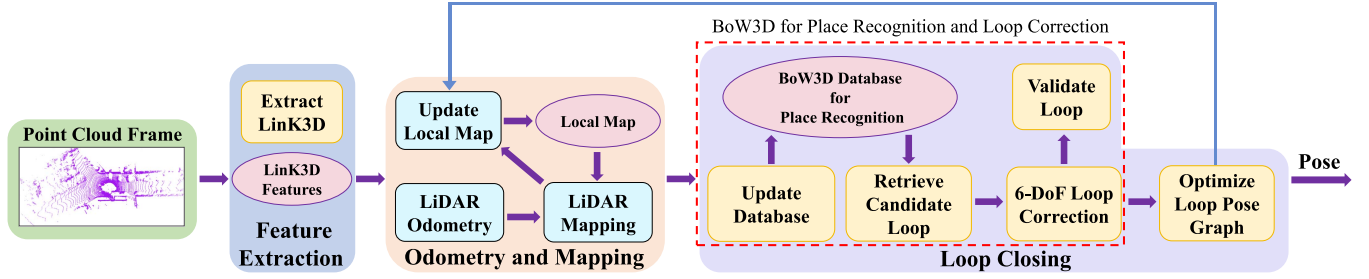To summarize, our main contributions are as follows:

Fig. 1. The workflow of our system mainly consists of three modules: (i) Feature Extraction; (ii) Odometry and Mapping of A-LOAM; (iii) Loop Closing. We embed our BoW3D to the loop closing thread, which closes the loops and obtains the globally consistent SLAM results.

- We propose a novel BoW-based method using 3D LiDAR features for the loop closing of LiDAR SLAM. It builds the database of LinK3D features, and effectively recognize the revisited places in real time.
- The proposed BoW3D can also be used to correct the full 6-DoF loop pose in real time, which provides accurate loop closing constraints for subsequent pose graph optimization in online operation.
- The proposed method has been embedded to the LiDAR odometry system in practice. Experimental results show that our method can significantly eliminate the drifts and improve the accuracy of 3D LiDAR SLAM.

II. RELATED WORK

Loop closing usually contains two steps. It first finds similar places in the database compared to the current observation, also called place recognition. Then, it estimates the loop pose and corrects the pose estimation by pose graph optimization. Many different sensor modalities have been used for finding loops, including camera images and 3D LiDAR points.

In recent years, a variety of image-based methods [1], [4], [5], [15], [16], [17] have been proposed. A great example is FAB-MAP [18], which uses a tree structure to learn the offline words' covisibility probability. However, as relying on SURF [19], it spends a lot of time on feature extraction. DBoW2 [4], used for the first time bag of binary words based on BRIEF descriptors [20] with the very efficient FAST feature detector [21], greatly improving the efficiency of loop retrieval. Due to its high efficiency, DBoW2 has been successfully used for loop closing and relocalization in several famous visual SLAM systems [5], [6], [7], [8], [22], [23]. It is worth noting that, the retrieved feature points based on DBoW2 can also be used for subsequent loop correction and optimization after integrating in long-term visual SLAM. There are three main reasons for the success of applying BoW to visual SLAM: (i) There are many 2D features that have been successfully used in vision tasks, such as SIFT [24], SURF [19], BRIEF [20] and ORB [25]. (ii) Camera SLAM usually extracts 2D features, and requires to detect the loop or perform relocalization. As a consequence, bag of words is quite suitable for camera SLAM. (iii) Bag of words compresses the image information into a more compact form. Moreover, it builds a tree to speed up the retrieving process of words. Both of

these ensure the efficiency of the algorithm in place recognition of camera SLAM.

For methods using 3D point cloud, Steder et al. [26] propose a place recognition method operating on range images generated from 3D LiDAR data, which uses a combination of bag-of-words and NARF-feature-based [27]. M2DP [28] projects a point cloud to multiple 2D planes and generates a density signature for points in each of the planes. The singular value decomposition (SVD) components of the signature are then used to compute a global descriptor. Scan contex [29] converts the point cloud scan into a visible space, and uses the similarity score to calculate the distance between two scan contexts. However, this method can only provide relative 1-DoF yaw angle estimation of loop LiDAR pair and fails to correct the full 6-DoF pose of loops. PointNetVLAD [30] leverages PointNet [31] and NetVLAD [15] to generate global descriptors. Then it proposes a "lazy triplet and quadruplet" loss function to tackle the retrieval task. ISC [32] proposes a global descriptor based on the geometry and intensity informations, then uses a two-stage hierarchical intensity scan context to detect loops. LiDAR-Iris [33] generates the LiDAR-Iris image representation to detect potential loops. SGPR [34] uses a semantic graph representation for the point cloud scenes by reserving the semantic and topological information of the raw point cloud. OverlapNet [35] adopts a siamese network to estimate an overlap of range image generated from LiDAR scans, and provides a relative yaw angle estimate of matching LiDAR pair. SSC [36] uses global semantic scan context to detect loops. OverlapTransformer [37] uses a lightweight neural network expoiting the range image representation of point cloud to achieve fast retrieval. Overall, most existing methods are either too time-consuming to detect loops in real time for online 3D LiDAR SLAM or can not provide a 6-DoF pose estimation required for LiDAR loop closing.

III. BACKGROUND REVIEW

A. Review of LinK3D Features

In this work, the proposed BoW3D is based on the LinK3D [10] feature. LinK3D consists of three parts: keypoint extraction, descriptor generation and feature matching. As shown in Fig. 2, the core idea of LinK3D descriptor is to represent the current keypoint using the neighborhood information, which is inspired by the 2D image features SIFT [24] and ORB [25]. The LinK3D descriptor is represented by a
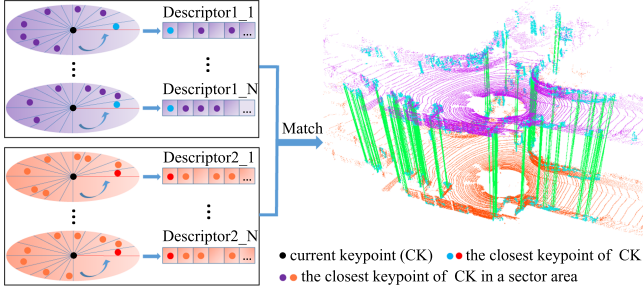
Fig. 2. The core idea of LinK3D and the matching result of two LiDAR scans based on LinK3D. The green lines are the valid matches. The descriptor of current keypoint is represented with its neighbor keypoints and described as a multi-dimensional vector. Each dimension of the descriptor corresponds to a sector area. The first dimension corresponds to the sector area where the closest keypoint of current keypoint located, and the others correspond to the areas arranged in a counterclockwise order. If there are keypoints in a sector area, the closest keypoint in the sector area is searched and used to represent the corresponding dimension of the descriptor.

180-dimension vector. Each dimension of the descriptor corresponds to a sector area. The first dimension corresponds to the sector area where the closest keypoint located, and the others correspond to the areas arranged in a counterclockwise order. LinK3D is lightweight and takes an average of 32 ms to extract features from the point cloud collected by a 64-ray laser beam, when executed on a notebook with an Intel Core i7 @2.2 GHz processor. What's more, LinK3D can be used to achieve accurate point-to-point matching, which enables it to be applied to fast 3D registration. For more details about LinK3D, please refer to its original letter [10].

B. Review of Bag of Words

In the field of 2D image, bag of words (BoW) [4] is used to recognize the revisited place by retrieving the 2D features (such as SIFT [24], ORB [25], etc.). In particular, the database of BoW for binary ORB feature has successfully applied to real-time place recognition task. BoW creates a visual vocabulary as a tree structure, in an offline step over a set of descriptors extracted from a training image dataset. When processing a new image, the vocabulary converts the extracted features of the image into a low-dimensional vector, containing the term frequency and inverse document frequency (tf-idf) [9] score. The tf-idf score is computed by:

$$tf\text{-}idf = \frac{n_{wi}}{n_i} log \frac{N}{n_w}, \quad (1)$$

where the n_{wi} represents the number of word w in image I_i, n_i represents the total number of words in I_i, N is the total number of images seen so far, and n_w represents the number of images containing word w. The idea of idf is that the less times a word appears in all images, the higher the final score will be, which indicates that the word has a higher discrimination. The higher of the total tf-idf score, the more frequent of the word in the image. If the score of a word is high enough, the similarity between the word and the words in the database is computed. If there are similar words in the database, then the invert index is used to search the corresponding images.

IV. METHODOLOGY

In this section, we introduce the proposed loop closing system based on our BoW3D. To verify the performance of BoW3D in practice, the loop closure system is embedded to the state-of-the-art A-LOAM[1] [14]. As shown in Fig. 1, the proposed system mainly consists of three parts. It first extracts the LinK3D features in the raw point clouds. Then it uses the odometry and the mapping algorithms of A-LOAM to estimate the robot poses. The third part uses the proposed BoW3D to detect loop closures and corrects the loop poses. If a loop is detected and optimized, the loop closing will give feedback to the mapping algorithm and update the local map, which provides more accurate estimation for subsequent steps.

A. BoW3D Algorithm

In this section, we introduce the proposed BoW3D algorithm. As introduced in Section III-A, each dimension of LinK3D descriptor represents a specific keypoint in the corresponding area, which makes LinK3D descriptor quite discriminative. As a result, our method needs no further conversion for features by building a tree-structured vocabulary in an offline step. Therefore, our BoW3D does not require to load additional vocabulary file, which is more convenient for users. The structure of the database in memory is shown in Fig. 3, the hash table is used to build a one-to-one mapping between the word and the places where the word has appeared. The computational cost of hash table is $O(1)$ theoretically, which makes it be suitable for efficient retrieval. As shown in Fig. 3, the word in the vocabulary of BoW3D is constructed by two parts: One is the non-zero dimension of LinK3D; The other is the corresponding dimension where the word located. The place (point cloud frame) also consists of two parts: One is the frame Id; The other is the descriptor ID in the frame.

1) Retrieval Algorithm: For the words of a LinK3D descriptor, we retrieve the words and count the frequency of each place (point cloud frame) appeared. If the highest one is greater than the frequency threshold Th_f, this place is considered as a candidate place. In addition, for fast retrieval, the inverse document frequency (idf) is used to avoid retrieving the word appeared in multiple places. More specifically, for the words appears obviously more often than other words, they are less discriminative and reduce the retrieval efficiency. Therefore, we define a $ratio$ factor that is similar to idf to measure the difference between the number of places in current set and the average in all sets. We use it to determine whether we should reserve the place set of current word when count the number of places. The $ratio$ is defined as follows:

$$ratio = N_{set}/\left(\frac{N}{n_w}\right), \quad (2)$$

where N_{set} is the number of places in current set. $\frac{N}{n_w}$ represents the quantity average of one place set, n_w is the number of words in vocabulary and N is the total number of places seen so far. If the $ratio$ of a place set is greater than threshold Th_r, the

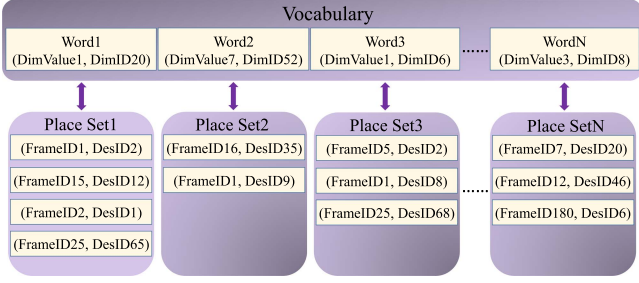[1]https://github.com/HKUST-Aerial-Robotics/A-LOAM

Fig. 3. The data structure of BoW3D. The hash table is used for the retrieval. The word of BoW3D consists of the non-zero value (Dim-value) in the descriptor and the corresponding dimension (Dim-ID). Each word corresponds to a place set, in which the word has appeared. The place also consists of two parts, one is frame ID, the other is the descriptor ID in the frame.

place set will not be used for counting. The retrieval algorithm is shown in Algorithm 1.

2) Loop Correction: Loop correction is used to provide constraint for the pose graph optimization in backend. The observation constraint of the loop is firstly calculated based on the matching result of LinK3D, then RANSAC [12] is used to remove the mismatches. We follow the approach in [13] to calculate the loop pose $\boldsymbol{T_{l,c}}$, which is the estimation between the loop frame l and the current frame c.

Given the point set $\{S\}_c$ of current frame and the matching point set $\{S\}_l$ of loop frame, we compute the loop by minimizing the following cost function:

$$r_{l,c}(\boldsymbol{R_{l,c}}, \boldsymbol{t_{l,c}}) = \frac{1}{2} \sum_{i=1}^{n} \| s_l^i - (\boldsymbol{R_{l,c}} s_c^i + \boldsymbol{t_{l,c}}) \|^2, \quad (3)$$

where $\boldsymbol{s}_l^i \in \{S\}_l$ and $\boldsymbol{s}_c^i \in \{S\}_c$ are the corresponding matching points. $\boldsymbol{R_{l,c}}$ and $\boldsymbol{t_{l,c}}$ are the rotation and the translation of $\boldsymbol{T_{l,c}}$ transformation respectively. $\boldsymbol{T_{l,c}}$ is defined as follows:

$$\boldsymbol{T_{l,c}} = \begin{bmatrix} \boldsymbol{R_{l,c}} & \boldsymbol{t_{l,c}} \\ \boldsymbol{0}^T & 1 \end{bmatrix}. \quad (4)$$

We first calculate the centroid $\boldsymbol{s}_l$ and $\boldsymbol{s}_c$ from $\{S\}_l$ and $\{S\}_c$. Let $\hat{\boldsymbol{s}}_l^i$ and $\hat{\boldsymbol{s}}_c^i$ be the coordinates of the corresponding point $\boldsymbol{s}_l^i$ and $\boldsymbol{s}_c^i$, which removes the centriod $\boldsymbol{s}_l$ and $\boldsymbol{s}_c$ respectively. Then we calculate the matrix:

$$\boldsymbol{W} = \sum_{i=1}^{n} \hat{\boldsymbol{s}}_l^i \hat{\boldsymbol{s}}_c^{iT}. \quad (5)$$

Find the SVD decomposition of $\boldsymbol{W}$,

$$\boldsymbol{W} = \boldsymbol{U\Sigma V}^T. \quad (6)$$

If W is full rank, we will get the solution of $\boldsymbol{R_{l,c}}$ and $\boldsymbol{t_{l,c}}$ by:

$$\boldsymbol{R_{l,c}} = \boldsymbol{VU}^T,$$
$$\boldsymbol{t_{l,c}} = \boldsymbol{s}_l - \boldsymbol{R_{l,c}} \boldsymbol{s}_c. \quad (7)$$

This allows us to correct the loop pose. Moreover, the loop pose is also used for the geometry verification of candidate loop frame, and we set distance threshold Th_{dis} to verify wether the candidate loop is valid.

Algorithm 1: Retrieval Algorithm.

Input : **Des** (LinK3D descriptor of a frame)
Output: **Candidate Loop Frame ID**

1 **Define:**
2 Tab_h: the hash table used to count how many times of each place occurs
3 $MaxFreqPlace$: the place with the maximum frequency in Tab_h
4 **Main Loop:**
5 **for** *a Word in* **Des** **do**
6   **if** *the Word is in* **Vocabulary** **then**
7      $PlaceSet$ = GetPlaceSet($Word$) in **Database**;
8      $ratio$ = ComputeRatio($PlaceSet$);
9   **if** *ratio* $> Th_r$ **then**
10   continue next cycle;
11   **else**
12   **for** *a Place in PlaceSet* **do**
13   **if** *the Place appear in* Tab_h **then**
14   The corresponding frequency in Tab_h + 1;
15   **else**
16   Push the *Place* into Tab_h;
17   **end**
18   **end**
19   **end**
20   **else**
21   continue next cycle;
22   **end**
23 **end**
24 $MaxFreqPlace$ = GetPlaceWithMaxFrequency(Tab_h);
25 **if** *the maximum frequency* $> Th_f$ **then**
26   **Candidate Loop Frame ID** = Frame ID in $maxFreqPlace$;
27   return **Candidate Loop Frame ID**;
28 **else**
29   return -1;
30 **end**

Algorithm 2: Update Algorithm.

Input: **Des** (LinK3D descriptor of a frame)
1 **Main Loop:**
2 **for** *a Word in* **Des** **do**
3   **if** *the Word is in* **Vocabulary** **then**
4      $PlaceSet$ = GetPlaceSet($Word$) in **Database**;
5      $PlaceSet$.insert(($FrameID, DesID$));
6   **else**
7      $PlaceSet$ = DefineNewSet(($FrameID, DesID$));
8   **Database**.insertNewWordSet($Word$, $PlaceSet$);
9   **end**
10 **end**

---

3) Update Algorithm: We propose an update algorithm to add new words and places to the database. To improve the efficiency of update and retrieval, we only add a certain number of features to the database. The descriptors are selected based on the distance between their corresponding keypoints and the LiDAR centers. Specifically, only a certain number of closer features are added to the database. Similarly, when retrieve the

loops, a certain number of closer features are used to retrieve. The update algorithm is shown in Algorithm 2.

B. Loop Optimization

After correcting the loop pose, the pose graph for the loop frames is built for subsequent loop optimization. The vertexs of pose graph are the global poses to be optimized. The edges (connections between the vertexs) of pose graph are the observation constraints, which consist of the relative poses between the sequential frames, and the corrected loop poses. We define the residual between frame i and j as:

$$r_{i,j}(\boldsymbol{T}_{w,i}, \boldsymbol{T}_{w,i}) = \ln(\boldsymbol{T}_{i,j}^{-1}\boldsymbol{T}_{w,i}^{-1}\boldsymbol{T}_{w,j})^{\vee}. \tag{8}$$

The pose graph is optimized by minimizing the following cost function:

$$\min_{T} \left\{ \sum_{(i,j) \in S} \|r_{i,j}\|^2 + \sum_{(i,j) \in L} \|r_{i,j}\|^2 \right\}, \tag{9}$$

where S is the set of all sequential edges and L is the set of all loop closure edges. We use the Levenberg-Marquadt method implemented in the graph optimizer g2o [38] to solve the optimization.

After poses have been optimized, we update the local map of mapping thread. Specifically, the points in local map are updated based on the optimized global poses, which are more accurate. This can ensure the global consistency of subsequent estimations.

V. Experiments

To comprehensively evaluate the performance of our algorithm, we perform experimental evaluation from four aspects: (i) Evaluate place recognition performance; (ii) Evaluate the performance of our method when integrated in 3D LiDAR SLAM; (iii) Test the hyperparameters and analyze the robustness of our method; (iv) Test the runtime for each part of the system. We perform experimental verification on KITTI [39] dataset which contains data from different street environments ranging from inner cities to suburb areas. The point clouds in KITTI were collected by a Velodyne HDL-64E S2 at the sensor rate of 10 Hz. There are 11 sequences (i.e., from 00 to 10) with ground truth poses, which are used to determine the number of true loops. Furthermore according to [40], some ground truth poses in KITTI involve large errors, and we do also find that the ground truth pose of sequence 02 and 08 involve large errors in experiment. Therefore, we use the more consistent poses refined by ICP to determine if there is a loop closure in sequence 02 and 08. Similar to SGPR [34], two point cloud scenes are regarded as a true positive loop pair if the Euclidean distance between them is less than 3 m and the time difference is greater than 30 s. These sequences (00, 02, 05, 06, 07 and 08) with loop closures are selected for the evaluation. We set $Th_{dis} = 3$ in our experiments. Experiments are performed on a notebook with an Intel Core i7 @2.2 GHz processor and 16 GB RAM.
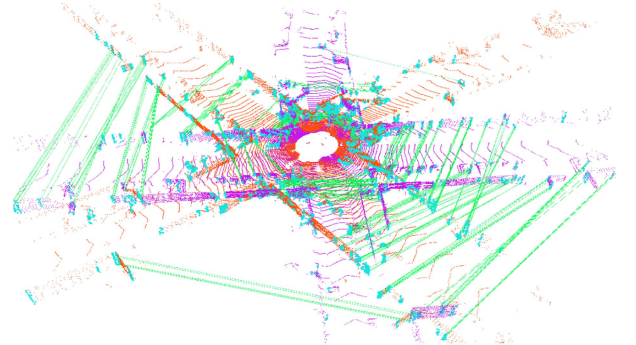
(a) The recognized loop with an angle $45°$ approximately between scan 113 (red) and scan 1564 (purple) on KITTI 00
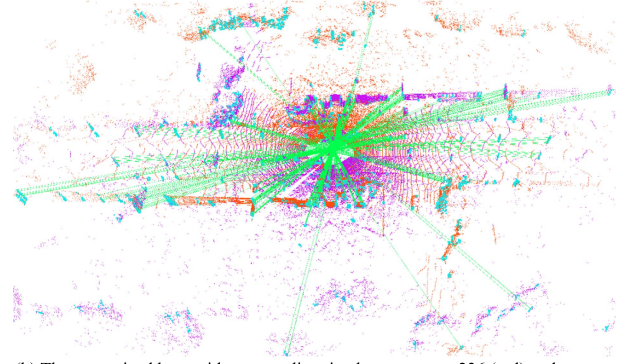
(b) The recognized loop with reverse direction between scan 236 (red) and scan 1643 (purple) on KITTI 08

Fig. 4. The recognition results based on our place recognition system and the matching results based on LinK3D on loops with different directions. The green lines are the valid matches.

A. Place Recognition Performance

In this section, we compare our method with the state-of-the-art LiDAR loop closure detection and place recognition methods, including M2DP [28], Scan context [29], PointNetVLAD [30], Intensity Scan Context (ISC) [32], LiDAR Iris [33], SGPR [34], OverlapNet [35] and SSC [36]. Following the evaluation metrics introduced in [36], we compute the maximum value of F_1 score and Extended Precision [41] (EP) of our method, and adopt the results about F_1 score and EP presented in [36] for comparison methods. In addition, we also show whether these methods can be used to correct the full 6-DoF loop pose. The F_1 score is defined as follows:

$$F_1 = 2 \times \frac{P \times R}{P + R}, \tag{10}$$

where P and R are the precision and recall, respectively. F_1 represents the harmonic mean of precision and recall, which treats precision and recall as equally important and is used to evaluate the overall performance of classification. We use the maximum F_1 score of each method for the comparison.

The Extended Precision is defined as follows:

$$EP = \frac{1}{2}(P_{R0} + R_{P100}), \tag{11}$$

where P_{R0} represents the precision at minimum recall, and R_{P100} represents the max recall at 100% precision. EP is specifically designed to validate the place recognition performance. The comparison results are shown in Table I.

(a) Trajectories on KITTI 00   (b) Trajectories on KITTI 02   (c) Trajectories on KITTI 05

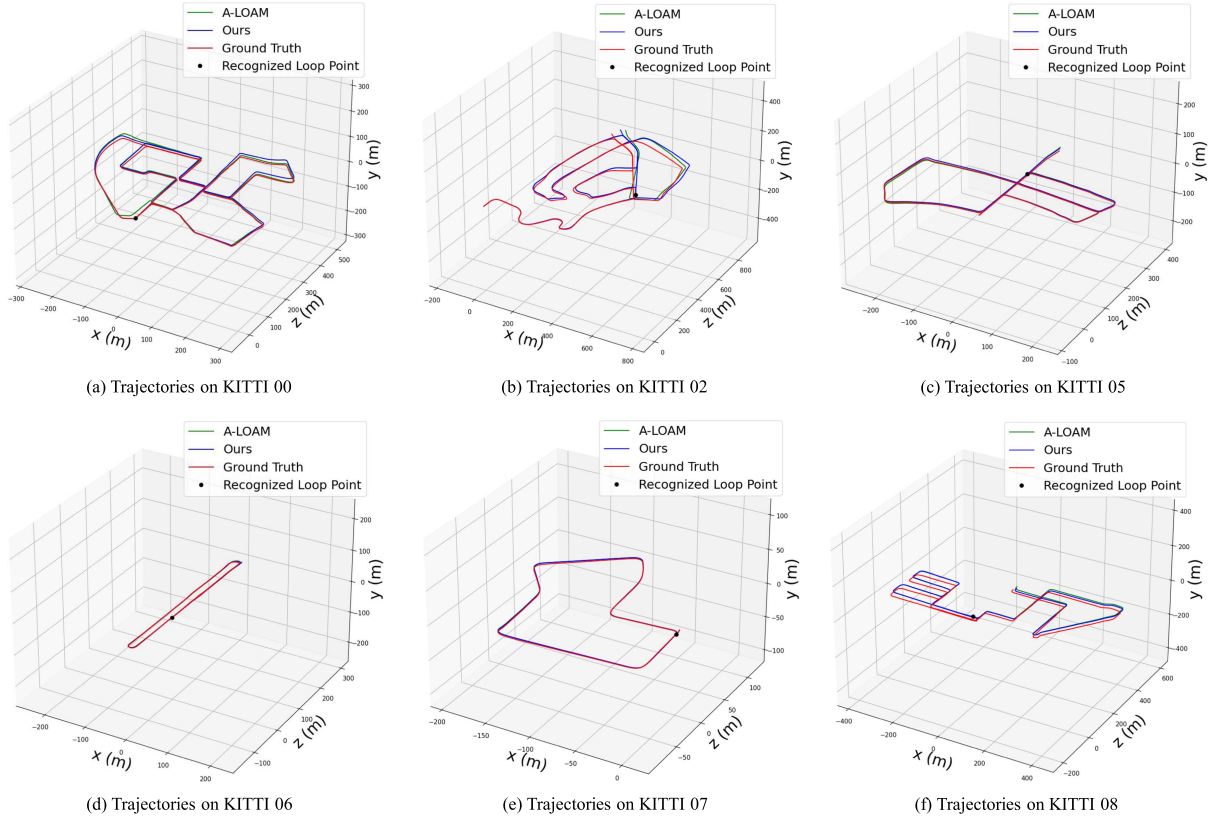(d) Trajectories on KITTI 06   (e) Trajectories on KITTI 07   (f) Trajectories on KITTI 08

Fig. 5. Comparisons between the trajectories of A-LOAM and the trajectories with our place recognition system on KITTI. We can see that our place recognition system can effectively recognize the revisited place and correct the loop pose.

TABLE I
THE COMPARISON RESULTS ABOUT F_1 MAX SCORES, EXTENDED PRECISION (F_1 MAX SCORES / EXTENDED PRECISION) AND WHETHER THESE METHODS CAN BE USED TO CORRECT THE FULL 6-DoF LOOP POSE. THE BEST SCORES ARE MARKED IN BOLD AND THE SECOND BEST SCORES ARE UNDERLINED

Method	00	02	05	06	07	08	loop correction
M2DP [28]	0.708/0.616	0.717/0.603	0.602/0.611	0.787/0.681	0.560/0.586	0.073/0.500	-
PointNetVLAD [30]	0.779/0.641	0.727/0.691	0.541/0.536	0.852/0.767	0.631/0.591	0.037/0.500	-
ISC [32]	0.657/0.627	0.705/0.613	0.771/0.727	0.842/0.816	0.636/0.638	0.408/0.543	-
LiDAR Iris [33]	0.668/0.626	0.762/0.666	0.768/0.747	0.913/0.791	0.629/0.651	0.478/0.562	-
SGPR [34]	0.820/0.500	0.751/0.500	0.751/0.531	0.655/0.500	0.868/0.721	0.750/0.520	-
Scan contex [29]	0.750/0.609	0.782/0.632	0.895/0.797	0.968/0.924	0.662/0.554	0.607/0.569	1-DoF
OverlapNet [35]	0.869/0.555	0.827/0.639	0.924/0.796	0.930/0.744	0.818/0.586	0.374/0.500	1-DoF
SSC-RN [36]	0.939/0.826	**0.890/0.745**	0.941/0.900	**0.986**/0.973	0.870/0.773	0.881/0.732	3-DoF
BoW3D (ours)	**0.977/0.981**	0.578/0.704	**0.965/0.969**	0.985/**0.985**	**0.906/0.929**	**0.900/0.866**	**6-DoF**

As shown in Table I, our BoW3D surpasses other methods in F_1 max score and EP on most sequences. Especially on sequence 08 with only reverse loops, our method can still retrieve most loops and also achieve high precision. This indicates that our method is robust to the change of view angle, and Fig. 4 shows the situations. In addition, the comparison methods cannot be used to correct the full 6-DoF loop pose, which limits subsequent loop optimization in 3D LiDAR SLAM. Our method performs well while also can be used to correct the full 6-DoF loop pose. The reasons why our method outperforms the baseline methods are as follows: (i) Our BoW3D is based on the LinK3D descriptors, and LinK3D is pose invariant. This allows our method quickly recognize the loops, even if the loops are reverse. (ii) As shown in Fig. 4, our system forms the constraints based on the accurate point-to-point matching results, and this ensures that the effective loop correction can be performed.

B. Performance on LiDAR-Based SLAM

In this experiment, we evaluate the performance of the loop closing system used in 3D LiDAR-based SLAM. To verify the accuracy of loop correction, we refer to the validation metrics defined in [40]: (i) Euclidean distance for translation; (ii) Θ for rotation. Θ is defined as follows:

$$\Theta = 2 \arcsin \left(\frac{\|R - \overline{R}\|_F}{\sqrt{8}} \right), \tag{12}$$

where $\|R - \overline{R}\|_F$ is the Frobenius norm of the chordal distance [42] between estimation and ground truth rotation matrix.
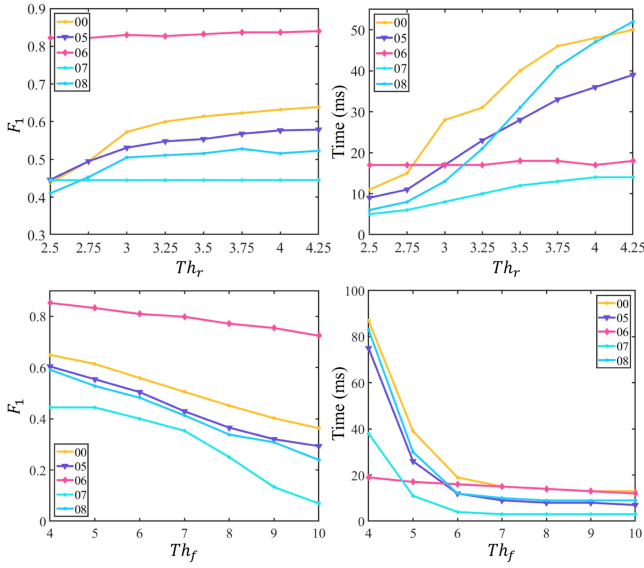
Fig. 6. The results of F_1 score and runtime with different Th_r and Th_f settings on KITTI 00.

TABLE II
THE RESULTS OF ANGULAR, TRANSLATION ERRORS AND THE VARIATION OF
RMSE BASED ON THE GROUND TRUTH WHEN BOW3D USED TO LOOP
CORRECTION AND OPTIMIZATION

Error	00	02	05	06	07	08
Angular Err (°)	0.685	1.130	0.598	0.289	0.532	1.480
Transl. Err (m)	0.764	0.162	0.238	0.060	0.138	0.037
$RMSE$ (m) (↓)	1.069	3.173	0.958	0.542	0.153	3.945

To verify the accuracy of the whole trajectories, the root mean square error ($RMSE$) is used for the evaluation. $RMSE$ is defined as follows:

$$RMSE = \left(\frac{1}{m} \sum_{i=1}^{m} \| trans(Q_i^{-1} P_i) \|^2 \right)^{\frac{1}{2}}, \qquad (13)$$

where Q_i is the ground truth pose and P_i is the estimated pose. The loop correction accuracy and the trajectory comparison results between our SLAM system and A-LOAM are shown in Table II and Fig. 5, respectively.

From the comparison results in Fig. 5, we can see that the loop closing system can effectively correct the cumulative errors, and the pose estimations with loop closing are better than the original ones. We show the quantitative decrease of $RMSE$ after using our loop closing compared to the original SLAM pose estimation results in Table II. The results demonstrate that the proposed loop closing can effectively reduce the drifts of 3D LiDAR SLAM system.

C. Hyperparameters Setup and Robustness Analyzation

In this section, we provide more studies on parameter settings about Th_r and Th_f in Algorithm 1 through experiment, which is important to the performance of BoW3D. For a wide test, we set eight parameters about Th_r, and it starts from 2.5 to 4.25 with an interval of 0.25. Th_f starts from 4 to 10 with an interval of 1. The F_1 score and the average detection time of BoW3D are
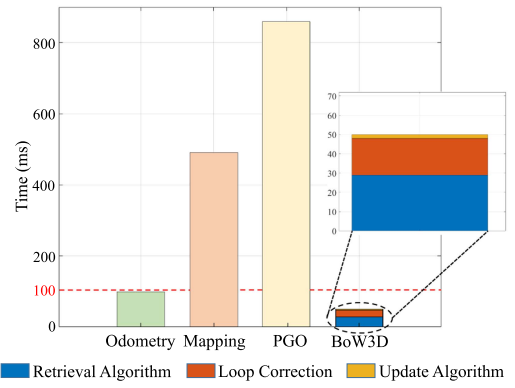

Fig. 7. The average runtime for each part of the system to process one point cloud frame on KITTI 00.

used to measure the performance of different parameter settings. The results are shown in Fig. 6.

Robustness Analyzation. We can see from Fig. 6 that as Th_r increases, the required runtime also increases and the same as the F_1 scores. When we set Th_r larger than 3, the F_1 scores not change much on most sequences and the runtime still increases. When we set Th_f less than 5, although the F_1 is higher, the algorithm also requires much more time to detect the true places. If we set Th_f larger than 8, this will reduce the robustness of our algorithm because of the low F_1 score. To make our algorithm robustly work, and trade off the runtime and accuracy, we set the Th_r as 4 and the Th_f as 5.

D. System Runtime

In this experiment, we evaluate the average runtime of each module in the SLAM system after integrating our loop closing for processing one LiDAR scan. KITTI 00 is used for the evaluation, which includes 4K+ LiDAR scans. We set the $Th_r = 4$, $Th_f = 5$. Moreover, we set the number of closer features as 5 when add them to the database, and set it as 3 when retrieve from the database. The runtime of each module is shown in Fig. 7. Note that each module of the system operates separately in different threads. Although the runtime of mapping thread and pose graph optimization (PGO) are more than 100 ms, they can be performed online due to their low frequency. In particular, BoW3D takes overall less than 100 ms to process one frame, which ensures the real-time performance of the system when BoW3D applied to 3D LiDAR SLAM.

VI. CONCLUSION

In this letter, we propose a novel 3D-feature-based bag of words algorithm for place recognition. The proposed BoW3D exploits the LinK3D features to build the bag of words. It consists of three parts, i.e. place retrieval, loop correction and database update. The hash table is used as the overall structure of the database in BoW3D, which enables to retrieve efficiently. Compared with the state-of-the-art methods, BoW3D not only achieves competitive results, but also corrects the full 6-DoF relative loop pose in real time. Compared with deep-learning-based methods, our method does not require pre-training and GPU

resources. In addition, we also embed the proposed BoW3D to LiDAR odometry system to verify its performance in practice. Compared with original odometry algorithm, the 3D LiDAR SLAM system with BoW3D has lower drifts when there are loops. It would be interesting to overcome the shortcomings of current 3D LiDAR odometry systems without loop closing thread, and we will also extend our method to relocalization, local optimization and mapping in 3D LiDAR SLAM, to improve the efficiency, accuracy and robustness of current 3D LiDAR SLAM system.

REFERENCES

[1] M. Cummins and P. Newman, "FAB-MAP: Probabilistic localization and mapping in the space of appearance," *Int. J. Robot. Res.*, vol. 27, no. 6, pp. 647–665, 2008.

[2] A. Angeli, D. Filliat, S. Doncieux, and J.-A. Meyer, "Fast and incremental method for loop-closure detection using bags of visual words," *IEEE Trans. Robot.*, vol. 24, no. 5, pp. 1027–1037, Oct. 2008.

[3] C. Cadena, D. Gálvez-López, J. D. Tardós, and J. Neira, "Robust place recognition with stereo sequences," *IEEE Trans. Robot.*, vol. 28, no. 4, pp. 871–885, Aug. 2012.

[4] D. Gálvez-López and J. D. Tardos, "Bags of binary words for fast place recognition in image sequences," *IEEE Trans. Robot.*, vol. 28, no. 5, pp. 1188–1197, Oct. 2012.

[5] R. Mur-Artal and J. D. Tardós, "Fast relocalisation and loop closing in keyframe-based SLAM," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2014, pp. 846–853.

[6] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardos, "ORB-SLAM: A versatile and accurate monocular SLAM system," *IEEE Trans. Robot.*, vol. 31, no. 5, pp. 1147–1163, Oct. 2015.

[7] T. Qin, P. Li, and S. Shen, "VINS-mono: A robust and versatile monocular visual-inertial state estimator," *IEEE Trans. Robot.*, vol. 34, no. 4, pp. 1004–1020, Aug. 2018.

[8] C. Campos, R. Elvira, J. J. G. Rodríguez, J. M. Montiel, and J. D. Tardós, "ORB-SLAM3: An accurate open-source library for visual, visual–inertial, and multimap SLAM," *IEEE Trans. Robot.*, vol. 37, no. 6, pp. 1874–1890, Dec. 2021.

[9] J. Sivic and A. Zisserman, "Video Google: A text retrieval approach to object matching in videos," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2003, pp. 1470–1470.

[10] Y. Cui, Y. Zhang, J. Dong, H. Sun, and F. Zhu, "LinK3D: Linear keypoints representation for 3D LiDAR point cloud," 2022, *arXiv:2206.05927*.

[11] C. Shi, X. Chen, K. Huang, J. Xiao, H. Lu, and C. Stachniss, "Keypoint matching for point cloud registration using multiplex dynamic graph attention networks," *IEEE Robot. Automat. Lett.*, vol. 6, no. 4, pp. 8221–8228, Oct. 2021.

[12] M. A. Fischler and R. C. Bolles, "Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography," *Commun. ACM*, vol. 24, no. 6, pp. 381–395, 1981.

[13] K. S. Arun, T. S. Huang, and S. D. Blostein, "Least-squares fitting of two 3-D point sets," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. PAMI-9, no. 5, pp. 698–700, Sep. 1987.

[14] J. Zhang and S. Singh, "LOAM: Lidar odometry and mapping in real-time," in *Proc. Robot. Sci. Syst.*, 2014, pp. 1–9.

[15] R. Arandjelovic, P. Gronat, A. Torii, T. Pajdla, and J. Sivic, "NetVLAD: CNN architecture for weakly supervised place recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 5297–5307.

[16] Z. Chen et al., "Deep learning features at scale for visual place recognition," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2017, pp. 3223–3230.

[17] M. Zaffar, S. Ehsan, M. Milford, and K. McDonald-Maier, "CoHOG: A light-weight, compute-efficient, and training-free visual place recognition technique for changing environments," *IEEE Robot. Automat. Lett.*, vol. 5, no. 2, pp. 1835–1842, Apr. 2020.

[18] M. Cummins and P. Newman, "Appearance-only SLAM at large scale with FAB-MAP 2.0," *Int. J. Robot. Res.*, vol. 30, no. 9, pp. 1100–1123, 2011.

[19] H. Bay, T. Tuytelaars, and L. V. Gool, "SURF: Speeded up robust features," in *Proc. Eur. Conf. Comput. Vis.*, 2006, pp. 404–417.

[20] M. Calonder, V. Lepetit, C. Strecha, and P. Fua, "BRIEF: Binary robust independent elementary features," in *Proc. Eur. Conf. Comput. Vis.*, 2010, pp. 778–792.

[21] E. Rosten and T. Drummond, "Machine learning for high-speed corner detection," in *Proc. Eur. Conf. Comput. Vis.*, 2006, pp. 430–443.

[22] C. Forster, Z. Zhang, M. Gassner, M. Werlberger, and D. Scaramuzza, "SVO: Semidirect visual odometry for monocular and multicamera systems," *IEEE Trans. Robot.*, vol. 33, no. 2, pp. 249–265, Apr. 2017.

[23] R. Mur-Artal and J. D. Tardós, "ORB-SLAM2: An open-source SLAM system for monocular, stereo, and RGB-D cameras," *IEEE Trans. Robot.*, vol. 33, no. 5, pp. 1255–1262, Oct. 2017.

[24] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *Int. J. Comput. Vis.*, vol. 60, no. 2, pp. 91–110, 2004.

[25] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "ORB: An efficient alternative to SIFT or SURF," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2011, pp. 2564–2571.

[26] B. Steder, M. Ruhnke, S. Grzonka, and W. Burgard, "Place recognition in 3D scans using a combination of bag of words and point feature based relative pose estimation," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2011, pp. 1249–1255.

[27] B. Steder, R. B. Rusu, K. Konolige, and W. Burgard, "NARF: 3D range image features for object recognition," in *Proc. Workshop Defining Solving Realistic Percep. Problems Pers. Robot. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2010.

[28] L. He, X. Wang, and H. Zhang, "M2DP: A novel 3D point cloud descriptor and its application in loop closure detection," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2016, pp. 231–237.

[29] G. Kim and A. Kim, "Scan context: Egocentric spatial descriptor for place recognition within 3D point cloud map," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2018, pp. 4802–4809.

[30] M. A. Uy and G. H. Lee, "PointNetVLAD: Deep point cloud based retrieval for large-scale place recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 4470–4479.

[31] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "PointNet: Deep learning on point sets for 3D classification and segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 652–660.

[32] H. Wang, C. Wang, and L. Xie, "Intensity scan context: Coding intensity and geometry relations for loop closure detection," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2020, pp. 2095–2101.

[33] Y. Wang, Z. Sun, C.-Z. Xu, S. E. Sarma, J. Yang, and H. Kong, "LiDAR iris for loop-closure detection," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2020, pp. 5769–5775.

[34] X. Kong et al., "Semantic graph based place recognition for 3D point clouds," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2020, pp. 8216–8223.

[35] X. Chen et al., "OverlapNet: Loop closing for LiDAR-based SLAM," 2021, *arXiv:2105.11344*.

[36] L. Li et al., "SSC: Semantic scan context for large-scale place recognition," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2021, pp. 2092–2099.

[37] J. Ma, J. Zhang, J. Xu, R. Ai, W. Gu, and X. Chen, "Overlaptransformer: An efficient and yaw-angle-invariant transformer network for LiDAR-based place recognition," *IEEE Robot. Automat. Lett.*, vol. 7, no. 3, pp. 6958–6965, Jul. 2022.

[38] R. Kümmerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard, "G^{2}o: A general framework for graph optimization," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2011, pp. 3607–3613.

[39] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? The KITTI vision benchmark suite," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2012, pp. 3354–3361.

[40] W. Lu, G. Wan, Y. Zhou, X. Fu, P. Yuan, and S. Song, "DeepVCP: An end-to-end deep neural network for point cloud registration," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2019, pp. 12–21.

[41] B. Ferrarini, M. Waheed, S. Waheed, S. Ehsan, M. J. Milford, and K. D. McDonald-Maier, "Exploring performance bounds of visual place recognition using extended precision," *IEEE Robot. Automat. Lett.*, vol. 5, no. 2, pp. 1688–1695, Apr. 2020.

[42] R. Hartley, J. Trumpf, Y. Dai, and H. Li, "Rotation averaging," *Int. J. Comput. Vis.*, vol. 103, no. 3, pp. 267–305, 2013.