BTC: A Binary and Triangle Combined Descriptor for 3D Place Recognition

Chongjian Yuan[1,2,*], Jiarong Lin[1,*], Zheng Liu[1], Hairuo Wei[1], Xiaoping Hong[2], and Fu Zhang[1]

***Abstract*—Accurate and robust place recognition is essential for robot navigation, yet achieving full pose invariance and high performance across diverse scenes remains challenging. In this work, we propose a novel global and local combined descriptor named Binary Triangle Combined (BTC) descriptor. We first extract the keypoints of a point cloud by projecting the points to planes extracted therein. Any three keypoints form a unique triangle, with the lengths of its sides constituting a triangle descriptor that captures the global appearance of the point cloud. Thanks to the distinct shape of a triangle given three side lengths, the similarity between two triangles and their vertices (i.e., keypoints) correspondence can be naturally determined from the side lengths of the triangle descriptors. The matched triangle pairs evaluate the appearance similarity between two point clouds, while the vertices' correspondence enables accurate estimation of their relative pose; both are crucial for the place recognition task. To enhance the accuracy of triangle matching, BTC introduces a binary descriptor, which describes the point distribution neighboring each keypoint. The local geometry information encoded by the binary descriptor augments descriptiveness and discriminativeness to the triangle descriptor. Collectively, the two descriptors achieve both global and local descriptions of the environment with high accuracy, efficiency, and robustness. We extensively compare the proposed BTC descriptor against state-of-the-art methods (e.g., Scan Context, LCD-Net) on a wide range of datasets collected using different types of LiDAR sensors (spinning LiDARs and non-repetitive scanning LiDARs) in various environments (urban, campus, forest, park, mountain). The quantitative results demonstrate that BTC exhibits greater adaptability and significant improvement in precision compared to its counterparts, especially in challenging cases with large viewpoint variations (e.g., reverse direction, large translation and/or rotation). To share our findings and contribute to the community, we open-source our code on GitHub:** `https://github.com/hku-mars/btc_descriptor`**.**

***Index Terms*—SLAM, Recognition, Localization, Mapping**

I. Introduction

PLACE recognition, also known as loop closure detection, is a process that identifies and matches a robot's current location to previously visited places within a known environment. By reducing drift in pose estimation and enabling relocalization, it proves indispensable in robotic applications, particularly within Simultaneous Localization and Mapping (SLAM) systems such as visual SLAM [1, 2] and LiDAR SLAM [3, 4], where accurate and reliable navigation is necessary. As robots traverse large, complex environments, the demand for robust and efficient place recognition becomes increasingly crucial to ensure precise localization and consistent map generation [5]. Furthermore, place recognition facilitates rapid and exact relocalization, allowing robots to operate within pre-built maps, which avoids redundant or inconsistent mapping results [6]. As a consequence, loop detection and relocalization have drawn consistent research interests in this field, as evidenced by numerous recent works [7–11].

For the task of place recognition and relocalization, two mainstream methods have emerged, primarily distinguished by the type of sensor utilized. One approach relies on images captured by visual cameras, such as FAB-MAP [12], DBoW2 [13], and SeqSLAM [14]. The alternative method employs 3D point clouds, often acquired by LiDAR sensors. Camera-based methods offer certain advantages, such as lower cost and widespread availability. However, they also exhibit limitations and often struggle to handle challenging situations characterized by significant variations in illumination, appearance, or sensor viewpoint. In contrast, LiDAR-based methods provide several potential benefits due to their three-dimensional measurements: less sensitivity to sensor viewpoint changes and higher robustness to changes in illumination and texture of environments. This paper investigates the problem of place recognition and relocalization using LiDAR sensors.

Despite the potential benefits of LiDAR sensors for place recognition and relocalization, several challenges remain. First, achieving invariance to viewpoint changes remains a difficult task, as the point cloud representation can vary significantly depending on the sensor's position and orientation. Second, the point cloud acquired by LiDAR is spatially sparse, with non-uniform density at different distances, presenting additional complexities when processing and analyzing the point data for place recognition. This issue is further compounded by the diversity of LiDAR types, as different LiDAR types exhibit considerable differences in scanning patterns, resolution, and noise characteristics. Lastly, the use of various carrying platforms (handheld [15], vehicle-mounted [16], aerial [17]) and the wide range of environments (urban [16], campus [18], unstructured [19]) result in significant variations in point cloud distribution, scale, and appearance, which further complicates the place recognition problem.

In response to these challenges, this paper introduces the Binary Triangle Combined (BTC) descriptor, a novel place recognition descriptor for 3D point clouds. Inspired by [20], we employ point cloud accumulation to mitigate the issue of non-uniform point cloud density resulting from varying distances or LiDAR types. To achieve viewpoint invariance, we devise a global triangle descriptor composed of three side

Manuscript received July 2, 2023; revised September 12, 2023; accepted December 22, 2023. This work is supported by the University Grants Committee of Hong Kong General Research Fund (project number 17206421) and DJI Donation. *(Corresponding author: Fu Zhang.)*

*These two authors contribute equally to this work.

[1]C. Yuan, J. Lin, Z. Liu, H. Wei and F. Zhang are with the Department of Mechanical Engineering, The University of Hong Kong, Hong Kong Special Administrative Region, People's Republic of China. {ycj1, zivlin, u3007335, hairuo}@connect.hku.hk, fuzhang@hku.hk

[2]C. Yuan and X. Hong are with the School of System Design and Intelligent Manufacturing, Southern University of Science and Technology, Shenzhen, People's Republic of China.{yuancj2020,hongxp}@sustech.edu.cn

lengths of a triangle formed by three keypionts extracted from the accumulated point cloud. The triangle descriptor inherently exhibits viewpoint invariance, as the side lengths remain unchanged regardless of viewpoint or direction. While the triangle descriptor captures the global appearance of the point cloud, to further enhance its descriptive and discriminative capabilities, we develop a local descriptor called binary descriptor that encodes the local geometric information of each keypoint forming the triangle. This combination of global and local descriptors allows for accurate and robust place recognition, even in large-scale unstructured environments. The main contributions of this work are as follows:

- We design a triangle descriptor to capture the global appearance of point clouds. This descriptor is entirely invariant to rotation and translation and naturally provides vertices association for full 6D relative pose estimation.
- To enhance the descriptiveness and discrimination of the triangle descriptor especially in large complex scenes, we design a local descriptor, called binary descriptor, which encodes the local geometric information of each triangle vertex in the triangle descriptor.
- We propose a loop retrieval strategy that leverages the binary and triangle combined descriptor. This strategy enhances the matching process by considering the global appearance and local geometric information, allowing for robust and precise place recognition in diverse and challenging environments.
- Our binary and triangle-combined approach, termed BTC, is assessed on numerous datasets encompassing a variety of LiDAR types, carrying platforms, sequence lengths, and environments. It demonstrates superior performance over other state-of-the-art methods in terms of robustness, scalability, and computational efficiency.
- We integrate BTC into a SLAM system and verify its ability to improve the accuracy of the odometry. Building upon this, we further develop a multi-map fusion system that enables the fusion of multiple indoor and outdoor point cloud maps collected at different times. The entire BTC implementation code, the map fusion system, and the complete SLAM system are open-sourced on GitHub: `github.com/hku-mars/btc_descriptor`.[1]

This paper is an extension of the previous conference paper [11], which proposed the original idea of triangle descriptor. Compared to [11], this paper extends in four critical aspects: 1) the introduction of the binary descriptor, which provides a more detailed and discriminative local representation of the point cloud geometry; 2) an improved keypoint extraction strategy tailored for the binary descriptor, mitigating discretization issues caused by point cloud voxelization; 3) the loop retrieval strategy that incorporates both the triangle and binary descriptor; 4) a more extensive and detailed experimental evaluation, which demonstrates the superior performance of BTC in terms of robustness, accuracy, and efficiency across a wide range of challenging scenarios and applications.

[1]Our codes will be released once this work is accepted.

II. Related Works

In this section, we provide a literature review on both vision-based and LiDAR-based place recognition methods, focusing on global and local descriptions.

A. Vision-based Place Recognition Methods

In the past, handcrafted local features such as SIFT [21], SURF [22], and ORB [23] were popular choices for image matching due to their invariance to multiple transformations. FAB-MAP [12] leveraged these local descriptors to construct a probabilistic model and achieved place recognition by computing likelihood scores for each candidate location. DBoW2 [13] used a bag of binary words approach based on BRIEF [24] to enable real-time loop retrieval.

Handcrafted features were typically sensitive to appearance changes (e.g., varying lighting conditions, seasonal variations, and weather conditions) and noise. In complex and dynamic environments, relying solely on local descriptors might be insufficient. Global descriptors capture the overall appearance of an image or a region of interest without depending on specific local feature points. For example, the GIST Descriptor [25] captures the global structure of a scene by computing the spatial envelope. SeqSLAM [14] uses a downsized version of the input images for place recognition. Although these whole-image global descriptors are more robust to local noise and outperform local feature descriptors when lighting conditions change [26], they still struggle with pose invariance due to the projection of 3D scenes onto the 2D image planes. Recently, Mo and Sattar [27] adapted LiDAR descriptors for 3D points obtained from stereo-visual odometry to achieve high accuracy and robustness against visual appearance changes. This work verified the advantages of the 3D point cloud in place recognition compared to appearance-only visual images.

B. LiDAR-based Place Recognition Methods

LiDAR-based place recognition methods have been developed in recent years as an alternative approach to vision-based methods. In the early days of LiDAR development, researchers extract local features from 2D range image [28] or directly from 3D point clouds (e.g., 3D SIFT [29], PFH [30], SHOT [31]), as these techniques were already well-established in the field of computer vision. Bosse and Zlot [32] further use a 3D adaptation of the Gestalt Descriptor [33] to encode the neighborhood of keypoints as a vector to enhance the descriptiveness of local keypoints. These local descriptors capture fine-grained geometric details, making them highly discriminative for matching and recognition tasks. However, these methods exhibit sensitivity to the density and resolution of 3D point clouds, as well as viewpoint changes. In real-world applications, LiDAR naturally collects sparse point clouds at a distance and denser point clouds up close, resulting in inconsistent densities across the point cloud. This variability can lead to fluctuations in feature extraction and matching, degrading the performance and accuracy of place recognition.

In recent years, researchers have gained interest in global descriptors for LiDAR-based place recognition since global

descriptors are more robust to local noise and point cloud density. Unlike local descriptors, global descriptors capture the overall appearance of the scene, such as the distribution of certain features or objects in the scene. One commonly used approach to achieve a global description is voxelization. Magnusson *et al.* [34] voxelizes the 3D point clouds and calculates a histogram matrix through normal distribution transform. VBRL [35] divides a 3D point cloud input into voxels and uses multi-modal features extracted from these voxels to perform place recognition. MinkLoc3D [36] creates a 3D voxel grid representation through sparse voxelization and a 3D convolutional neural network (CNN). Although voxelization can help reduce the impact of noise and outliers, thereby making the recognition process more robust, voxelization methods are limited by their lack of viewpoint invariance and the artifacts and aliasing introduced by discretization. Combining voxel-based approaches with octree methods [37] or Adaptive Receptive Fields [38] may help alleviate the impact of discretization, but the challenge of viewpoint invariance still remains.

Another popular global description approach is through projection. M2DP [39] generates the signature vector by projecting the raw point cloud into multiple 2D planes. Inspired by [40], Giseop Kim and Ayoung Kim develop a novel global descriptor called Scan Context [8], which combines projection and space partitioning to encode the 2.5-D information within an image. Their follow-up work, Scan Context++ [9], introduced a new spatial division strategy to achieve lateral invariance and utilized a sub-descriptor to speed up loop retrieval, showing promising results in urban environments. More recently, some methods [41, 42] generate Bird's Eye View (BEV) images through projection and then perform place recognition using these BEV images. Building on this idea, several methods [43–45] have been developed that extract descriptors in the frequency domain from the BEV images, achieving roto-translational invariance. Compared to voxel-based methods, projection methods are not affected by discretization and can provide some level of invariance to viewpoint changes (e.g., yaw angle [8]). However, these methods are still sensitive to viewpoint changes, as the resulting 2D or 2.5D representations can vary significantly if the sensor's position and orientation change. This can lead to recognition challenges when the sensor undergoes significant motion. Moreover, many of these methods often rely on registration algorithms like Generalized ICP [46] and TEASER [47] for pose correction.

Since both voxelization and projection have limitations in achieving invariance to pose changes, some approaches employ local features (e.g., point-wise features [48, 49], plane features [50], and semantic features [20]) for global descriptor encoding and local matching. BoW3D extracts keypoints through Link3D [51] and adopts the BoW (Bag of Words) for loop retrieval. V. Nardari *et al.* [49] propose a polygon descriptor constructed with 2D landmarks to achieve place recognition in forests. Lip-Match [50] formulates each submap as a fully-connected graph with nodes representing planes and achieves place recognition through geometric constraints. SegMatch [20] extracts semantic features to accomplish global localization. Recently, some DNN-based methods have integrated deep neural networks into the process of local features extraction (e.g., LCD-Net [10], Logg3D-Net [52]) and global descriptor encoding (e.g., LocNet [53], OverlapNet [54]). These methods aim to capture both the local details and the overall structure of the environment and may be more robust to viewpoint changes when compared with voxelization and projection-based methods. However, these methods are sensitive to local feature extraction since the performance of global descriptors generated by encoding local features is highly dependent on the quality and robustness of the local features. Besides, extracting local features and encoding them into global descriptors can be computationally expensive. DNN-based methods usually require GPU acceleration to achieve real-time performance.

Our *BTC (Binary Triangle Combined)* descriptor is designed to leverage both local and global descriptors to respectively preserve the local geometry and global appearance of a scene. In contrast, existing methods often focus on either local or global descriptions, struggling to balance the two and could lead to issues in large-scale environments or real-time applications. Moreover, the design of our local and global descriptions also differs significantly from existing methods. When compared to local description methods, such as 3D SIFT[29], PFH[30], SHOT[31], and the Gestalt Descriptor [33], the binary descriptor of BTC offers notable advantages in terms of pose invariance and computational efficiency. For global description, when compared to voxelization-based [34–36], and projection-based [8, 9, 39] techniques, the triangle descriptor of BTC demonstrates strong viewpoint invariance across all six degrees of freedom, while when compared to local feature-based global description methods [10, 20, 48–50, 52–54], our triangle descriptor has a more concise and efficient encoding process while preserving the full 6-DoF pose invariance. Lastly, when compared to DNN-based methods [10, 36, 52–54], BTC offers better adaptability to various LiDAR types, equipment types, and environmental conditions.

III. System Overview

The proposed *BTC (Binary Triangle Combined)* descriptor aims to achieve efficient and accurate loop detection, a crucial component of Simultaneous Localization and Mapping (SLAM) systems. By introducing binary descriptors, our approach provides a compact and efficient local representation of the environment. Meanwhile, the triangle descriptors capture the high-level structure and exhibit strong viewpoint invariance, enabling fast and robust loop retrieval.

Fig. 1 provides an overview of our system. The input consists of a sequence of registered LiDAR scans, which are first accumulated into submaps. Then, keypoints are extracted from the submap and encoded into both binary descriptors and triangle descriptors. The two descriptors are combined to form the Binary Triangle-Combined (BTC) descriptors, which are queried from the descriptor database to retrieve a fixed number of candidate submaps. From the candidate pool, the most suitable one is selected through binary descriptor matching and geometrical verification and the 6 DoF pose is calculated. Once the results are obtained, the current BTC descriptor is added to the database for future use.

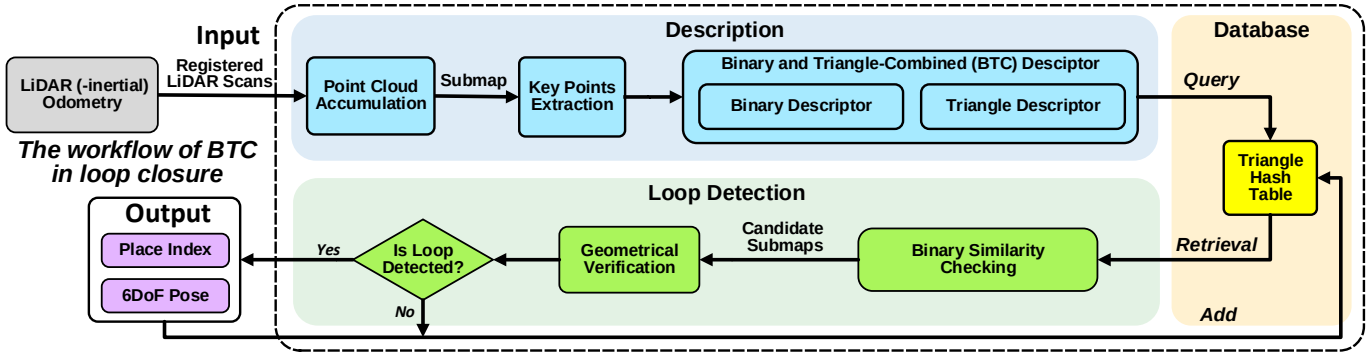


Fig. 1: System overview of the Binary Triangle Combined (BTC) descriptor for place recognition in a SLAM system.

In the following sections, we present the proposed BTC descriptor in Section IV, database construction in Section V, and loop detection in Section VI

IV. Description

In this section, we present our approach to attaining both local and global descriptions of a 3D point cloud. The pipeline commences with the construction of submaps and extraction of keypoints from the submap. We then present our Binary Triangle-Combined (BTC) descriptor, a novel representation that captures the essential local geometry and global appearance necessary for efficient place recognition.

A. Submap Construction

Inspired by [20], we mitigate the effects of uneven point cloud density and various scanning patterns by performing loop detection on submaps, which is an accumulation of certain recent LiDAR scans. This submap accumulation strategy is particularly helpful for low-resolution LiDARs (e.g., 16-ray spinning LiDARs) or non-repetitive scanning LiDARs (e.g., Livox series LiDARs), where accumulation increases the point cloud density irrespective of LiDAR scanning patterns.

A submap can be constructed from a LiDAR(-inertial) odometer [4], which registers each new incoming LiDAR scan into the current submap in accumulation. Each submap consists of points accumulated from n_s consecutive scans, leading to a denser point cloud representation.

B. Keypoints Extraction

1) Plane Detection: When given a point cloud submap, we first perform plane detection by region growing. Specifically, we divide the entire point cloud into voxels of a given size ΔL (e.g., $\Delta L = 1 \sim 2$m). Each voxel contains a group of points $\mathbf{p}_i$ $(i = 1, ..., N)$. We then calculate the point covariance matrix $\mathbf{\Sigma}$ for each voxel:

$$\bar{\mathbf{p}} = \frac{1}{N}\sum_{i=1}^{N}\mathbf{p}_i; \quad \mathbf{\Sigma} = \frac{1}{N}\sum_{i=1}^{N}(\mathbf{p}_i - \bar{\mathbf{p}})(\mathbf{p}_i - \bar{\mathbf{p}})^T; \tag{1}$$

Perform the eigenvalue decomposition of matrix $\mathbf{\Sigma}$ to obtain its eigenvalues $\lambda_1, \lambda_2, \lambda_3$ (with $\lambda_1 \geq \lambda_2 \geq \lambda_3$) and corresponding eigenvectors $\mathbf{u}_1, \mathbf{u}_2, \mathbf{u}_3$. Then the plane criterion is defined by two pre-set thresholds, σ_1 and σ_2, such that a voxel is classified as a plane if $\lambda_3 < \sigma_1$ and $\lambda_2 > \sigma_2$. A plane voxel is represented as $\boldsymbol{\pi}$, which contains the plane normal vector $\mathbf{u}_3$, center point $\bar{\mathbf{p}}$, number of points N, and point covariance matrix $\mathbf{\Sigma}$. Applying this criterion to all voxels, we obtain a list of planes denoted by $\Pi = (\boldsymbol{\pi}_1, \boldsymbol{\pi}_2, \ldots, \boldsymbol{\pi}_k)$. Fig. 2 illustrates the plane detection result obtained through voxelization on the first submap of the KITTI00 dataset. The plane points are colored according to their voxel ID. These planes, encapsulating key geometric information of the scene, will be used for the following keypoint extraction and also the geometrical verification shown in Section VI-B.

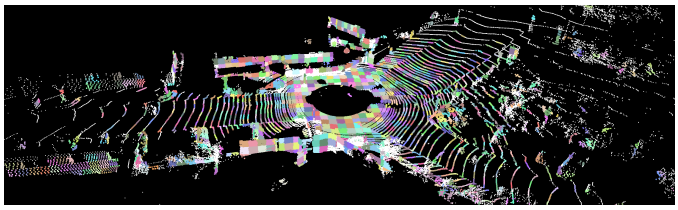


Fig. 2: Plane detection using voxelization on the first keyframe of the KITTI00 dataset with a voxel size of 2m. Points form a plane are collored based on their Voxel ID.

2) Reference Plane Generation: Upon acquiring the list of planes Π, we proceed to generate reference planes. This involves merging adjacent planes to yield larger planes. Specifically, the plane merging begins by selecting an initial plane voxel and progressively examining the planes in neighboring voxels. If the plane in neighboring voxels has a similar normal vector and a near-to-zero distance, it is merged with the initial plane. Specifically, if the initial plane voxel $\boldsymbol{\pi}_i$ and the neighboring plane voxel $\boldsymbol{\pi}_j$ have centers $\bar{\mathbf{p}}_i, \bar{\mathbf{p}}_j$, and normals $\mathbf{u}_{3i}, \mathbf{u}_{3j}$, the merging criteria is

$$\begin{aligned} &\max\left(\mathbf{u}_{3i}^T(\bar{\mathbf{p}}_i - \bar{\mathbf{p}}_j), \mathbf{u}_{3j}^T(\bar{\mathbf{p}}_j - \bar{\mathbf{p}}_i)\right) \leq \sigma_d \\ &\arccos(\mathbf{u}_{3i} \cdot \mathbf{u}_{3j}) \leq \sigma_u \end{aligned} \tag{2}$$

where σ_d, σ_u are two thresholds. The merged plane $\boldsymbol{\pi}_m$ has a points number N_m, center point $\bar{\mathbf{p}}_m$ and point covariance matrix $\mathbf{\Sigma}_m$ as follows:

$$\begin{aligned} &N_m = N_i + N_j, \quad \bar{\mathbf{p}}_m = \frac{N_i\bar{\mathbf{p}}_i + N_j\bar{\mathbf{p}}_j}{N_m}, \\ &\mathbf{\Sigma}_m = \frac{N_i(\mathbf{\Sigma}_i + \bar{\mathbf{p}}_i\bar{\mathbf{p}}_i^T) + N_j(\mathbf{\Sigma}_j + \bar{\mathbf{p}}_j\bar{\mathbf{p}}_j^T)}{N_m} - \bar{\mathbf{p}}_m\bar{\mathbf{p}}_m^T. \end{aligned} \tag{3}$$

In addition, the normal vector $\mathbf{u}_m$ of the merged plane is calculated through the eigenvalue decomposition of $\mathbf{\Sigma}_m$. This merging process continues in a region-growing manner until neighboring voxels have no planes.

We sort the merged planes in descending order according to the number of contained points. Then, the first M planes with the most points are selected as reference planes. Most of the time, selecting one reference plane ($M = 1$) is sufficient. In certain cases characterized by uneven terrain (e.g., mountainous regions, urban environments with tall buildings, or areas with significant changes in elevation), the selection of two or more reference planes ($M \geq 2$) may be necessary to account for the complexity of the landscape. Fig. 3 shows the generation of three reference planes (i.e., $M = 3$) in an urban environment. Despite different voxel sizes, the generated reference plane remains consistent.

3) Height-encode Image Generation: After obtaining the reference planes, we project the 3D point cloud onto each reference plane, creating M height-encoded images with a pixel area $r \times r$ m^2. The choice of r is a trade-off between computational efficiency and the ability to capture sufficient detail in the height-encoded image.

To encode the height information, as depicted in Fig. 4, we select a maximum height $h_{\max}$ above each pixel on the plane and divide it into m layers with a fixed resolution Δh. For each pixel, we compute a binary string $\mathtt{b}$ composed of m bits, where each bit is set to one if the corresponding layer contains any points at this height range or else set to zero. Summing all the m bit values leads to the pixel intensity, which is saved to each pixel along with the binary string. A height-encoded image example is depicted in Fig. 5(b).

4) Keypoints Extraction: With the M height-encoded images in hand, we proceed to extract keypoints on each image, a process demonstrated in Fig. 5(c). Keypoints are determined by identifying pixels with the maximum intensity in their local 5×5 area. These local maxima represent areas with high points population, hence retaining the most information of the original 3D point cloud. To suppress the number of keypoints, we set a threshold σ_I on the local maximum intensity. Only above this threshold, a pixel with the local maximum intensity is selected as a keypoint. Once a keypoint is identified in the height-encoded image, we determine its 3D coordinates within the submap. To do this, we first determine the point location on the reference plane by averaging the 2D coordinates of all points above the pixel used for height encoding. This in-plane location is then used to calculate the full 3D location of the keypoint based on the plane parameter. By utilizing the average 2D coordinates of projected points rather than the pixel's center, we attain sub-pixel conversion accuracy. Fig. 5(d) illustrates the extracted keypoints (depicted as yellow squares) and the attached binary string.

C. Binary Triangle-Combined (BTC) Descriptor

In this section, we present the formulation of the Binary Triangle-Combined (BTC) descriptor, which consists of the binary descriptor and the triangle descriptor.

1) Binary Descriptor: The binary descriptor is the binary string in height-encode image generation, which provides a compact yet effective representation of the local geometry of the extracted keypoint. The formal definition of the binary descriptor is detailed in Data Structure 1, which encompasses:

- $\mathtt{b}$: the binary string of the keypoint, encapsulating the vertical distribution of points above the keypoint.
- $\mathtt{p}$: the 3D location of the keypoint. While the 3D location does not constitute a descriptor attribute due to its lack of pose invariance, it is included in the data structure to provide essential information for subsequent geometric verification and relative pose estimation.

Remark 1: Since the keypoints and binary descriptors are extracted from the reference plane detected from the point cloud itself, instead of a fixed projection plane (e.g., M2DP [39], Scan Context [8] and BEVPlace [41]), they are invariant to pose transformations. That is, transforming the point cloud from one frame to another leads to the same keypoint (with coordinates transformed accordingly) and binary descriptor.

2) Triangle Descriptor: To encapsulate the geometric relation among keypoints and develop a global representation of the scene, we introduce a triangle descriptor. The rationale behind this choice lies in the inherent properties of triangles. First, triangle shapes are completely invariant to rigid transformations, meaning that at different poses, a triangle has the same side lengths. Second, as the simplest polygons, a triangle is a stable shape, meaning that once the side lengths are determined, the shape of the triangle is uniquely determined. Additionally, if the three side lengths are distinct, the triangle vertices' correspondence can also be determined uniquely. Such properties make them very suitable for representing spatial configurations formed by groups of keypoints.

The formal definition of the triangle descriptor is shown in Data Structure 2, which has the following elements:

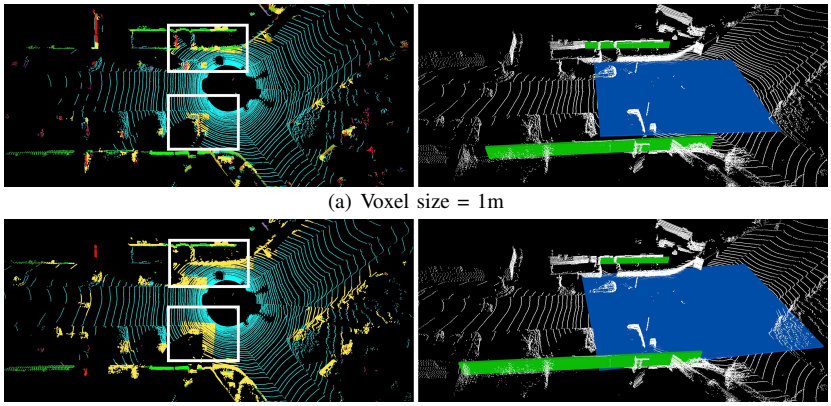
- $\mathtt{l}_1$, $\mathtt{l}_2$, $\mathtt{l}_3$: the lengths of the three sides of a triangle, in the convention that $\mathtt{l}_1$ is the side between vertices $\mathtt{p}_1$ and $\mathtt{p}_2$, $\mathtt{l}_2$ is the side between $\mathtt{p}_2$ and $\mathtt{p}_3$, and $\mathtt{l}_3$ is the side between $\mathtt{p}_2$ and $\mathtt{p}_3$. Moreover, the side lengths are sorted in ascending order (i.e., $\mathtt{l}_1 \leq \mathtt{l}_2 \leq \mathtt{l}_3$).

The ascending (or descending) side lengths order ensures that all triangles with the same side lengths (i.e., they are the same shape) will have a unique triangle descriptor $\mathtt{T} = (\mathtt{l}_1, \mathtt{l}_2, \mathtt{l}_3)$. In return, if two descriptors have the same sorted side lengths, they are of the same triangle shape and their vertices $\mathtt{p}_i$'s would naturally correspond.

3) Binary Triangle-Combined Descriptor: The Binary Triangle-Combined (BTC) descriptor integrates the strengths of both binary and triangle descriptors. The BTC descriptor consists of a triangle descriptor and three binary descriptors corresponding to the three keypoints (i.e., vertices) that form the triangle. This combination allows the triangle descriptor to provide pose invariance and correspondence between vertices, while the binary descriptor offers local point cloud distribution further enhancing the matching accuracy. The BTC descriptor, as shown in Data Structure 3 and visually illustrated in Fig. 6, comprises the following elements:

- $\mathtt{T}$: a triangle descriptor.
- $\mathtt{B}_1$, $\mathtt{B}_2$, $\mathtt{B}_3$: three binary descriptors corresponding to three vertices of the triangle descriptor.
- $\mathtt{f}$: the ID of the submap.

To build the BTC descriptors of a submap, we begin by constructing a 3-dimensional k-D tree using the 3D locations



(a) Voxel size = 1m

(b) Voxel size = 2m

Fig. 3: Comparison of all merged planes and the selected M (M = 3) reference planes, at different voxel sizes. The left subfigures show the results of all merged planes (colored by plane normals, whereas non-plane points are depicted in yellow), and the right subfigures display the selected M reference planes (with size determined by the largest and second largest eigenvalues of the covariance matrix Σ_m of the merged planes). (a) Voxel size = 1m. (b) Voxel size = 2m. As can be seen, despite the different voxel sizes, the selected reference planes are the same.

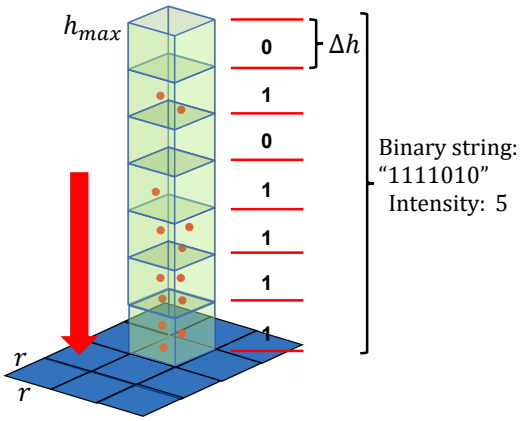


Fig. 4: Height encoding of a pixel on the reference plane with pixel resolution r. Points above the pixel are divided into m layers, with each layer being encoded as a '1' if it contains any points and '0' otherwise, leading to a binary string "1111010". The pixel intensity "5" is the bit sum of the binary string.

of the extracted keypoints. For each keypoint, we query the k-D tree to find its K nearest neighbors (e.g., $K = 10$), which gives us $C(K, 3)$ candidate triangles for each keypoint. The subsequent stage involves filtering this set of candidate triangles based on a series of selection criteria as follows.

Firstly, we discard any triangles that have two identical side lengths. This is because such triangles could cause ambiguity in vertices correspondence; a triangle where two sides are of the same length does not have a unique order of vertices for the subsequent binary descriptor matching or pose estimation.

Secondly, we discard triangles with side lengths that fall outside a specified range. Specifically, we eliminate any triangles with side lengths smaller than a lower bound $l_{\min}$ or larger than an upper bound $l_{\max}$. In our implementation, we set $l_{\min} = 2$ m and $l_{\max} = 30$ m. The reason for this is twofold: On the one hand, very small triangles represent only local spatial relations and thus provide limited information about the broader scene's appearance. On the other hand, very large triangles are less likely to be matched when the two submaps have small overlaps, reducing the effectiveness of the BTC descriptors in such scenarios.

Finally, we remove redundant triangles. As an illustrative example, consider three keypoints, denoted as $\mathtt{p}_1$, $\mathtt{p}_2$, and $\mathtt{p}_3$ in Fig. 6. These keypoints might all find each other within their K nearest neighbors, and each would consequently generate a triangle corresponding to the triangle formed by the same vertices $\mathtt{p}_1$, $\mathtt{p}_2$, and $\mathtt{p}_3$. We eliminate such redundant triangles by checking each new triangle against previous triangles of the submap. If the new one has the same set of vertices (irrespective of the order), it is discarded.

The filtered set of candidate triangles is finally used to construct the BTC descriptors of the submap. For each triangle, we construct a BTC descriptor, where the three side lengths of the triangle form the triangle descriptor component, and the three vertices and corresponding binary strings form the binary descriptor component.

V. Database Construction

This section introduces the design and implementation of our database system, which is optimized for efficient storage and querying of the BTC descriptors.

Data Structure 1: Binary Descriptor

```
struct BinaryDescriptor{
    std::bitset<m> b;
    std::array<double, 3> p;};
```

Data Structure 2: Triangle Descriptor

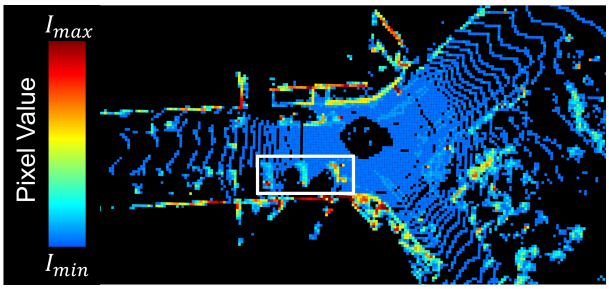
```
struct TriangleDescriptor{
    double l1,l2,l3;};
```

Efficient management of a large volume of descriptors is crucial for efficient loop detection. Therefore, we opt for a Hash table as the data structure for descriptor storage and retrieval. Hash tables offer significant advantages over other popular data structures, such as k-D trees commonly used in existing loop detection systems [8, 9], as detailed below.

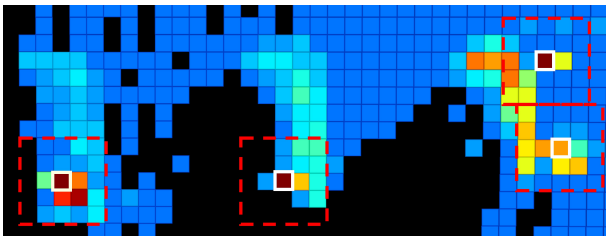
The first advantage is scalability. In large-scale environments, the descriptors from submaps can reach a substantial quantity; adding descriptors of a new submap will trigger a reconstruction of the k-D tree containing all descriptors in the past, leading to considerable construction time that is also linearly growing with time. Hash tables, on the other hand, demonstrate exceptional scalability and adaptability, offering constant time complexity $O(n)$ for both inserting and retrieving n descriptors in the new submap. They can handle dynamic datasets effectively; as new descriptors are generated, they can be inserted into the Hash table efficiently without the need for rebalancing in tree-based structures.

The second advantage stems from the structure of our triangle descriptors. They allow us to construct a straightforward yet effective Hash function, which computes the Hash key from the quantized side lengths of a triangle. Specifically, to quantize the side lengths of a triangle $\mathbf{L} = [\mathtt{l}_1, , \mathtt{l}_2, \mathtt{l}_3]^T$, we

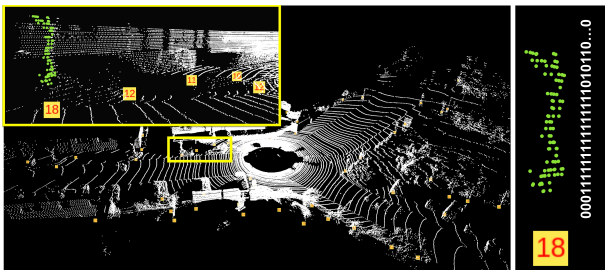
(a) The reference plane. All points are projected to this plane.



(b) Height-encode image generated on the reference plane.



(c) Keypoints extraction process.



(d) Extracted keypoints in the submap.

Fig. 5: Keypoints extraction: (a) The reference plane for height-encode image generation. (b) The generated height-encoded image, each pixel encoding the points distribution above it. (c) A zoomed-in region of a white square in (b) illustrates the process of detecting local maxima (white squares) in a 5×5 windows (red squares) and keypoints are generated at the corresponding pixel locations. (d) The extracted keypoints in the submap. keypoints are represented by yellow squares. The red number within each yellow square denotes the pixel intensity. The right sub-figure shows the points distribution above an extracted keypoint and its corresponding binary string.

employ a fixed resolution Δl and calculate their quantized side lengths $\bar{\mathtt{l}}_1$, $\bar{\mathtt{l}}_2$, and $\bar{\mathtt{l}}_3$ as follows:

$$\bar{\mathtt{l}}_1 = \mathtt{round}\left(\frac{\mathtt{l}_1}{\Delta l}\right), \bar{\mathtt{l}}_2 = \mathtt{round}\left(\frac{\mathtt{l}_2}{\Delta l}\right), \bar{\mathtt{l}}_3 = \mathtt{round}\left(\frac{\mathtt{l}_3}{\Delta l}\right). \tag{4}$$

The Hash function $\mathtt{Hash}(\mathbf{L})$ uses these quantized side lengths as input and calculates the Hash key:

$$\begin{aligned}\mathtt{Hash}(\mathbf{L}) &= \mathtt{Hash}(\mathtt{l}_1, \mathtt{l}_2, \mathtt{l}_3) = \mathtt{Int\_Hash}(\bar{\mathtt{l}}_1, \bar{\mathtt{l}}_2, \bar{\mathtt{l}}_3) \\ &= \mathtt{Mod}\left([\mathtt{Mod}\left((\bar{\mathtt{l}}_3 \cdot p + \bar{\mathtt{l}}_2) \cdot p, B\right)] + \bar{\mathtt{l}}_1, B\right)\end{aligned} \tag{5}$$

where p is a large prime number chosen to minimize the chance of Hash collisions by uniformly distributing the keys across the Hash table, and B is a maximum value set to prevent out-of-bounds indices. The design of our Hash function draws inspiration from the voxel Hashing technique [55].

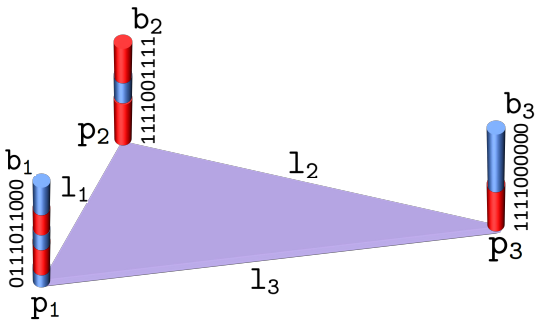


Fig. 6: Illustration of the Binary Triangle-Combined descriptor.

Data Structure 3: Binary Triangle-Combined (BTC) Descriptor

```
struct BtcDescriptor{
    TriangleDescriptor T;
    BinaryDescriptor B1,B2,B3;
    int f;};
```

In our Hash table, each Hash key maps to a container that saves the corresponding BTC descriptors. Descriptors in the same container mean that they have similar side lengths (i.e., similar triangle shapes). This organization allows us to efficiently group triangle descriptors of similar shapes.

In summary, our database system exhibits several key advantages. Firstly, the efficiency of our Hash function allows for rapid storage and retrieval of BTC descriptors, which is critical when dealing with large-scale environments. Secondly, the scalability of our system is ensured through the mapping of Hash keys to containers, allowing us to effectively handle increasing data volumes without significant degradation in performance. Lastly, given the finite range of possible triangle side lengths, we have a limited number of potential Hash keys. This constraint results in a predictable and compact Hash table size, further enhancing the system's efficiency. Together, these features improve the robustness and reliability of our system, making it an ideal solution for large-scale loop detection tasks.

VI. Loop Detection

In this section, we describe how to perform loop detection and relative pose estimation based on the BTC descriptors.

A. Selection of Candidate Submaps

For a querying submap, we first extract all its BTC descriptors following the method we detailed in Sec IV-C. For each BTC descriptor $\mathtt{C}_q$, we query the descriptor database (e.g., built from previous submaps) by calculating its Hash key as in (5). This allows us to locate it in the corresponding container within the Hash table.

Next, we carry out a binary descriptor similarity check for all candidate descriptors $\mathcal{C} = [\cdots, \mathtt{C}_c, \cdots]$ within the container to reject possible outliers. Specifically, for a candidate descriptor $\mathtt{C}_c$, if it is a true match of the query descriptor $\mathtt{C}_q$, its triangle vertices should have similar binary descriptors.

Moreover, since the side lengths are stored in a unique ascending order, the binary descriptors $(\mathtt{B}_1^q, \mathtt{B}_2^q, \mathtt{B}_3^q)$ of the query descriptor and $(\mathtt{B}_1^c, \mathtt{B}_2^c, \mathtt{B}_3^c)$ of the candidate descriptor naturally correspond with each other. Therefore, the binary similarity of this pair of descriptors can be evaluated as follows:

$$\begin{aligned} \mathtt{SimBTC}(\mathtt{C}_q, \mathtt{C}_c) &= \frac{1}{3}\sum_{i=1}^{3} \mathtt{SimBin}(\mathtt{B}_k^q, \mathtt{B}_k^c), \\ \mathtt{SimBin}(\mathtt{B}_k^q, \mathtt{B}_k^c) &= \frac{2 \cdot \mathtt{HW}(\mathtt{b}_k^q \ \& \ \mathtt{b}_k^c)}{\mathtt{HW}(\mathtt{b}_k^q) + \mathtt{HW}(\mathtt{b}_k^c)}. \end{aligned} \tag{6}$$

where $\mathtt{b}$ denotes binary string, '&' denotes the bitwise AND operation of two binary strings, and $\mathtt{HW}$ stands for the Hamming Weight [56], which counts the number of '1's in the binary string, i.e., the number of set bits. The binary descriptor similarity $\mathtt{SimBin}$ is essentially the normalized sum of the shared '1' bits between two binary strings.

If the binary similarity (6) between the current BTC descriptor $\mathtt{C}_q$ and the candidate descriptor $\mathtt{C}_c$ is above a threshold σ_s, we regard the triangle match to be correct and cast one vote for the submap ID $\mathtt{f}$ of the candidate descriptor $\mathtt{C}_c$. We enumerate all BTC descriptors of the querying submap, and for each BTC descriptor, we cast votes for all candidate descriptors in the corresponding container whose binary descriptors match with the query one. Finally, the submap IDs with the top κ votes are saved for use in the fine loop detection step.

Fig. 7 illustrates our approach to descriptor querying and matching using a Hash table, as well as outlier rejection via binary similarity checking.

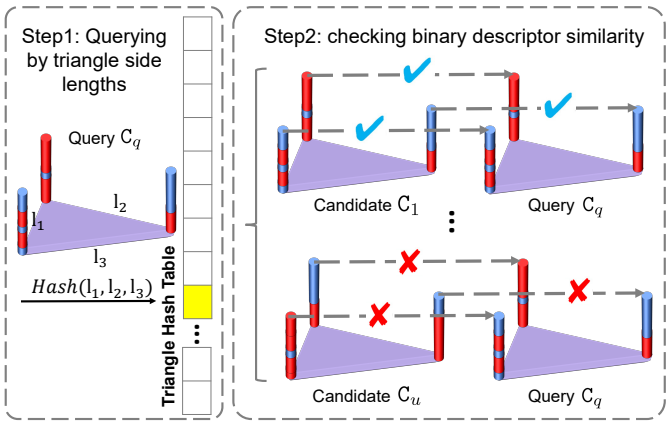


Fig. 7: Illustration of the descriptor querying and matching process in two steps. Step 1 (left dashed box) shows the input BTC descriptor and the Hash key computation to locate the container. Step 2 (right dashed box) demonstrates the binary similarity checking between the querying descriptor and each candidate descriptor, using color columns to visually represent the binary string at each vertex, with red indicating 1 and blue indicating 0.

B. Geometrical Verification

After identifying the potential κ loop candidate submaps through the descriptor voting process, we perform geometrical verification on each of them in sequence. This step is designed to filter out false detections that primarily arise from local areas with similar overall appearance.

As mentioned in Section VI-A, the vertices correspondence between binary descriptors in a BTC descriptor pair have been established during the descriptor matching process. Specifically, for a query descriptor $\mathtt{C}_q$ and its corresponding candidate descriptor $\mathtt{C}_c$, each binary descriptor of the triangle vertices in $\mathtt{C}_q$ naturally corresponds to a binary descriptor in $\mathtt{C}_c$. Utilizing this vertices correspondence, we can efficiently compute the relative transformation ${}_Q^C\mathbf{T} = ({}_Q^C\mathbf{R}, {}_Q^C\mathbf{t}) \in SE(3)$ between the querying submap and candidate submap using Singular Value Decomposition (SVD):

$$\begin{aligned} &\mathbf{H} = \sum\nolimits_{i=1}^{3} (\mathtt{p}_i^q - \bar{\mathbf{p}}^q)(\mathtt{p}_i^c - \bar{\mathbf{p}}^c), \\ &\bar{\mathbf{p}}^q = \frac{1}{3}\sum\nolimits_{i=1}^{3} \mathtt{p}_i^q, \quad \bar{\mathbf{p}}^c = \frac{1}{3}\sum\nolimits_{i=1}^{3} \mathtt{p}_i^c, \\ &[\mathbf{U}, \ \mathbf{S}, \ \mathbf{V}] = \mathtt{SVD}(\mathbf{H}), \\ &{}_Q^C\mathbf{R} = \mathbf{V} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & \det(\mathbf{V}\mathbf{U}^T) \end{bmatrix} \mathbf{U}^T, \\ &{}_Q^C\mathbf{t} = -{}_Q^C\mathbf{R} * \bar{\mathbf{p}}^q + \bar{\mathbf{p}}^c. \end{aligned} \tag{7}$$

Here, this transformation, ${}_Q^C\mathbf{T} = ({}_Q^C\mathbf{R}, {}_Q^C\mathbf{t})$, when applied to the triangle vertices in the querying submap expressed in the submap local coordinate frame, aligns them directly to the candidate submap. To enhance the robustness, we employ RANSAC[57] to find the transformation that maximizes the number of correctly matched vertices. We term this transformation as a rough transformation and denote it as ${}_Q^C\mathbf{T}_r = ({}_Q^C\mathbf{R}_r, {}_Q^C\mathbf{t}_r)$.

With the rough transformation ${}_Q^C\mathbf{T}_r$, we calculate the plane overlap between the querying submap and the candidate submap for geometrical verification. Denote the plane list of the querying submap be $\Pi^q = [\cdots, (\bar{\mathbf{p}}_i^q, \mathbf{u}_{3i}^q), \cdots]$ extracted from Sec IV-B1, and the plane list of the candidate submap be $\Pi^c = [\cdots, (\bar{\mathbf{p}}_j^c, \mathbf{u}_{3j}^c), \cdots]$. We construct a three-dimensional k-D tree with the center points $[\cdots, \bar{\mathbf{p}}_j^c, \cdots]$ of the candidate submap planes Π^c. Then, for each plane center point $\bar{\mathbf{p}}_i^q \in \Pi^q$, we first transform $\bar{\mathbf{p}}_i^q$ using the transformation ${}_Q^C\mathbf{T}_r$, and then search for the nearest point $\bar{\mathbf{p}}_j^c$ in the k-D tree. We judge whether the two planes coincide by examining the plane distance and difference in normal vectors:

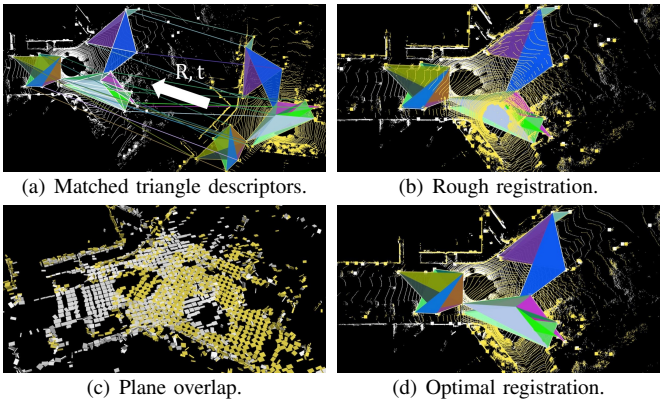
$$\begin{aligned} &\max\left((\mathbf{u}_{3j}^c)^T({}_Q^C\mathbf{T}_r\bar{\mathbf{p}}_i^q - \bar{\mathbf{p}}_j^c), (\mathbf{u}_{3i}^q)^T({}_Q^C\mathbf{T}_r^T\bar{\mathbf{p}}_j^c - \bar{\mathbf{p}}_i^q)\right) \le \gamma_d \\ &\arccos({}_Q^C\mathbf{R}_r\mathbf{u}_{3i}^q \cdot \mathbf{u}_{3j}^c) \le \gamma_u \end{aligned} \tag{8}$$

where γ_d and γ_u are preset hyperparameters, slightly larger than σ_d and σ_u used in the merging criteria equation (2), to determine whether planes in the two submap overlap. If a pair of planes satisfy the plane distance and normal constraints in equation (8), the pair of planes is considered to be overlapping. After checking all planes of the querying frame, we calculate the percent of plane overlap (N_c) between the querying submap and the candidate submap:

$$N_c = \frac{N_{\mathtt{overlap}}}{N_{\mathtt{total}}} \times 100 \tag{9}$$

where $N_{\mathtt{overlap}}$ is the number of overlapping planes and $N_{\mathtt{total}}$ is the total number of planes in the querying submap.

For each candidate submap, we determine whether the computed percept of plane overlap exceeds a certain threshold



(a) Matched triangle descriptors. (b) Rough registration.

(c) Plane overlap. (d) Optimal registration.

Fig. 8: Example of loop detection and correction using BTC descriptors. (a) Loop detection with 45° rotation and $50m$ translation error between submap 0 (white) and submap 4405 (yellow) of KITTI00. Matched descriptors are shown in the same color and connected with three colored lines. (b) Registration result after transforming the yellow point cloud to the white point cloud using the rough transformation ${}^{C}_{Q}\mathbf{T}_r$ obtained from (7). (c) Schematic representation of the plane overlap ratio between the two point clouds, with the plane overlap in this case being 58.9%. (d) The registration result using the fine transformation ${}^{C}_{Q}\mathbf{T}_f$.

σ_{p}. If it does, the loop detection for this candidate submap is considered valid. In cases where multiple candidate submaps pass the geometrical verification, we select the one with the highest percent of plane overlap N_c as the best candidate.

By performing this geometrical verification, we significantly reduce the chance of false loop detections, thus improving the accuracy of our system. Moreover, the efficiency of our geometric verification surpasses ICP-based methods used in existing loop detection works [8, 10] since the number of planes is considerably lower than the number of points.

With the best candidate submap and its rough pose transformation ${}^{C}_{Q}\mathbf{T}_r$, we further refine it by minimizing the plane distance and normal vector difference as defined in equation (8) using Ceres-Solver [58]. The optimized pose is termed as the fine transformation and is denoted as ${}^{C}_{Q}\mathbf{T}_f$. We illustrate the loop detection process between two submaps in Figure 8.

Finally, after performing loop detection, we add the descriptors of the new submap to the Hash table database by appending each descriptor to its corresponding container.

VII. Benchmark Setup

In this section, we first introduce the datasets utilized for benchmark evaluation. Next, we discuss the criteria for experiments and provide an overview of the implementation details for each method in comparison.

A. Datasets for Evaluation

We have selected various datasets detailed below to thoroughly evaluate the performance of our proposed method under different conditions. The datasets are collected by different types of LiDAR sensors and in diversified environments.

1) KITTI: The KITTI Odometry dataset [16] is acquired at 10 Hz using a 64-beam spinning LiDAR (Velodyne HDL-64E) mounted on the rooftop of a car moving in urban road environments. It contains six sequences with loops (i.e., sequence 00,02,05,06,07,08), which are all utilized for evaluation. Since the ground-truth poses provided by the KITTI Odometry dataset are not very accurate to determine the loop ground-truth, we use a global hybrid LiDAR bound adjustment method [59] to obtain the optimum pose estimate for all these six sequences and use them as the ground-truth pose.

2) NCLT: The NCLT dataset [18] offers long-term measurements captured on different days or seasons along similar routes in a campus environment. The NCLT dataset is collected at 10 Hz using a 32-beam spinning LiDAR (Velodyne HDL-32E) mounted on a Segway mobile platform. Four sets of sequences are selected according to the number of loops and seasonal diversity, 2012-05-26, 2012-08-20, 2012-09-28, and 2013-04-05, marked as NCLT1 to NCLT4, respectively.

3) Complex Urban Dataset: The Complex Urban Dataset [60] covers various complex urban environments, characterized by numerous moving objects and high-rise buildings. Considering its complexity and number of loops, we select five sequences for evaluation: Campus00, Urban04, Urban09, Urban10, and Urban16, labeled as CU1, CU2, CU3, CU4, and CU5, respectively.

4) Wild-Places: The Wild-Places dataset [19] is a challenging large-scale dataset designed for LiDAR place recognition in unstructured, natural environments. It comprises eight sequences captured using a handheld sensor over fourteen months. We use all eight sequences for both inter-sequence and intra-sequence (i.e., multi-session) evaluations, marked as K-01 to K-04 and V-01 to V-04, respectively. These datasets are collected by a 16-beam rotating LiDAR (Velodyne VLP-16) at 20 Hz. The raw scans are then locally registered into submaps, leading to a total of 67K undistorted submaps, each with an accurate 6DoF ground-truth pose. Given that the data is captured by a handheld device moving at a slow pace, we sample submaps with a 1-meter distance to avoid redundancy.

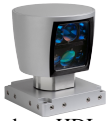
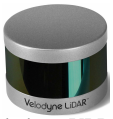
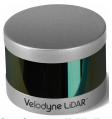
5) Livox Dataset: The aforementioned datasets all utilize spinning LiDARs leading to 360° horizontal Fied of View (FoV). To verify the adaptability of our method to different LiDAR types, we have also collected solid-state LiDAR datasets. We employ Livox series LiDARs (Horizon and Avia) and collect data in urban, campus, and unstructured environments. For urban environments, we select the open-source Ka_Urban_East sequence from the Lili-OM[61], referred to as HORIZON1. For campus environments, we provide two datasets of varying lengths with loops, collected at the campus of SUSTech, labeled as HORIZON2 and HORIZON3. For unstructured environments, we provide two datasets collected with a handheld device in parks and two datasets collected with a drone above mountains, labeled AVIA1 to AVIA4.

In summary, we have used a total number of 25 sequences in our evaluation. A comprehensive overview of the key characteristics of the datasets is shown in TABLE I. The diverse datasets enable us to thoroughly assess the performance, robustness, and adaptability of our proposed method across a wide range of scenarios.

B. Comparison Methods

We compare the proposed methods against four existing methods: Scan Context [8], M2DP [39], NDT [34], BoW3D

TABLE I: Datasets for benchmark evaluation.

Dataset	KITTI	NCLT	Complex Urban Dataset	Wild Place	Livox Dataset	
Platform						
LiDAR	 Velodyne HDL-64E	 Velodyne HDL-32E	 Velodyne VLP-16	 Velodyne VLP-16	 Livox Horizon	 Livox Avia
Scanning Mechanism	spinning 64-line	spinning 32-line	spinning 16-line	spinning 16-line	Risley prism	
Field of View	360.0° × 26.8°	360.0° × 41.3°	360.0° × 30°	360.0° × 30°	120° × 40°	70.4° × 77.2°
No. of Sequences	6	4	5	8	7	
Environment	Urban	Campus	Urban, Campus	Forest	Urban, Campus, Unstructured	

[48] and FreSCo [45]. Among them, M2DP, NDT, BoW3D, and BTC have the capability to operate on accumulated point cloud submaps. Therefore, we prepare submaps for these datasets by accumulating consecutive n_s LiDAR scans using the ground-truth poses in each dataset. Specifically, for the KITTI and Wild-Places datasets, additional accumulation is not necessary since each LiDAR scan in KITTI is already sufficiently dense (so a single scan is used as a submap), and the Wild-Places dataset already provides accumulated dense submaps. For NCLT and Livox datasets, we accumulate every $n_s = 10$ consecutive LiDAR scans to one submap. For the Complex Urban dataset, it employs two 16-beam spinning LiDARs (Velodyne VLP16) installed on the left and right side of the vehicle. We merge the point clouds from both LiDARs into a single scan and then accumulate every $n_s = 10$ consecutive scans to one submap. The submaps are directly fed to M2DP, NDT, BoW3D, and BTC for evaluation. The default implementation of Scan Context accepts individual scans as its inputs. To enable a fair comparison, we also run it on submaps. To do so, we reproject an accumulated submap back to the middle scan to mimic a denser "scan" measured at this pose. For all loop detection methods, we excluded the 100 most recent submaps in loop detection, avoiding the detection of loops that are too close in time. All methods are evaluated on the same computing platform with implementation details as follows.

1) Scan Context: We utilize the open-source C++ implementation[2] of Scan Context. To reproduce the results reported in the original Scan Context paper, we keep the original implementation to our best efforts. Specifically, we enable descriptor augmentation and apply a 0.5 m^3-resolution voxel downsampling to the input point cloud submap. For the parameter settings, we set the candidate number to 50 and use the default configurations for the other parameters as provided in their open-source implementation. As the open-source version of Scan Context is incompatible with solid-state LiDAR, so its evaluation on the Livox dataset is omitted.

2) M2DP: The open-source code for M2DP[3] is provided in MATLAB. To ensure a fair comparison in terms of computational efficiency, we replicate the algorithm in C++, following the MATLAB code provided by the original authors. To best reproduce the original M2DP implementation, we adhered to the same configuration provided in their open-source code, except for incorporating a 0.5 m^3 point cloud downsampling before descriptor calculation, which we find further improve the efficiency of M2DP from their original implementation.

3) NDT: The authors of NDT [34] do not provide open-source code for their implementation. Thus, we re-implement the algorithm in C++, based on the descriptions in the original publication, and open-source our re-implementation on GitHub[4] for the reproduction of our evaluated results. In line with the guidance of the original authors, we use submaps as the input, which leads to improved detection performance. Additionally, due to the high computational cost of the descriptor similarity calculation for NDT, we limit the NDT search range to 500m from the query submap, especially in larger such as Wild-Places. This step ensures the method maintains an acceptable computation time.

4) BoW3D: We employ the open-source C++ implementation[5] provided by BoW3D [48]. Link3D, the corner point extraction module in BoW3D, exhibits improved performance when the point cloud is denser. As such, we input submaps into the BoW3D algorithm in our evaluation. For parameter settings, we use the default configurations as provided in their open-source implementation. BoW3D is only adapted to the KITTI dataset, so we only evaluate it on KITTI.

5) FreSCo: FreSCo [45] first pre-processes the original LiDAR point cloud of a single scan to obtain BEV (Bird's Eye View) images. Subsequently, descriptors are extracted from these BEV images for loop detection. We directly use the MATLAB implementation[6] provied by FreSCo [45], where the pre-processing part is implemented using C++[7] while the rest

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

[3]https://github.com/LiHeUA/M2DP
[4]https://github.com/ChongjianYUAN/ndt_loop
[5]https://github.com/YungeCui/BoW3D
[6]https://github.com/soytony/FreSCo
[7]https://github.com/soytony/Point-Cloud-Preprocessing-Tools

of descriptor extraction and loop detection are implemented in MATLAB. It's noteworthy that the point cloud preprocessing module of FreSCo is only designed for single-scan point clouds, so only it's results on the KITTI dataset are included in our evaluation to ensure a fair comparison. For parameter setting, we use the defaults configurations in the open-source implementation, which is specifically tailored for KITTI.

6) BTC: Our proposed BTC is implemented in C++ with parameters presented in TABLE II. While we've listed all parameters to provide a comprehensive understanding of our method, it should be noted that only a couple of them, specifically κ (number of candidates) and M (number of reference planes), needs to be tuned with little efforts. The parameter κ holds significance as it affects the selection of loop submap candidates, thereby influencing the balance between recall rate and computational efficiency. The parameter M determines the BTC's adaptability to diverse terrains. In our experimental evaluations, only the parameter M is slightly adjusted among different datasets. Specifically, for the AVIA1 to AVIA4 datasets, due to the very uneven terrains and the absence of (or very small) ground areas, we use two reference planes (i.e., $M = 2$). For all other dataset sequences, we use one reference plane (i.e., $M = 1$). Except for this minor adjustment, all rest parameters remain identical across all sequences, showcasing their robustness and adaptability.

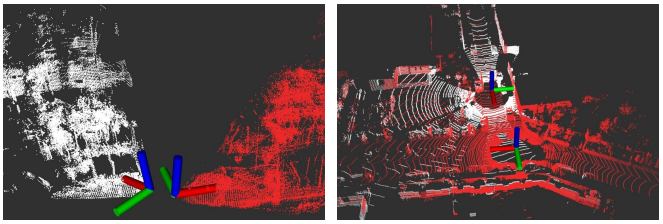
TABLE II: Parameters for BTC.

Parameter	Value	Description
n_s	1,10	scans number to form a submap
ΔL	$2m$	voxel size for plane detection
σ_1, σ_2	0.01, 0.05	threshold for plane determination criteria
σ_d, σ_u	0.3m, 20°	threshold for plane merging criteria
M	$1, 2$	number of reference planes
r	0.5m	pixel resolution for height-encode image
$h_{\max}$	5m	maximum height for binary encoding
Δh	0.1m	height resolution for binary encoding
σ_I	10	intensity threshold for keypoint selection
K	10	Nearest keypoints No. for triangle formation
$l_{\min}, l_{\max}$	2m, 30m	valid triangle side lengths range
Δl	0.2m	side length quantization resolution
p	116101	prime number for Hash key distribution
B	10^{10}	maximum value for Hash key range
σ_s	0.7	threshold for binary similarity checking
κ	50	No. of loop submap candidates
γ_d, γ_u	0.5m, 30°	threshold for plane overlap criteria
σ_{p}	0.5	threshold for geometrical verification

C. Evaluation Criteria

In order to evaluate the performance of all loop detection methods, we rely on precision and recall metrics. Precision is defined as the ratio of true positives (TP) over all predicted positives, which is the sum of true positives and false positives (FP), i.e., Precision = TP / (TP + FP). The recall is defined as the ratio of true positives over the actual positives (i.e., ground-truth loops (GT)), i.e., Recall = TP / GT.

Determination of true positives (TP) and false positives (FP) of a loop submap predicted by a method can be a complex task in loop detection scenarios. Many existing methods [8, 10, 39, 48] rely on a distance-based criterion, where if the ground-truth pose distance between the query and the predicted loop submap is less than a threshold (e.g., 4m), the detection is considered a true positive (TP), or else false positive (FP).



(a) AVIA01 submap 575–31. (b) KITTI00 submap 3250–2338.

Fig. 9: Examples illustrating the problem of distance-based criterion for true/false positives determination. (a) Two submaps observing completely different areas have a small distance (3.5 m with no point cloud overlap); (b) Two submaps sharing large co-visible areas have a large distance (18.3 m with 51% cloud overlap).

Determining the true or false positives based on distances suffers from fundamental limitations. For LiDARs with a frontal FoV (e.g., Livox LiDARs), the sensors could be closely positioned but face opposite directions, observing completely different areas of the scene (see Fig. 9(a). In such cases, using the distance-based criterion will identify incorrect true positives. On the other hand, LiDAR's long-range capability can result in situations where the submap distance is large, yet the point cloud share substantial co-visible areas (see Fig. 9(b)). In this case, the distance-based criterion will identify incorrect false positives.

To address the above issue, we adopted an overlap-based criterion similar to that used in OverlapNet [54]. Specifically, for each querying submap, if the predicted loop submap shares an overlap over 50%, it is viewed as a true positive (TP); Otherwise, it is viewed as a false positive (TP). When calculating the overlap between two submaps, we use the ground-truth pose to register them to the same frame. The registered point cloud is then split into voxels of size 0.5m. Afterward, the overlap is determined as the ratio of the number of voxels containing points from both submaps over the number of voxels containing points. The overlap-based criterion offers a more sensible evaluation metric by measuring the degree of overlap between point clouds from the querying submap and loop submap.

Finally, to determine the number of actual positives (or ground-truth loops), we adopt the following procedure. For each submap, we conduct a search in all previous submaps except the 100 most recent ones to avoid detecting temporally-close loops (as we did for all loop detection methods). Utilizing the ground-truth poses, we compute the overlap between the current submap and each of the previous submap. If any previous submap shares more than 50% overlap with the current submap, it contributes an effective loop, and the number of actual positives, GT, is increased by one.

VIII. Becnhmark Results

A. Precision Recall Evaluation

1) Single-session Evaluation: In this experiment, we aim to evaluate the performance of all benchmarked loop detection methods in terms of precision-recall curves, following the evaluation criteria outlined in Section VII-C. To generate the precision-recall curve, we vary the threshold σ_{p} used in the geometrical verification process (Section VI-B) and obtain a

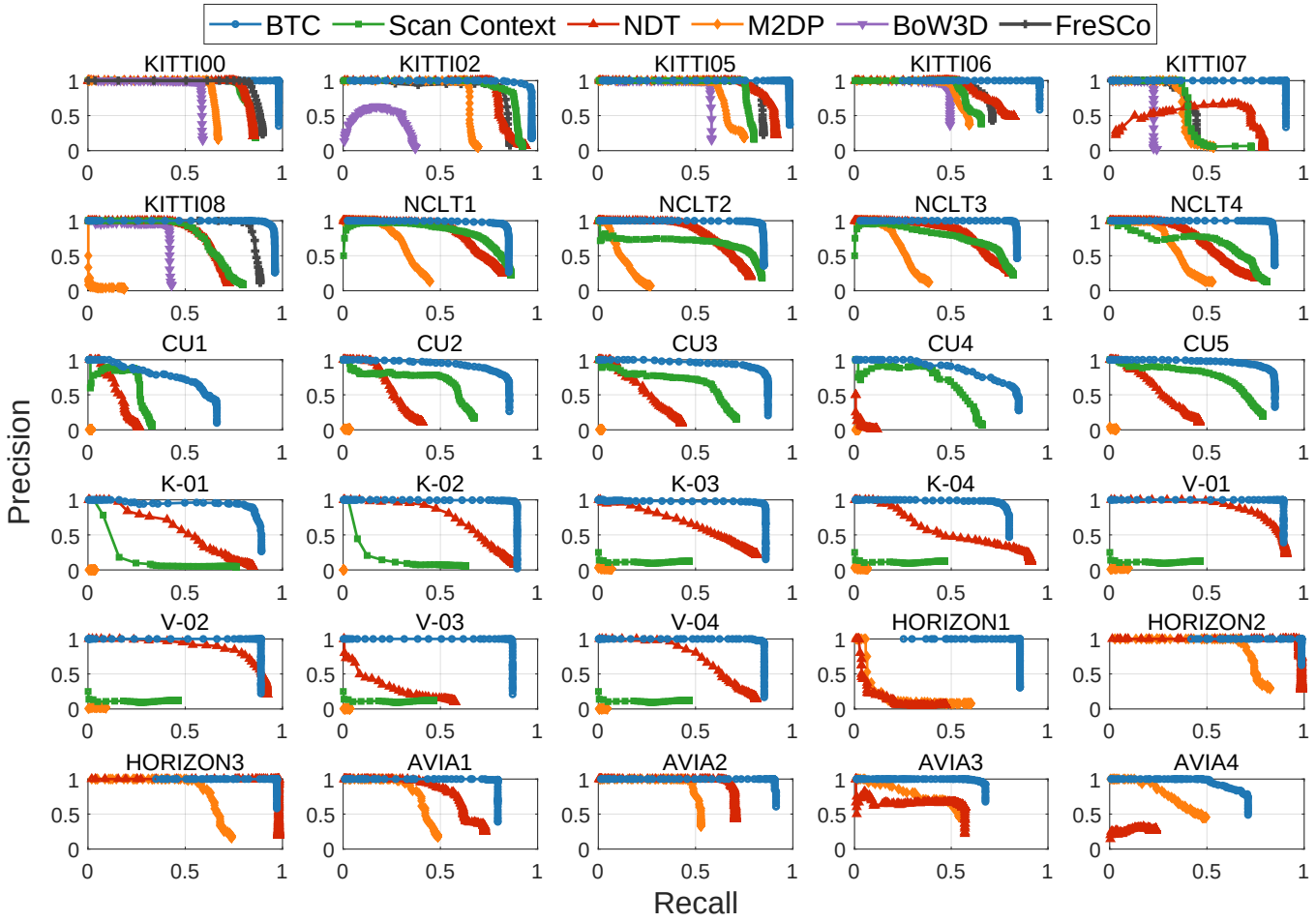


Fig. 10: Single-session evaluation of all benchmark methods on 30 sequences. Each subfigure corresponds to a different sequence, showcasing the precision-recall performance of each method. The results show that BTC consistently outperforms the other methods across all sequences, demonstrating its robustness and superior adaptability to various types of data and environments. Each subfigure corresponds to a different sequence, illustrating the precision-recall performance of all the benchmark methods.

set of precision and recall pairs at different threshold levels. For other comparison methods, we follow a similar approach by varying the threshold of key parameters to generate their respective precision-recall curves.

The precision-recall curves of all methods on the 30 sequences are plotted in Fig. 10. As can be seen, our proposed method consistently outperforms the other four methods across all datasets. In contrast, other methods exhibit large performance variations in different scenes. Scan Context achieves satisfactory performance in urban and campus environments (as shown in KITTI and NCLT datasets, respectively), but its performance significantly declines in Wild Place forest datasets. This performance degradation can be primarily attributed to the 2.5D image descriptor employed by Scan Context, which is generated through BEV (bird's-eye view) spatial partitioning and maximum height encoding that lack of sufficient discriminativeness in forest scenes. For the Livox dataset, M2DP and NDT demonstrate good results in short-term campus sequences (e.g., HORIZON2, HORIZON3); however, their performances drop considerably in long-term sequence (HORIZON1), unstructured (AVIA1 AVIA4), urban (KITTI) or campus (NLCT) datasets. FreSco demonstrates superior performance than other comparitive methods on the KITTI dataset, but is still outperformed by BTC in terms of both accuracy and recall with big margins. Overall, our BTC method exhibits much higher recall rates while maintaining superior precision even in complex unstructured scenes and in environments with numerous dynamic obstacles and significant occlusions (e.g., the Complex Urban dataset).

For Scan Context and BoW3D, their recall rates on the KITTI dataset are lower than the results presented in the original paper [8, 48]. This discrepancy primarily stems from the different criteria used for determining the ground-truth loops (GT). Scan Context [8] used the distance-based criterion for determining the ground-truth loops (GT) as well as true or false positives where the distance threshold was 4m. BoW3D further reduces this threshold to 3m in their evaluation. Such distance threshold is much less than the LiDAR measuring range (up to hundreds of meters) and hence causes the ground-truth loops (GT) to be much smaller than the actual number, which causes a higher recall rate. In contrast, our overlap-based evaluation computes the overlap between querying submaps and loop submap candidates to determine the ground-truth loops as well as true and false positives. Due to the long LiDAR measuring range, the overlap of two submaps could still be large (e.g., more than 50%) even when the

two submaps are far apart, leading to a much higher number of ground-truth loops (GT). For instance, considering the KITTI00, 02, 05, and 08 sequences utilized in Scan Context [8], the ground-truth loops are 790, 309, 493, and 332 when applying the 4-m distance criterion. However, when using the overlap criterion, ground-truth loops are 1037, 358, 662, and 478, respectively. Furthermore, Scan Context [8] and BoW3D [48] exhibit weaker pose invariance, causing missing loop detection when the distance is large whilst the point cloud overlap is still over 50%. Such missing loop detection reduces the number of true positives (hence lower precision and recall) for these methods when using the overlap criteria.

The robustness of our BTC method is further highlighted through a series of challenging scenarios, where other benchmarked methods often struggle due to their limited pose invariance. These situations, selected from the benchmark dataset, are illustrated in Fig. 11:

- Reversed loops: As shown in Fig. 11 (a-b), our method successfully recognizes reversed loops in both KITTI08 and Livox sequence AVIA1. A reversed loop occurs when the vehicle revisits a place from an opposite direction, presenting a significantly different perspective of the environment. This situation poses a big challenge for methods without strong pose invariance.
- Loops with large rotation and translation: 11 (c-d) show loops under large rotation and translation from the KITTI08 sequence and the forest sequence V-04, respectively. In these scenarios, BTC effectively handles substantial changes in viewpoint.
- Loops with different revisit routes: As demonstrated in Fig. 11 (e-f), our method is able to recognize a loop with the same direction but different routes from the Livox sequence AVIA1. This type of loop can be challenging due to the partial overlap of point clouds between the first and revisited pass.
- Down-looking scenarios: In Fig. 11 (g-h), we highlight two cases from sequence AVIA3, which is collected by an airborne LiDAR flying at the height of 50 meters, creating a downward-looking view angle that is often not demonstrated in existing works [16, 18, 19]. The BTC method still performs well despite the natural, unstructured environments.

2) Multi-session Evaluation: In this experiment, we evaluate the multi-session capability of the proposed method by examining its ability to detect loops across different time intervals. We employ two benchmark datasets, the NCLT and Wild-Places datasets, which represent typical campus and forest environments, respectively. The datasets are collected in their respective environment at different times. For the NCLT dataset, the NCLT1 is used as the database, and the NCLT2, NCLT3, NCLT4 are selected as query sequences with time intervals of 3 months, 4 months, and 11 months, respectively. The Wild-Places dataset includes sequences from two distinct forest environments, Karawatha and Venman. For Karawatha, the K-01 is utilized as the database, and the K-02, K-03, and K-04 are selected as query sequences with time intervals of the same day, 6 months, and 14 months, respectively. The Venman sequences are handled in a similar manner.

For assessment, since both datasets do not offer precise multi-session trajectory alignment, we employ Hierarchical Bundle Adjustment (HBA) [59] to optimize all poses of all sessions, leading to a global consistent pose estimate across all sequences. Fig. 12 shows the global point cloud registered by this approach on the NCLT and WildPlace datasets. Then we use the estimated pose trajectory as the ground-truth pose to determine true positives and false positives (detailed in Section VII-C) for the multi-session evaluation. For the assessment criterion, we use the average precision (AP), which is the area under the precision-recall curve.

Fig. 13 shows the average precisions (APs) of BTC, Scan Context, NDT, and M2DP on the NCLT dataset, and BTC, Scan Context, and NDT on the Wild-Places dataset. As can be seen, while all methods' APs degrade as the time interval increases, our proposed method outperforms other methods consistently overall at all time intervals on all datasets. Notably, for the Wild-Places dataset, our method's AP at the longest time interval surpasses those of the other methods even at the shortest interval.

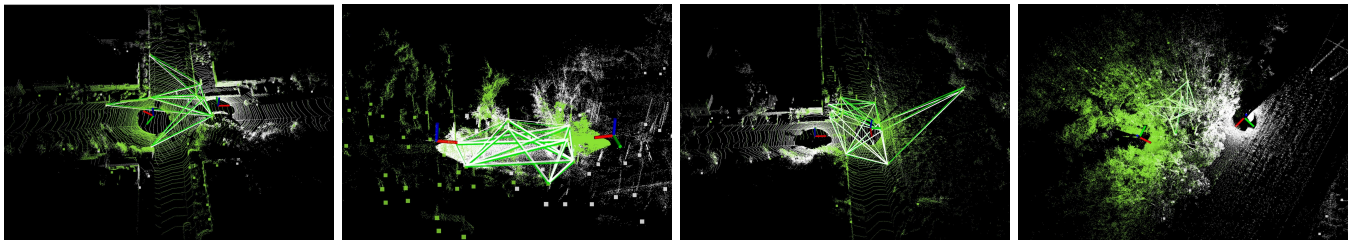
B. Runtime Evaluation

In this section, we compare the runtime of BTC with other methods. All experiments are carried out on the same system with an Intel i7-11700k @ 3.6 GHz and 64 GB memory. The implementation of all methods is described in Section VII-B. Since BoW3D is only compatible with the Velodyne 64-line LiDAR and ScanContext is only compatible with mechanical spinning LiDARs, the results will not be recorded for datasets that are not compatible with these two methods.

Table III presents the average descriptor extraction time, loop detection time (comprising candidate submap selection and geometrical verification, if performed), and total time (both descriptor extraction and loop detection, representing the total time duration of the loop detection process) for each method across different sequences, along with the path length, loop node number, total node number, and average points per submap for each dataset.

Unlike global description methods such as ScanContext, M2DP, NDT, and FreSCo, which utilize a single descriptor for a submap, BTC extracts multiple descriptors for each submap. This design allows each descriptor in the current query submap to search the database for similar descriptors and conduct binary similarity checks independently. Therefore, BTC employs multi-threading (four threads) in candidate submap search (Section VI-A), while the rest of the BTC process (including descriptor extraction in Section IV and geometrical verification in Section VI-B) operates in a single-threaded mode. For a comprehensive comparison, we've also presented results with all BTC steps executed using a single thread.

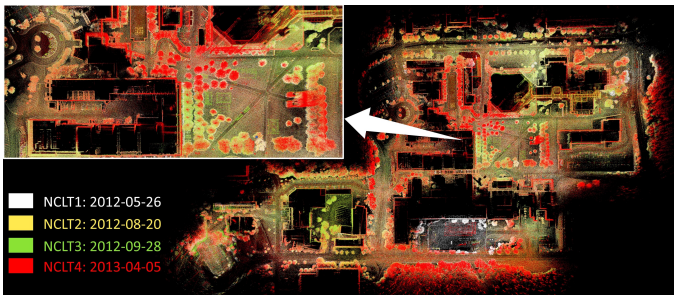
As can be seen, BTC consumes the least time for descriptor extraction in average and also on all individual sequences except KITTI, where its extraction time closely follows that of Scan Context. The primary reason for the efficient descriptor extraction in BTC is the efficient incremental plane merging



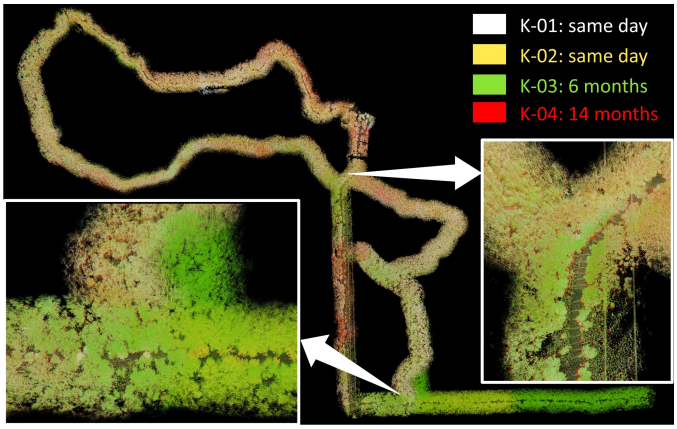
(a) diff $(\Delta_t, \Delta_\theta) : (1m, 163°)$ (b) diff $(\Delta_t, \Delta_\theta) : (27m, 174°)$ (c) diff $(\Delta_t, \Delta_\theta) : (17m, 148°)$ (d) diff $(\Delta_t, \Delta_\theta) : (25m, 108°)$

(e) diff $(\Delta_t, \Delta_\theta) : (7m, 11°)$ (f) diff $(\Delta_t, \Delta_\theta) : (6m, 6°)$ (g) diff $(\Delta_t, \Delta_\theta) : (7m, 94°)$ (h) diff $(\Delta_t, \Delta_\theta) : (1m, 93°)$

Fig. 11: Successful place recognition in challenging scenarios selected from the benchmark datasets. This figure presents eight subplots, each depicting a unique scenario. (a) and (b) show reversed loops from KITTI08 and AVIA1 sequences, respectively. (c) and (d) demonstrate loops with large rotation and translation from KITTI08 and forest sequence V-04, respectively. (e) and (f) present loops with the same direction but different routes from the Livox sequence AVIA1. Lastly, (g) and (h) display down-looking cases from sequence AVIA3, with a flight height of about 50 meters. In each subplot, the caption indicates the relative rotation and translation between the two submaps in a loop pair.



(a) NCLT Dataset



(b) Wild-Places Dataset

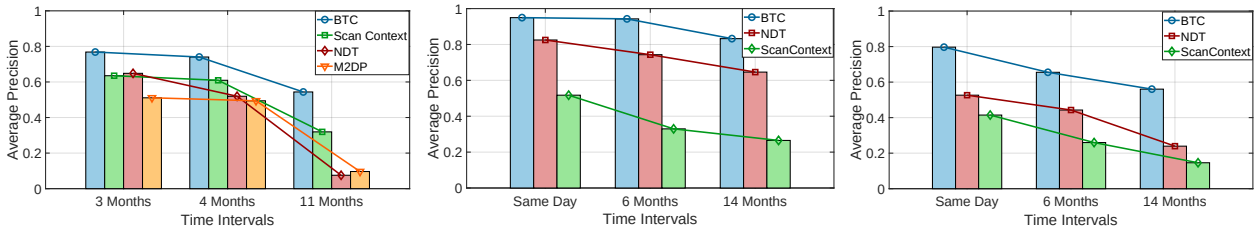
Fig. 12: Point cloud alignment results on the NCLT and Wild-Places datasets. The aligned poses are used to determine ground-truth loops (GT) and assess whether detected loops are true or false positives during multi-session evaluations.

for reference plane generation and the simple yet efficient keypoint extraction. FreSCo's descriptor extraction is most time consuming due to the requirement of a ground removal preprocessing step (implemented in C++) to generate a BEV (Bird's Eye View) image, which significantly lengthens its extraction time.

For loop detection, BTC achieves the second-fastest speed after M2DP when using multi-threading. Under single-threaded operation, BTC's performance is closely competitive with Scan Context. Fast query Hash tables are utilized to accelerate the query process, enabling faster similarity searches among a large number of descriptors and thus improving detection speed. Additionally, many incorrect descriptor matches are eliminated through binary similarity checking, resulting in a high ratio of correct keypoints correspondence that further speeds up RANSAC in the geometric verification step. Furthermore, our geometric verification step employs plane-to-plane registration, which is more efficient than ICP due to the smaller number of planes than points. Although M2DP has the fastest query time by directly using the L2 norm for rapid descriptor similarity comparison, its high descriptor extraction time results in overall high computation time. For Scan Context, numerous engineering optimizations (e.g., align key and partial comparison for distance calculations between two scan contexts) available in the open-source code but not described in the original paper have significantly reduced the actual computation time when compared to the original paper [8]. However, despite these optimizations and others (e.g., point downsample), the overall computation efficiency of Scan Context is still considerably lower than that of BTC. NDT exhibits the lowest efficiency, especially in large-scale scenes. This is because it requires comparing each individual descriptor in the database with the querying descriptor during retrieval. When the database contains a large number of descriptors, loop retrieval becomes increasingly time-consuming. In sum, BTC consumes the least total computation time in all the 30 sequences, both in multi-threaded and single-threaded environments.

C. Pose Estimation Evaluation

In this section, we evaluate the accuracy and efficiency of the relative pose estimation of the benchmark methods.



(a) NCLT (b) Venman (c) Karawatha

Fig. 13: Multi-session evaluation of different methods across varying time intervals on the NCLT and Wild-Places datasets. The bars represent the average precision for each method at different time intervals, while the lines with markers show the overall performance trend. The results suggest that BTC consistently outperforms the other methods across all time intervals.

TABLE III: Dataset size, submap number, and runtime performance on 30 sequences. For BTC, we evaluate both multi-thread and single-thread implementation. In multi-thread implementation, four threads are employed in the candidate submap search of loop detection (including descriptor querying and binary similarity checking), while the geometrical verification of loop detection and the descriptor extraction run in a single thread. The results of single-thread implementation is shown inside parentheses. For BoW3D and FreSCo, they are only evaluated on KITTI, so their average time are not computed to avoid biased results.

Sequence	**Path length [km]** (# revisits / # total)	**Avg. points / submap**	**Descriptor Extraction [ms] / Loop Detection [ms] / Total [ms]**					
			BTC	Scan Context	NDT	M2DP	BoW3D	FreSCo
KITTI00	3.72 (1037 / 4541)	121,495	11.0 / 6.6 (19.6) / **17.7** (30.6)	**10.6** / 23.1 / 33.7	31.8 / 180.9 / 212.7	29.5 / **0.3** / 29.8	15.1 / 21.3 / 36.4	561.9 / 37.2 / 599.1
KITTI02	5.06 (358 / 4661)	125,627	14.0 / 8.3 (25.4) / **22.4** (39.4)	**10.6** / 23.0 / 33.6	33.1 / 179.9 / 213.0	30.9 / **0.3** / 31.2	12.6 / 17.8 / 30.4	547.3 / 46.6 / 593.9
KITTI05	2.21 (662 / 2761)	125,037	12.1 / 6.4 (18.4) / **18.5** (30.5)	**11.0** / 22.7 / 33.7	35.5 / 119.4 / 154.8	33.8 / **0.2** / 34.0	17.1 / 22.7 / 39.8	537.2 / 41.1 / 578.3
KITTI06	1.23 (606 / 1101)	122,300	13.8 / 9.1 (24.4) / **22.9** (38.2)	**12.6** / 21.2 / 33.8	49.6 / 46.2 / 95.8	45.7 / **0.1** / 45.8	16.4 / 12.5 / 28.9	528.9 / 37.2 / 566.1
KITTI07	3.22 (141 / 1101)	121,330	12.0 / 3.6 (9.6) / **15.6** (21.6)	**9.7** / 21.2 / 30.9	31.7 / 41.7 / 73.4	29.9 / **0.1** / 30.0	16.3 / 13.5 / 29.9	565.7 / 37.6 / 603.3
KITTI08	2.45 (478 / 4047)	122,593	13.4 / 6.2 (20.8) / **19.6** (34.2)	**11.8** / 23.0 / 34.7	39.9 / 258.3 / 298.2	38.3 / **0.3** / 34.6	17.6 / 22.2 / 39.8	547.7 / 44.9 / 592.6
NCLT1	6.35 (1280 / 5284)	194,611	**18.5** / 16.3 (33.2) / **34.7** (51.7)	23.0 / 23.8 / 46.9	74.2 / 430.1 / 504.3	42.6 / **0.3** / 42.9	– / – / –	– / – / –
NCLT2	6.02 (992 / 4994)	198,008	**18.2** / 16.7 (33.7) / **34.8** (51.9)	23.2 / 23.8 / 47.0	72.9 / 439.9 / 512.8	42.9 / **0.3** / 43.1	– / – / –	– / – / –
NCLT3	5.58 (1208 / 4647)	197,000	**18.7** / 14.3 (24.6) / **33.0** (43.3)	22.3 / 23.4 / 45.7	70.8 / 325.5 / 396.4	40.7 / **0.2** / 40.9	– / – / –	– / – / –
NCLT4	6.54 (601 / 4153)	207,651	**23.6** / 17.6 (35.1) / **41.3** (58.7)	27.2 / 23.6 / 50.8	110.9 / 415.8 / 526.7	59.4 / **0.2** / 59.6	– / – / –	– / – / –
CU1	9.56 (120 / 1470)	356,824	**37.3** / 11.9 (26.8) / **49.3** (64.1)	41.8 / 22.2 / 64.0	179.8 / 89.3 / 269.1	88.2 / **0.1** / 88.3	– / – / –	– / – / –
CU2	16.35 (537 / 2175)	347,975	**29.6** / 15.3 (42.3) / **44.9** (71.9)	32.9 / 23.8 / 56.7	154.2 / 189.4 / 343.6	79.8 / **0.2** / 80.0	– / – / –	– / – / –
CU3	15.70 (439 / 2079)	360,970	**26.5** / 8.3 (21.8) / **34.8** (48.3)	34.2 / 22.8 / 57.0	182.0 / 231.2 / 413.1	84.1 / **0.2** / 84.3	– / – / –	– / – / –
CU4	14.67 (159 / 1561)	391,331	**29.7** / 8.4 (24.1) / **38.1** (53.8)	39.4 / 22.3 / 61.7	197.9 / 198.1 / 396.0	96.2 / **0.2** / 96.4	– / – / –	– / – / –
CU5	21.84 (765 / 3185)	397,388	**25.7** / 11.4 (33.7) / **37.1** (59.4)	32.0 / 23.3 / 55.3	206.8 / 342.1 / 548.9	92.8 / **0.2** / 93.0	– / – / –	– / – / –
K-01	5.12 (231 / 3932)	315,715	**30.7** / 7.7 (20.3) / **38.4** (51.0)	43.5 / 22.7 / 66.2	185.0 / 245.6 / 430.7	89.9 / **0.2** / 90.0	– / – / –	– / – / –
K-02	5.63 (381 / 4253)	225,240	**22.0** / 5.5 (14.7) / **27.5** (36.7)	38.6 / 23.4 / 62.0	196.2 / 329.2 / 525.5	86.2 / **0.2** / 86.4	– / – / –	– / – / –
K-03	6.18 (1258 / 5059)	334,566	**32.0** / 7.2 (23.1) / **39.2** (55.1)	46.6 / 23.0 / 69.7	196.2 / 325.7 / 522.0	90.4 / **0.2** / 90.6	– / – / –	– / – / –
K-04	5.55 (514 / 3975)	323,248	**28.8** / 5.2 (13.2) / **34.1** (42.0)	41.7 / 22.8 / 64.4	187.8 / 348.1 / 535.9	81.3 / **0.2** / 81.4	– / – / –	– / – / –
V-01	2.64 (529 / 2109)	332,214	**32.8** / 4.8 (13.3) / **37.6** (46.1)	50.6 / 22.2 / 72.9	236.8 / 407.9 / 644.7	109.0 / **0.1** / 109.1	– / – / –	– / – / –
V-02	2.62 (442 / 1996)	332,680	**31.6** / 4.6 (11.7) / **36.3** (43.3)	49.1 / 22.2 / 71.3	237.0 / 371.9 / 608.9	108.6 / **0.1** / 108.7	– / – / –	– / – / –
V-03	4.56 (576 / 3436)	366,496	**29.8** / 4.6 (11.9) / **34.4** (41.7)	50.4 / 22.6 / 72.9	260.1 / 758.6 / 1018.6	106.3 / **0.1** / 106.4	– / – / –	– / – / –
V-04	2.80 (361 / 2320)	326,698	**22.3** / 8.4 (27.9) / **30.7** (50.2)	45.7 / 22.4 / 68.1	229.4 / 633.5 / 862.9	98.1 / **0.1** / 98.2	– / – / –	– / – / –
HORIZON1	3.60 (138 / 1251)	72,135	**9.5** / 8.5 (20.4) / **18.1** (29.9)	– / – / –	31.5 / 45.6 / 77.1	28.4 / **0.1** / 28.5	– / – / –	– / – / –
HORIZON2	0.96 (74 / 317)	207,748	**18.5** / 11.9 (14.9) / **30.4** (33.4)	– / – / –	58.4 / 25.8 / 84.2	59.7 / **0.1** / 59.8	– / – / –	– / – / –
HORIZON3	1.67 (103 / 559)	208,055	**19.2** / 15.9 (28.1) / **35.2** (47.3)	– / – / –	65.4 / 32.5 / 97.9	63.1/ **0.1** / 63.2	– / – / –	– / – / –
AVIA1	0.89 (122 / 599)	238,400	**25.8** / 14.7 (31.9) / **40.5** (57.7)	– / – / –	63.8 / 211.7 / 275.5	61.0 / **0.1** / 61.1	– / – / –	– / – / –
AVIA2	0.80 (151 / 539)	238,404	**24.9** / 18.3 (35.3) / **43.2** (60.2)	– / – / –	74.8 / 244.7 / 319.5	58.2 / **0.1** / 58.3	– / – / –	– / – / –
AVIA3	0.89 (179 / 399)	211,741	**39.6** / 15.6 (27.2) / **55.1** (66.8)	– / – / –	129.5 / 34.7 / 164.2	91.5 / **0.1** / 91.6	– / – / –	– / – / –
AVIA4	0.96 (187 / 380)	178,778	**37.2** / 15.9 (30.0) / **54.1** (67.2)	– / – / –	116.9 / 45.4 / 162.2	78.0 / **0.1** / 78.1	– / – / –	– / – / –
Average	5.42 (487 / 2706)	243,408	**22.6** / 10.3 (23.9) / **32.9** (46.5)	30.4 / 22.9 / 53.3	124.6 / 251.4 / 376.0	68.1 / **0.2** / 68.3	– / – / –	– / – / –

Our proposed BTC provides the relative transformation of all six degrees of freedom (DoF) between the query and loop submaps. BoW3D [48] is also capable of estimating the full 6-DoF relative pose because the Link3D descriptor [51] it uses is pose invariant, hence allowing to establish keypoints correspondences necessary for relative pose estimation. Scan Context [8] can only estimate the yaw angle between two submaps, which, as claimed by the authors [8], can accelerate point cloud registration and enhance ICP accuracy; thus, we additionally run a G-ICP registration [46] following the Scan Context to determine the 6 DoF relative pose (denoted as SC+GICP). We also test the original G-ICP [46] alone under the same conditions. For FreSCo [45], it first estimates the yaw angle from descriptor matching similar to Scan Context [8], then it executes a 2D normal-based ICP on the flattened 2D point clouds to obtain a full 6-DoF relative pose estimation. Finally, as M2DP [39] and NDT [34] do not offer any information regarding relative pose estimation, we exclude them from the results.

We conduct pose estimation evaluation on ground-truth loop submap pairs. For each loop pair, the evaluated method first performs loop detection by extracting descriptors (for both submaps), matching descriptors, and examining if the submap scene similarity exceeds the loop detection threshold. If true, the method proceeds to estimate the relative pose between these two submaps. To test the robustness of pose estimation under different initial poses, for each loop pair, we perturb the initial relative pose from the true relative pose at different

TABLE IV: Relative Pose Estimation Evaluation. The descriptor time includes the descriptor extraction time for both submaps in a loop pair (note that for Scan Context, the descriptor augmentation was performed only for one submap in a pair).

	KITTI00 with small perturbation						KITTI08 with small perturbation					
	Recall Rate	Sucess Rate	TE [m]	RE [deg]	Descriptor Time [ms]	Registration Time [ms]	Recall Rate	Sucess Rate	TE [m]	RE [deg]	Descriptor Time [ms]	Registration Time [ms]
BTC	**97.05%**	**99.68%**	**0.059**	**0.154**	25.23	**2.83**	**96.13%**	**100.00%**	**0.095**	**0.233**	31.17	**3.51**
BoW3D	70.57%	98.96%	0.088	0.481	35.11	11.29	36.33%	87.24%	0.142	1.593	38.78	13.09
FreSCo	89.78%	97.42%	0.297	0.305	1076.29	81.24	86.09%	96.27%	0.371	0.380	1083.16	103.21
SC+GICP	81.36%	99.37%	0.071	0.206	**11.66**	120.25	70.58%	98.75%	0.190	0.325	**12.37**	235.34
GICP	–	93.70%	0.102	0.281	–	169.52	–	94.91%	0.246	0.405	–	306.40
	KITTI00 with large perturbation						KITTI08 with large perturbation					
	Recall Rate	Sucess Rate	TE [m]	RE [deg]	Descriptor Time [ms]	Registration Time [ms]	Recall Rate	Sucess Rate	TE [m]	RE [deg]	Descriptor Time [ms]	Registration Time [ms]
BTC	**97.05%**	**99.68%**	**0.059**	**0.154**	25.23	**2.83**	**96.13%**	**100.00%**	**0.095**	**0.233**	31.17	**3.51**
BoW3D	70.57%	98.96%	0.088	0.481	35.11	11.29	36.33%	87.24%	0.142	1.593	38.78	13.09
FreSCo	89.78%	57.23%	0.384	0.316	1076.29	147.95	86.09%	37.55%	0.411	0.463	1083.16	167.18
SC+GICP	81.36%	48.94%	0.081	0.213	**11.66**	201.85	70.58%	30.01%	0.206	0.420	**12.37**	412.23
GICP	–	46.03%	0.115	0.298	–	230.50	–	30.09%	0.283	0.505	–	465.70

scales: small perturbations ($\pm 10^\circ$ in each axis of rotation and ± 5 m in each axis of translation) and large perturbations ($\pm 90^\circ$ in each axis of rotation and ± 10 m in each axis of translation). The parameters of each method are set to the values that yield the maximum F1 scores in its respective precision-recall curve shown in Section VIII-A.

We evaluate all methods in terms of the overall recall rate (successfully detected loop pairs over all ground-truth pairs), the success rate (successfully registered loop pairs over all successfully detected loop pairs), and the average translation error (TE) and rotation error (RE) of all successfully registered loop pairs. We consider a loop pair to be successfully registered if the TE is below 3m and the RE is below 5° when compared with the ground-truth pose. We also evaluate the computation efficiency of all methods in terms of averaged descriptor extraction time (for both submaps in a loop pair) and pose registration time (comprising descriptor matching, scene similarity assessment, and pose estimation).

The results on KITTI00 and KITTI08 sequences are reported in Table IV. As can be seen, both BTC and BoW3D demonstrate remarkable invariance to rotation and translation perturbations, as indicated by their consistent performance across both small and large initial pose disturbances. This stability arises from their ability to extract features completely in the local frame of the point cloud, hence not susceptible to initial pose perturbations. In contrast, the success rate of FreSCo, SC+GICP and GICP drops significantly when there is a large pose disturbance due to their dependence on good initial pose estimations for convergence.

Overall, BTC achieves the highest recall rate and success rate with the lowest TE and RE compared to all other methods. Meanwhile, BTC has the least registration time, about 4x faster than BoW3D and 50x faster than SC-ICP and G-ICP. BTC's superior performance and speed are attributed to two factors: its triangle matching scheme and binary similarity checking. These techniques significantly enhance the accuracy of keypoints matching, thus enhancing the efficiency and accuracy of the pose registration.

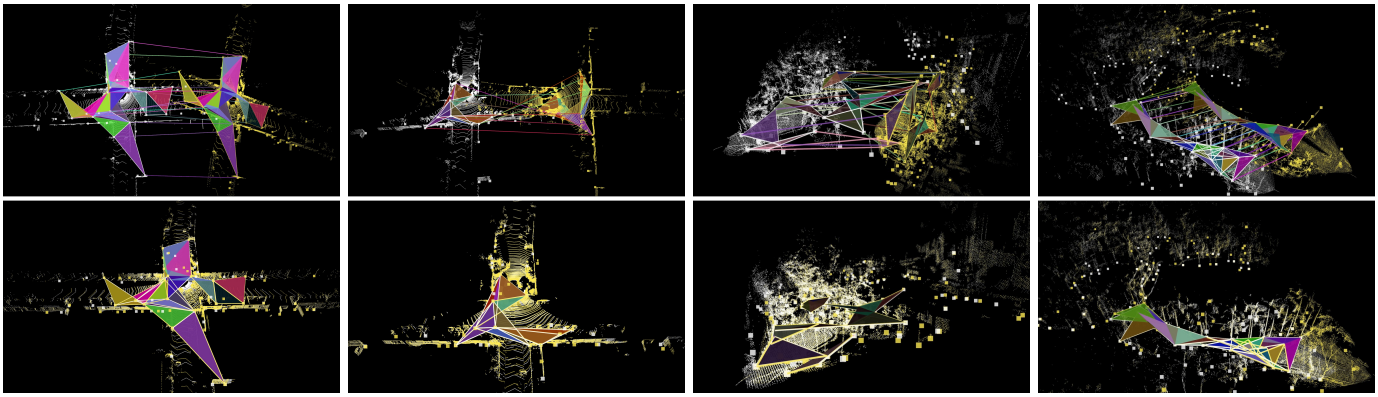
Fig. 14 further illustrates several examples of submap registration utilizing BTC. As demonstrated, BTC effectively aligns pairs of loop submaps in various environments, including both urban and unstructured environments.

D. Ablation Study

In this section, we provide ablation studies on the key designs of our method.

1) Binary Descriptor: A key novel design with respect to our previous conference paper [11] is the binary descriptor, which aims to improve the accuracy and efficiency of triangle matching. This experiment investigates the effectiveness of the binary descriptor. We select four representative sequences from the benchmark datasets for the ablation study: KITTI00 (urban environment with a spinning LiDAR), Wild-Place K-03 (forest environment with a spinning LiDAR), HORIZON1 (urban environment with a solid-state LiDAR), and AVIA1 (unstructured environment with a solid-state LiDAR). In the experiment, we evaluate the full BTC, denoted as *BTC (full)*, and compare it with the BTC without binary descriptor, termed as *BTC (w/o binary)*, while keeping all other configuration parameters the same. More specifically, the two methods use the same set of keypoints and triangle descriptors, but in *BTC (w/o binary)*, only the side lengths of the triangle descriptors are used for matching, and the binary similarity checking of triangle vertices detailed in Sec. VI-A is omitted.

We report the average precision (AP), average computation time (including both descriptor extraction and loop detection) and descriptor matching accuracy of all true positive loops in Table V. To calculate the descriptor matching accuracy, we use the relative pose between a loop pair provided by the geometrical verification to transform all descriptors of the querying submap to the loop submap's frame. Then, distances between the three triangle vertices of each BTC descriptor pair are calculated. Pairs of BTC descriptors with all vertices closer than 2 m are considered correct matches. The matching accuracy is then defined as the ratio of the number



(a) RPY: $(18.3°, -19.1°, -19.2°)$ (b) RPY: $(-4.8°, 30.4°, 81.3°)$ (c) RPY: $(-81.2°, 71.4°, 45.1°)$ (d) RPY: $(23.3°, 70.8°, 19.5°)$

Fig. 14: Registration of loop submap pairs using BTC in four different cases, each with varying initial RPY errors. The top row shows the point clouds before registration and displays their initial relative pose, with the two point clouds and their corresponding keypoints colored in white and yellow, respectively. Matched descriptor pairs are represented by triangle surfaces colored in the same color. The bottom row displays the point clouds after registration. In all cases, BTC successfully aligns the point clouds despite the large errors in initial poses.

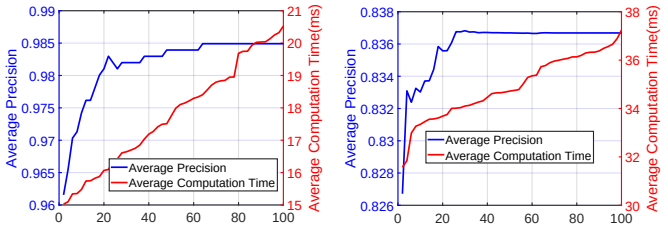
of correct matches to the total number of matches input to the geometrical verification step.

As can be seen from Table V, *BTC (full)* significantly outperforms *BTC (w/o binary)* with much higher AP and lower computation time. The improvement in AP is due to the significantly higher matching accuracy of the triangles, as evidenced in the last column, which effectively suppresses the votes on false loop submaps. The higher triangle matching accuracy also eliminates the number of false keypoints (i.e., triangle vertices) correspondences, hence reducing the time for RANSAC. The time reduction in RANSAC even outweighs the time for extracting the binary descriptor, leading to a less overall time for *BTC (full)*. These findings highlight the importance of binary descriptors, particularly for challenging environments such as forest scenes (K-03) and the Livox dataset (HORIZON1 and AVIA1).

TABLE V: Ablation Study: Comparison of place recognition performance using *BTC (full)* and *BTC (w/o binary)* on four representative sequences.

Sequence	*BTC (full)* / *BTC (w/o binary)*		
	Avg. Precision	Avg. Comp. Time[ms]	Match Accuracy
KITTI00	0.983/0.974	17.71/20.92	0.874/0.668
K-03	0.874/0.786	39.21/66.10	0.762/0.440
HORIZON1	0.890/0.801	14.15/30.79	0.773/0.369
AVIA1	0.867/0.687	19.86/36.73	0.664/0.209

2) Candidate number: In the default implementation, as detailed in Sec. VI-A, we selected $\kappa = 50$ candidate submaps from the Hash table database and eventually picked the best one through geometrical verification (Sec. VI-B). To investigate the effect of candidate number κ on loop detection performance, we conduct an ablation study by varying κ during the retrieval process. We select two sequences (KITTI00 and NCLT1), whose average precision and computation time (including both descriptor extraction and loop detection) are reported in Fig. 15. As can be seen, as κ increases, the computation time also increases due to the more number of candidate submap to process in loop detection. However, the detection precision does not continue to increase with κ after 63 because with the binary descriptor, the true loop submap, if any, easily stands out as a top 63 candidate. Considering both time efficiency and precision, we selected a candidate number of 50 in the default implementation.



(a) KITTI00 (b) NCLT1

Fig. 15: Ablation study: Analyzing the effect of candidate number κ on average precision and computation time. Results are presented for two distinct sequences: (a) KITTI00 and (b) NCLT1.

3) Number of Reference Planes: We further analyze the performance impact of varying the number of reference planes across diverse scenarios, specifically focusing on sequences KITTI08, NCLT2, AVIA3 and AVIA4. Table VI showcases the average precision and computation time. Our analysis suggests that for sequences featuring relatively flat terrains, such as KITTI08 and NCLT2, a single reference plane is ideal. Interestingly, in such environments, adding more reference planes can decrease detection performance. This is primarily because when a large flat ground serves as the main reference plane, any additional reference plane might often correspond to vertical structures, such as walls. Descriptors derived from these secondary planes can potentially interfere with the matching process of more salient descriptors from the primary plane, consequently diminishing detection accuracy. In contrast, in terrains marked by significant height variations like AVIA3 and AVIA4, the introduction of multiple reference planes does effectively improve the detection performance, but at the expense of increased computation time due to the extraction of more descriptors.

E. Comparision with Deep learning-based methods

In this section, we provide comparisons to recent state-of-the-art deep learning-based approaches, namely LCDNet [10] and Logg3D-Net [52]. For both methods, we use pre-trained

Sequence	Avg. Precision / Avg. Comp. Time[ms]		
	$M = 1$	$M = 2$	$M = 3$
KITTI08	0.98/19.63	0.92/23.31	0.89/26.88
NCLT2	0.85/34.83	0.79/41.25	0.73/48.62
AVIA3	0.60/48.39	0.74/55.12	0.75/63.41
AVIA4	0.52/46.52	0.68/54.13	0.69/67.83

TABLE VI: Ablation study: Comparison of place recognition performance across different sequences for varying values of M (number of reference planes)

models released by the authors. For LCDNet, it is noted that better performance is achieved on the KITTI dataset when ground points are removed during pre-processing. To ensure a fair comparison, we evaluate LCDNet with and without ground points removal, denoted as LCDNet* and LCDNet. On the other hand, the original implementation of Logg3D-Net excludes the recent 300 submps in the loop detection; we modify it to 100, the number used by all other methods, for a fair comparison.

Table VII shows the Average Precision (AP) of all methods. Considering that the ground-truth for the training set of the two deep learning methods is calculated using the distance criterion, we also record the AP based on the distance criterion in addition to the overlap-based criterion as described in Section VII-C. As can be seen, under the distance criterion, BTC achieves an AP that is significantly higher than Logg3D-Net, similar to LCDNet, and slightly lower than LCDNet*, which has additional ground points removal. Under the overlap criterion, BTC outperforms all other methods in all sequences with significant margins. This outcome is reasonable, as both LCDNet and Logg3D-Net are trained under the distance criterion, which may lead to missed loop submaps that are farther apart but still exhibit a relatively large overlap.

Table VIII shows the computation time of all methods. BTC boasts significant advantages in terms of computation time, with its much more efficient descriptor extraction than that of both LCDNet and Logg3D-Net. LCDNet* extracts descriptors more rapidly than LCDNet, as a large number of ground points has been removed. However, such ground points removal consumes considerable additional time.

TABLE VII: Comparisons with learning-based methods on KITTI Odometry dataset.

Sequence	AP in 4m distance criteria / AP in overlap criterion			
	BTC	Logg3D-Net	LCDNet	LCDNet*
KITTI00	0.977/**0.983**	0.982/0.828	0.987/0.906	**0.988**/0.906
KITTI02	0.848/**0.963**	0.783/0.772	0.931/0.913	**0.934**/0.913
KITTI05	0.938/**0.969**	0.955/0.852	0.955/0.922	**0.956**/0.922
KITTI06	0.957/**0.988**	**0.997**/0.548	0.992/0.606	0.988/0.859
KITTI07	**0.953/0.833**	0.910/0.691	0.858/0.713	0.858/0.713
KITTI08	**0.935/0.985**	0.465/0.806	0.843/0.926	0.917/0.927
Avg.	0.926/**0.968**	0.808/0.787	0.925/0.885	**0.946**/0.901

TABLE VIII: Runtime comparisons with learning-based methods on KITTI Odometry dataset. All units are in ms

Method	Pre-process	Descriptor Extraction	Retrieval	Total
BTC	0.00	12.70	6.88	19.58
Logg3d-Net	13.78	49.55	0.47	63.79
LCDNet	0.00	173.80	0.02	173.82
LCDNet*	1012.35	110.39	0.01	1122.75

IX. Applications

In this section, we demonstrate the practical applications of the BTC descriptor in two major robotic applications: SLAM and Multi-Map Alignment. Each application highlights the unique advantages and substantial enhancements offered by the BTC descriptor. When integrated into a SLAM system, the BTC descriptor provides a robust and efficient solution for loop closure detection, which consequently leads to more accurate localization and mapping. Furthermore, the BTC descriptor can be used for Multi-Map Alignment to identify co-visible areas and estimate the relative pose between different point cloud maps that are collected and constructed at different time. Due to page limit, we put the implementation details and result analysis of these applications in the Supplementary Material [62].

X. Conclusion

In this paper, we presented the Binary Triangle Combined (BTC) Descriptor, a novel approach to place recognition that combines local and global features for improved performance in diverse and multi-scale environments. BTC builds on the advantages of our previous work by employing the geometric invariance of triangle shapes and introducing a local binary descriptor to enhance discriminative capabilities during the matching process. This combination allows for more accurate and efficient place recognition while maintaining robustness across different conditions.

Our experimental evaluation demonstrated that BTC outperforms state-of-the-art methods across various environments and with different types of LiDAR sensors. In particular, our method exhibits significant improvements in precision and adaptability, especially in challenging environments. By open-sourcing our codes, we aim to contribute to the robotics community and promote further advancements in place recognition and robot navigation.

While the performance of BTC is impressive, there are situations where limitations have been observed. For instance, in smooth long corridor environments (e.g., tunnels, long halls), the method struggles to extract valid keypoints, resulting in unsuccessful place recognition (i.e., false negatives). Furthermore, false positives can be observed in cases where two places share high geometric similarity and point cloud distribution (e.g., two offices with the same layout). These limitations are ultimately due to the insufficient or non-discriminative geometry features offered by the point cloud. Addressing such issues would require the integration of other sensing modalities (e.g., visual information).

References

[1] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardós, "Orb-slam: A versatile and accurate monocular slam system," *IEEE Transactions on Robotics*, vol. 31, no. 5, pp. 1147–1163, 2015.

[2] Y. Zhu, C. Zheng, C. Yuan, X. Huang, and X. Hong, "Camvox: A low-cost and accurate lidar-assisted visual slam system," 2020.

[3] J. Lin and F. Zhang, "Loam_livox: A fast, robust, high-precision lidar odometry and mapping package for lidars of small fov," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 3126–3131.

[4] C. Yuan, W. Xu, X. Liu, X. Hong, and F. Zhang, "Efficient and probabilistic adaptive voxel mapping for accurate online lidar odometry," *IEEE Robotics and Automation Letters*, vol. 7, no. 3, pp. 8518–8525, 2022.

[5] M. Labbé and F. Michaud, "Appearance-based loop closure detection for online large-scale and long-term operation," *IEEE Transactions on Robotics*, vol. 29, no. 3, pp. 734–745, 2013.

[6] G. Kim, B. Park, and A. Kim, "1-day learning, 1-year localization: Long-term lidar localization using scan context image," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 1948–1955, 2019.

[7] A. Khaliq, S. Ehsan, Z. Chen, M. Milford, and K. McDonald-Maier, "A holistic visual place recognition approach using lightweight cnns for significant viewpoint and appearance changes," *IEEE Transactions on Robotics*, vol. 36, no. 2, pp. 561–569, 2020.

[8] 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.

[9] G. Kim, S. Choi, and A. Kim, "Scan context++: Structural place recognition robust to rotation and lateral variations in urban environments," *IEEE Transactions on Robotics*, vol. 38, no. 3, pp. 1856–1874, 2021.

[10] D. Cattaneo, M. Vaghi, and A. Valada, "Lcdnet: Deep loop closure detection and point cloud registration for lidar slam," *IEEE Transactions on Robotics*, vol. 38, no. 4, pp. 2074–2093, 2022.

[11] C. Yuan, J. Lin, Z. Zou, X. Hong, and F. Zhang, "Std: Stable triangle descriptor for 3d place recognition," *arXiv preprint arXiv:2209.12435*, 2022.

[12] M. Cummins and P. Newman, "Fab-map: Probabilistic localization and mapping in the space of appearance," *The International Journal of Robotics Research*, vol. 27, no. 6, pp. 647–665, 2008.

[13] 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.

[14] M. J. Milford and G. F. Wyeth, "Seqslam: Visual route-based navigation for sunny summer days and stormy winter nights," in *2012 IEEE international conference on robotics and automation*. IEEE, 2012, pp. 1643–1649.

[15] M. Ramezani, Y. Wang, M. Camurri, D. Wisth, M. Mattamala, and M. Fallon, "The newer college dataset: Handheld lidar, inertial and vision with ground truth," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020, pp. 4353–4360.

[16] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[17] T.-M. Nguyen, S. Yuan, M. Cao, Y. Lyu, T. H. Nguyen, and L. Xie, "Ntu viral: A visual-inertial-ranging-lidar dataset, from an aerial vehicle viewpoint," *The International Journal of Robotics Research*, vol. 41, no. 3, pp. 270–280, 2022.

[18] N. Carlevaris-Bianco, A. K. Ushani, and R. M. Eustice, "University of Michigan North Campus long-term vision and lidar dataset," *International Journal of Robotics Research*, vol. 35, no. 9, pp. 1023–1035, 2015.

[19] J. Knights, K. Vidanapathirana, M. Ramezani, S. Sridharan, C. Fookes, and P. Moghadam, "Wild-places: A large-scale dataset for lidar place recognition in unstructured natural environments," 2022. [Online]. Available: https://arxiv.org/abs/2211.12732

[20] 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.

[21] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International journal of computer vision*, vol. 60, pp. 91–110, 2004.

[22] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool, "Speeded-up robust features (surf)," *Computer vision and image understanding*, vol. 110, no. 3, pp. 346–359, 2008.

[23] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "Orb: An efficient alternative to sift or surf," in *2011 International conference on computer vision*. Ieee, 2011, pp. 2564–2571.

[24] M. Calonder, V. Lepetit, C. Strecha, and P. Fua, "Brief: Binary robust independent elementary features," in *Computer Vision–ECCV 2010: 11th European Conference on Computer Vision, Heraklion, Crete, Greece, September 5-11, 2010, Proceedings, Part IV 11*. Springer, 2010, pp. 778–792.

[25] A. Oliva and A. Torralba, "Modeling the shape of the scene: A holistic representation of the spatial envelope," *International journal of computer vision*, vol. 42, pp. 145–175, 2001.

[26] P. Furgale and T. D. Barfoot, "Visual teach and repeat for long-range rover autonomy," *Journal of field robotics*, vol. 27, no. 5, pp. 534–560, 2010.

[27] J. Mo and J. Sattar, "A fast and robust place recognition approach for stereo visual odometry using lidar descriptors," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020, pp. 5893–5900.

[28] B. Steder, G. Grisetti, and W. Burgard, "Robust place recognition for 3d range data based on point features," in *2010 IEEE International Conference on Robotics and Automation*, 2010, pp. 1400–1405.

[29] P. Scovanner, S. Ali, and M. Shah, "A 3-dimensional sift descriptor and its application to action recognition," in *Proceedings of the 15th ACM international conference on Multimedia*, 2007, pp. 357–360.

[30] R. B. Rusu, N. Blodow, Z. C. Marton, and M. Beetz, "Aligning point cloud views using persistent feature histograms," in *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2008, pp. 3384–3391.

[31] S. Salti, F. Tombari, and L. Di Stefano, "Shot: Unique signatures of histograms for surface and texture description," *Computer Vision and Image Understanding*, vol. 125, pp. 251–264, 2014.

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

[33] ——, "Keypoint design and evaluation for place recognition in 2d lidar maps," *Robotics and Autonomous Systems*, vol. 57, no. 12, pp. 1211–1224, 2009.

[34] 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.

[35] S. Siva, Z. Nahman, and H. Zhang, "Voxel-based representation learning for place recognition based on 3d point clouds," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020, pp. 8351–8357.

[36] J. Komorowski, "Minkloc3d: Point cloud based large-scale place recognition," in *2021 IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2021, pp. 1789–1798.

[37] 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, pp. 189–206, 2013.

[38] T.-X. Xu, Y.-C. Guo, Y.-K. Lai, and S.-H. Zhang, "Transloc3d: Point cloud based large-scale place recognition using adaptive receptive fields," *arXiv preprint arXiv:2105.11605*, 2021.

[39] L. He, X. Wang, and H. Zhang, "M2dp: A novel 3d point cloud descriptor and its application in loop closure detection," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2016, pp. 231–237.

[40] S. Belongie, J. Malik, and J. Puzicha, "Shape matching and object recognition using shape contexts," *IEEE transactions on pattern analysis and machine intelligence*, vol. 24, no. 4, pp. 509–522, 2002.

[41] L. Luo, S. Zheng, Y. Li, Y. Fan, B. Yu, S. Cao, and H. Shen, "Bevplace: Learning lidar-based place recognition using bird's eye view images," 2023.

[42] C. Fu, L. Li, L. Peng, Y. Ma, X. Zhao, and Y. Liu, "Overlapnetvlad: A coarse-to-fine framework for lidar-based place recognition," 2023.

[43] X. Xu, S. Lu, J. Wu, H. Lu, Q. Zhu, Y. Liao, R. Xiong, and Y. Wang, "Ring++: Roto-translation-invariant gram for global localization on a sparse scan map," *IEEE Transactions on Robotics*, 2023.

[44] H. Jang, M. Jung, and A. Kim, "Raplace: Place recognition for imaging radar using radon transform and mutable threshold," 2023.

[45] Y. Fan, X. Du, L. Luo, and J. Shen, "Fresco: Frequency-domain scan context for lidar-based place recognition with translation and rotation invariance," in *2022 17th International Conference on Control, Automation, Robotics and Vision (ICARCV)*, Dec 2022, pp. 576–583.

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

[47] H. Yang, J. Shi, and L. Carlone, "Teaser: Fast and certifiable point cloud registration," *IEEE Transactions on Robotics*, vol. 37, no. 2, pp. 314–333, 2021.

[48] Y. Cui, X. Chen, Y. Zhang, J. Dong, Q. Wu, and F. Zhu, "Bow3d: Bag of words for real-time loop closing in 3d lidar slam," *IEEE Robotics and Automation Letters*, 2022.

[49] G. V. Nardari, A. Cohen, S. W. Chen, X. Liu, V. Arcot, R. A. F. Romero, and V. Kumar, "Place recognition in forests with urquhart tessellations," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 279–286, 2021.

[50] J. Jiang, J. Wang, P. Wang, P. Bao, and Z. Chen, "Lipmatch: Lidar point cloud plane based loop-closure," *IEEE Robotics and Automation Letters*, vol. 5, no. 4, pp. 6861–6868, 2020.
[51] Y. Cui, Y. Zhang, J. Dong, H. Sun, and F. Zhu, "Link3d: Linear keypoints representation for 3d lidar point cloud," 2022.
[52] K. Vidanapathirana, M. Ramezani, P. Moghadam, S. Sridharan, and C. Fookes, "Logg3d-net: Locally guided global descriptor learning for 3d place recognition," in *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, 2022, pp. 2215–2221.
[53] 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.
[54] X. Chen, T. Läbe, A. Milioto, T. Röhling, O. Vysotska, A. Haag, J. Behley, and C. Stachniss, "OverlapNet: Loop Closing for LiDAR-based SLAM," in *Proceedings of Robotics: Science and Systems (RSS)*, 2020.
[55] M. Nießner, M. Zollhöfer, S. Izadi, and M. Stamminger, "Real-time 3d reconstruction at scale using voxel hashing," *ACM Transactions on Graphics (ToG)*, vol. 32, no. 6, pp. 1–11, 2013.
[56] V. K. Wei, "Generalized hamming weights for linear codes," *IEEE Transactions on information theory*, vol. 37, no. 5, pp. 1412–1418, 1991.
[57] M. A. Fischler and R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, 1981.
[58] S. Agarwal, K. Mierle, and Others, "Ceres solver," http://ceres-solver.org.
[59] X. Liu, Z. Liu, F. Kong, and F. Zhang, "Large-scale lidar consistent mapping using hierarchical lidar bundle adjustment," *IEEE Robotics and Automation Letters*, vol. 8, no. 3, pp. 1523–1530, 2023.
[60] J. Jeong, Y. Cho, Y.-S. Shin, H. Roh, and A. Kim, "Complex urban dataset with multi-level sensors from highly diverse urban environments," *International Journal of Robotics Research*, vol. 38, no. 6, pp. 642–657, 2019.
[61] K. Li, M. Li, and U. D. Hanebeck, "Towards high-performance solid-state-lidar-inertial odometry and mapping," *IEEE Robotics and Automation Letters*, vol. 6, no. 3, pp. 5167–5174, 2021.
[62] C. Yuan, J. Lin, Z. Liu, H. Wei, X. Hong, and F. Zhang, "Supplementary material for btc," 2023. [Online]. Available: https://github.com/hku-mars/btc_descriptor/blob/master/supply/Supplementary_Material_for_BTC.pdf