

Adaptive bird's eye view description for long-term mapping and loop closure in 3D point clouds

Andrew Qu

Shanghai High School International Division, Shanghai, China

quandrew0119@gmail.com

Abstract. In this paper, we proposed a robust, adaptive bird eye view descriptor for loop closure detection and long term mapping in 3D point cloud. Loop closure detection plays a crucial role in SLAM context, enhancing the quality and the accuracy of constructed point cloud map. While visual-based loop closure detection methods are vulnerable against perceptual and lightness variances, LiDAR-based methods are robust against such variance. Therefore, to construct our BEV descriptor, we accumulate and register each LiDAR scan to form a key-frame, and develop an algorithm to radially and azimuthally partition and encode the key-frame into a 2D BEV pixel image. Then, similarity scores between every BEV descriptor are calculated to find the best candidate for loop closure. The best candidate is further validated through ICP-based geometric verification, and its newly established restriction is the used for pose graph optimization to improve the quality of the point cloud map. In our experiments, we compare our loop closure detection method with Scan Conext, a state-of-art global descriptor, on public dataset (KITTI) and our private dataset (collected with a Livox solid-state LiDAR). The results of both datasets show that our proposed descriptor has stronger adaptability in different types of environment (both indoor and outdoor) and LiDAR, with a greatly improved accuracy and overall performance.

Keywords: BEV descriptor, Loop closure detection, 3D LiDAR scans, SLAM (robots), Place recognition.

1. Introduction

Mobile robots, while working under real life conditions, must be capable of navigating and locating themselves in large, unknown environments. To do so, these robots are required to map their current environments to localize themselves in it, which is known as the SLAM (Simultaneous Localization and Mapping) problem [1]. A typical SLAM system consists of two modules: odometry and mapping, which are used to approximate the robot's current location and map its current environment. However, in such system, localization inevitably drift with time due to the sensor error accumulation [2]. Loop closure detection, a process which the robot detects and recognizes its returning to previously visited locations, plays a vital role in the fields of SLAM and robotic application. It helps eliminate the accumulated drift due to sensor errors, increasing the accuracy of localization and ensures more reliable navigation in autonomous driving [3]. Further, it can potentially enhance the process of re-localization within pre-build high-resolution map, facilitating immediate recovery from tracking failures and seamless reconstruction of the high-resolution map. This enhanced map can subsequently be utilized in diverse

applications such as virtual reality (VR), augmented reality (AR), and motion/mission planning in autonomous driving [4]. In multi-robot collaboration, loop closure detection may contribute to aligning and associating sensory data from concurrent environment and merging two pre-build maps from two individual robots [5].

Loop closure detection methods for mobile robots can be primarily divided into visual-based and point cloud-based methods. Visual method is more frequently used in SLAM context for its low cost and widespread availability. By comparing visual features and semantic information between two visual images, it is able to determine whether an active loop closure has occur [6]. However, mere visual-based method cannot obtain the geometric and structural information of the environment, and is sensitive to environmental changes: lighting conditions, shadows, and seasonal changes can all alter the results of visual-based loop closure detection [7][8][9]. Hence, in search of a more stable and robust method for loop closure detection, there is a prevailing preference to utilize the point cloud data, often acquired by LiDAR [10]. Such method primarily focuses on developing and matching descriptors from geometric and structural information in current environment [9]. Nevertheless, there are still three inherent issues that existing point cloud-based method need to overcome. First, it must require a high resolution LiDAR scanner to operate, which is a currently high-cost sensor. Second, as shown in Figure 1, the point density in each LiDAR scan decreases along with increasing distance, meaning the density of a typical point cloud varies with distance, hence the descriptors might have varying accuracy. Finally, the numerous point data in a typical point cloud largely expands the cost of time for processing and computing point cloud information [3]. Hence, in order to achieve real-time and seamless loop closure, the point cloud-based method must implement an efficient and less time-consuming algorithm to transform point cloud data into distinctive descriptors.

To solve these technical challenges, in this study, we design a robust Bird's Eye View (BEV) descriptor that is invariant with point cloud densities, scanning pattern, and noise variance between different kind of LiDAR scanners. To achieve real-time and seamless loop closure detection, we further introduce an efficient algorithm that transforms point cloud data into such descriptors and retrieves possible candidates for loop closure through these descriptors. In conclusion, the main contribution of this work can be summarized as below:

- We integrate a robust BEV descriptor into a LiDAR-inertial-based odometer and form a complete SLAM system, suitable for long-term localization and mapping. The implementation is lightweight through a single C++ and is readily integrable to any existing SLAM framework.
- We evaluate our algorithm under multiple scenarios, both indoor and outdoor. Sufficient experimental results verify the effectiveness of our method.
- We extensively evaluate our method on both public and private datasets, and on different types of LiDAR scanner to demonstrate the adaptability of our proposed descriptor. The performance of our method is compared to a state-of-art descriptor, Scan Context [8], and exhibits a considerable improvement.

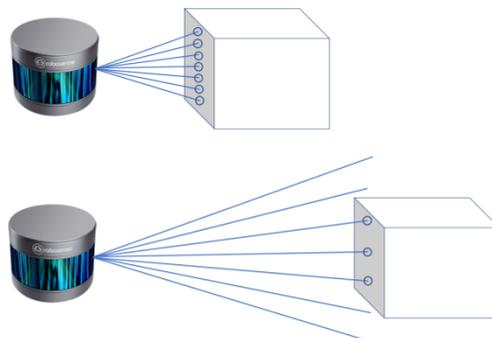


Figure 1. Point density varies from distance

2. Relative Work

Loop closure detection methods, as mentioned previously, are categorized into visual-based and LiDAR-based methods.

2.1. Visual-based Methods

Visual methods are more commonly used for loop closure detection in SLAM frameworks [8] [11] [12] [13]. However, unlike LiDAR-based methods, visual based loop closure detection are usually less robust against perceptual changes. FAB-MAP [8] proposed by Cummins et al. approaches to the problem by proposing a probabilistic generative model of place appearance. Although it does successfully account for perceptual aliasing in surrounding--low probability of revisiting due to indistinctive appearance, visual representation is still vulnerable to weather/lightness changes [14], limiting its real life performance. In contrast, our point cloud representation remains invariant under environment changes. Hence, the descriptor that we proposed is adaptive to different environment. SeqSLAM [15] from Milford et al. achieves loop closure detection with a far improved performance from FAB-MAP through sequential place recognition. Rather than determining a sole optimal match for a given image, SeqSLAM examines the best match location within every local navigation sequence, which proves to be resilient against environment changes. SRAL [16] fuses different features such as colors and lightness from an environment to represent its long-term appearance and transforms loop closure detection into a single regularized optimization problem.

2.2. LiDAR-based Methods

Since LiDAR sensor only extracts geometric information from its surrounding, most LiDAR-based methods exhibit strong resilience against environment and perceptual changes. LiDAR-based methods rely on key-point descriptors, in which nearby points of selected key-points are separated into different cells, and surrounding cells are then encoded into a histogram based on a shared pattern. However, most of these key-point descriptors are originally devised for 3D model part matching [8] and hence revealed limitations when used in loop closure detection context. The conditions of a point cloud deviate much from those of a 3D model. For instance, the density of a 3D model remains constant throughout the model, yet that of a point cloud varies through distance. Moreover, since derived from a real life environment, a point cloud contains more noises than a model. Both suggests that key-point descriptors are not suitable to use for loop closure detection. Scan Context [8] by Kim et al. is a non-histogram based descriptor from LiDAR scans. It transforms the 3D point cloud from various LiDAR scans into 2D visual representations through partitioning the ground areas into different bin based on point cloud top view azimuthal and radial direction. Then, these visual representations are being compared to determine whether loop closure has occur. However, such method stands on the strong assumption that the vehicle has no notable changes in the z-axis, which makes the descriptor sensitive to changes in roll and pitch angles. Our method is adaptive projection, which can be use in multiple extreme conditions such as in crowded urban areas, on a handheld device, in aerial mapping with large motions. SegMatch [17] developed a loop closure detection through deep learning and object segmentation. The training process of the network requires large datasets and hardware support (GPU acceleration). The method that we proposed requires only a workable CPU with no other specific needs for hardware to operate.

In this paper, we propose a robust and adaptive LiDAR-based descriptor that transforms a point cloud into a BEV image, which is then used to search for possible candidates of loop closure. Our method can be applied to variant SLAM contexts, unaffected by changes in environments or types of LiDAR sensors. The process of BEV image generation is inspired by Scan Context [8], which also increases the descriptor's resilience against azimuthal changes. In detail, our proposed method consists of three major component: BEV image generation that preserves the geometric and structural information of point cloud key-frames, retrieval that searches and verifies for possible loop closure candidates, and pose graph optimization according to the loop closure detection results.

3. Method

In this section, we describe how to construct an adaptive BEV descriptor from a 3D point cloud. First, we introduce the LiDAR odometry to extract point cloud key-frames. Next, we apply plane projection and space partition on the key-frames to generate our BEV descriptor. Finally, descriptor-based candidate retrieval and geometric verification are proposed for a complete loop closure detection pipeline. The overall pipeline of our method is depicted in Figure 2.

3.1. LiDAR-Inertial Odometry

Since our proposed method performs loop closure detection on point cloud key-frames, a LiDAR odometer is required to register point cloud from each LiDAR scans to the current key-frame. Then, a new key-frame is generated when a certain number of sub-frames accumulates. By combining data from different sensors, we can get a clearer picture of environment and thus output key-frames with a higher accuracy. Among the various methods available, we've picked the Extended Kalman Filter (EKF) to merge information from LiDAR and IMU sensors. Using EKF helps us map the environment more accurately and deal with difficult situations better. This is mainly because the EKF can handle complex calculations by simplifying them step by step, especially when we assume the errors are normally distributed.

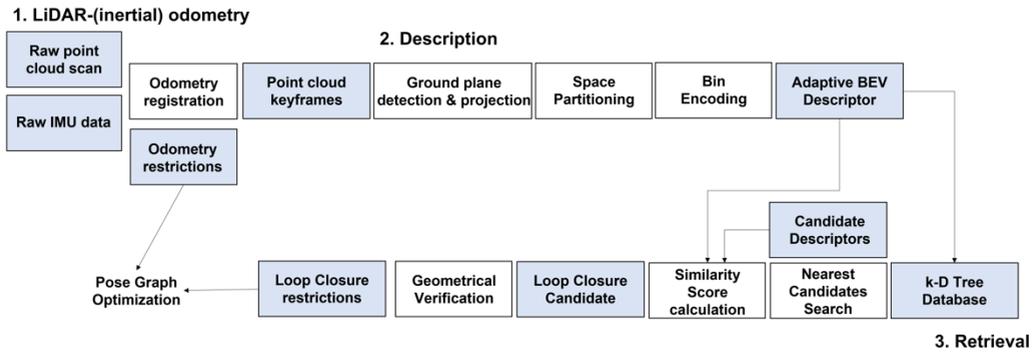


Figure 2. Method Overview. First, LiDAR scans are registered through an odometer based on extended kalman filter, and accumulates to form key-frames. Then, the ground plane is detect to project the points from the key-frame. Key-frames after plane projection will be partitioned and encoded to generate BEV descriptors. Candidates of loop closure are retrieved by their descriptor, then the final candidate is validated through ICP-based geometric verification. Once loop closure is triggered, its provided restrictions will be used for pose graph optimization.

3.1.1. *State Definition.* The state vector, \mathbf{x} , is formulated as:

$$\mathbf{x} = [\mathbf{p}^T \quad \mathbf{R}^T \quad \mathbf{v}^T \quad \mathbf{b}_a^T \quad \mathbf{b}_g^T] \quad (1)$$

Where:

\mathbf{p} represents the position of the LiDAR in a global reference frame. This is the 3D location given by:

$$\mathbf{p} = [x, y, z] \quad (2)$$

Here, x , y , and z denote the coordinates in the east, north, and up directions, respectively.

\mathbf{R} describes the orientation of the LiDAR in the global frame, represented as a rotation matrix:

$$\mathbf{R} = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \quad (3)$$

Each element r_{ij} of the matrix represents the cosine of the angle between the respective axes of the LiDAR's local frame and the global frame.

\mathbf{v} indicates the linear velocity of the LiDAR in the global frame:

$$\mathbf{v} = [v_x, v_y, v_z] \quad (4)$$

This showcases how fast the vehicle is moving and in which direction.

\mathbf{b}_a stands for the accelerometer bias, which can drift over time and impact the accuracy of acceleration measurements:

$$\mathbf{b}_a = [b_{a_x}, b_{a_y}, b_{a_z}] \quad (5)$$

\mathbf{b}_g represents the gyroscope bias, accounting for drifts that affect angular velocity measurements:

$$\mathbf{b}_g = [b_{g_x}, b_{g_y}, b_{g_z}] \quad (6)$$

3.1.2. Process Model. Building upon the state vector \mathbf{x} defined earlier, which includes the position \mathbf{p} , orientation \mathbf{R} , velocity \mathbf{v} , accelerometer bias \mathbf{b}_a , and gyroscope bias \mathbf{b}_g , the process model predicts the next state based on the current state and control inputs.

For LiDAR-Inertial odometry using EKF, the primary control inputs are the IMU measurements. These measurements, particularly angular velocities from the gyroscope and linear accelerations from the accelerometer, can be influenced by the biases \mathbf{b}_a and \mathbf{b}_g , respectively.

Given state vector \mathbf{x} at time t , the state's evolution to time $t + \Delta t$ is predicted using the IMU readings. The accelerations \mathbf{a} (adjusted for \mathbf{b}_a) and angular velocities ω (adjusted for \mathbf{b}_g), usually provided by the accelerometer and gyroscope, are integrated to estimate changes in position, velocity, and orientation.

$$\begin{aligned} \mathbf{p}_{t+\Delta t} &= \mathbf{p}_t + \mathbf{v}_t \Delta t + \frac{1}{2} \mathbf{a}_t \Delta t^2 \\ \mathbf{v}_{t+\Delta t} &= \mathbf{v}_t + \mathbf{a}_t \Delta t \\ \mathbf{R}_{t+\Delta t} &= \mathbf{R}_t \Delta \mathbf{R}(\omega_t \Delta t) \\ \mathbf{b}_{t+\Delta t} &= \mathbf{b}_t \end{aligned} \quad (7)$$

Here: \mathbf{p} represents the position. \mathbf{v} denotes the velocity. \mathbf{R} is the rotation. \mathbf{b} represents the two biases. \mathbf{a} is the acceleration from the IMU. ω is the angular velocity from the gyroscope. $\Delta \mathbf{R}(\cdot)$ is a function that provides the rotation matrix increment given an angular velocity. Small angle approximations or Rodrigues' formula [18] can be used to obtain this incremental rotation. The predicted state is denoted as $\hat{\mathbf{x}}_t$.

3.1.3. Measurement Model. Upon obtaining the predicted state, $\hat{\mathbf{x}}_t$, from the process model, we utilize the LiDAR scan data in conjunction with the previously constructed point cloud map to establish a measurement model to obtain updated state $\bar{\mathbf{x}}_t$. The initial value of $\bar{\mathbf{x}}_t$ is the predicted state $\hat{\mathbf{x}}_t$.

For a given LiDAR point, ${}^B \mathbf{P}_i$, in the LiDAR's body frame, we first transform this point to the global frame:

$${}^G \mathbf{P}_i = {}^G \bar{\mathbf{R}}_t {}^B \mathbf{P}_i + {}^G \bar{\mathbf{p}}_t \quad (8)$$

Subsequently, the nearest plane with plane normal vector \mathbf{u}_i and centroid ${}^G \mathbf{q}_i$ corresponding to this point is identified, and a point-to-plane residual is constructed:

$$\begin{aligned} \mathbf{z}_i &= h(\bar{\mathbf{x}}_t, {}^B \mathbf{P}_i) \\ &= \mathbf{u}_i^T ({}^G \mathbf{P}_i - {}^G \mathbf{q}_i) \\ &= \mathbf{u}_i^T ({}^G \bar{\mathbf{R}}_t {}^B \mathbf{P}_i + {}^G \bar{\mathbf{p}}_t - {}^G \mathbf{q}_i) \end{aligned} \quad (9)$$

To continue, the Jacobian \mathbf{H}_t is computed by differentiating the state variables and is then evaluated at the current estimate $\hat{\mathbf{x}}_t$ of the state:

$$\mathbf{H}_t = [-\mathbf{u}_i^{TG} \widehat{\mathbf{R}}_t [{}^L \mathbf{p}_i]_{\wedge}, \mathbf{u}_i^T, \mathbf{0}_{1 \times 3}, \mathbf{0}_{1 \times 3}, \mathbf{0}_{1 \times 3}] \quad (10)$$

Considering all n LiDAR measurement points collectively, the residuals \mathbf{z}_i and the Jacobian matrix \mathbf{H}_t are amalgamated as:

$$\mathbf{z}_t = h(\bar{\mathbf{x}}_t, {}^B \mathbf{P}) = \begin{bmatrix} h(\bar{\mathbf{x}}_t, {}^B \mathbf{P}_1) \\ h(\bar{\mathbf{x}}_t, {}^B \mathbf{P}_2) \\ \dots \\ h(\bar{\mathbf{x}}_t, {}^B \mathbf{P}_n) \end{bmatrix} \quad (11)$$

$$\mathbf{H}_t = \begin{bmatrix} -\mathbf{u}_1^{TG} \widehat{\mathbf{R}}_t [{}^L \mathbf{p}_1]_{\wedge}, & \mathbf{u}_1^T, & \mathbf{0}_{1 \times 3}, & \mathbf{0}_{1 \times 3}, & \mathbf{0}_{1 \times 3} \\ -\mathbf{u}_2^{TG} \widehat{\mathbf{R}}_t [{}^L \mathbf{p}_2]_{\wedge}, & \mathbf{u}_2^T, & \mathbf{0}_{1 \times 3}, & \mathbf{0}_{1 \times 3}, & \mathbf{0}_{1 \times 3} \\ \dots & \dots & \dots & \dots & \dots \\ -\mathbf{u}_n^{TG} \widehat{\mathbf{R}}_t [{}^L \mathbf{p}_n]_{\wedge}, & \mathbf{u}_n^T, & \mathbf{0}_{1 \times 3}, & \mathbf{0}_{1 \times 3}, & \mathbf{0}_{1 \times 3} \end{bmatrix}$$

Lastly, the pose is optimized to minimize these residuals. As a result, we obtain the updated state $\bar{\mathbf{x}}_t$ of the LiDAR observation model.

3.1.4. State Update. After deducing the prior pose estimate $\hat{\mathbf{x}}_t$ and the associated covariance $\hat{\mathbf{P}}_t$ under the constant velocity model assumption, we incorporate n valid measurements from the latest LiDAR frame. Utilizing this data, we form a Maximum A Posteriori (MAP) estimation to enhance the precision of our current state estimate. The optimization problem can be formulated as:

$$\min_{\mathbf{x}_t} (\| \mathbf{x}_t \ominus \hat{\mathbf{x}}_t \|_{\hat{\mathbf{P}}_t}^2 + \sum_{i=1}^n \| z_i - \mathbf{H}_i \cdot (\mathbf{x}_t \ominus \hat{\mathbf{x}}_t) \|_{\mathbf{W}_i}^2) \quad (12)$$

Given n valid LiDAR measurements, we consolidate them as $\mathbf{H}_t = [\mathbf{H}_1^T, \mathbf{H}_2^T, \dots, \mathbf{H}_n^T]^T$ and $\mathbf{W}_t = \text{diag}(\mathbf{w}_1, \mathbf{w}_2, \dots, \mathbf{w}_n)$. The Kalman gain, \mathbf{K}_t , which represents the weight of the measurement relative to the state prediction, is then computed as:

$$\mathbf{K}_t = \hat{\mathbf{P}}_t \mathbf{H}_t^T (\mathbf{H}_t \hat{\mathbf{P}}_t \mathbf{H}_t^T + \mathbf{W}_t)^{-1} \quad (13)$$

Lastly, the Kalman gain \mathbf{K}_t is employed to procure the optimal updated state \mathbf{x}_t and its associated covariance matrix \mathbf{P}_t :

$$\begin{aligned} \mathbf{x}_t &= \hat{\mathbf{x}}_t \boxplus \mathbf{K}_t (z_t - h(\hat{\mathbf{x}}_t, {}^B \mathbf{P})) \\ \mathbf{P}_t &= (\mathbf{I} - \mathbf{K}_t \mathbf{H}_t) \hat{\mathbf{P}}_t \end{aligned} \quad (14)$$

The updated state \mathbf{x}_t offers a more accurate representation of the system's current state by combining prior knowledge and current measurements, thus refining the overall pose estimation.

3.2. BEV Descriptor

3.2.1. Ground Plane detection. To achieve an adaptive and robust descriptor to changes in roll and pitch angle, we first perform ground plane detection for each key-frame of the entire 3D point cloud. First, we divide the key-frame point cloud into various voxels of certain sizes (e.g., 1m). For each voxel, we want to find a best fitting plane π with crosspoint \mathbf{c} and a normal vector \mathbf{n} for the group of points \mathbf{x}_i ($i = 1, \dots, N$). Therefore, the problem is to mathematically solve:

$$\min_{\mathbf{c}, \mathbf{n}, \|\mathbf{n}\|=1} \sum_{i=1}^N ((\mathbf{x}_i - \mathbf{c})^T \mathbf{n})^2 \quad (15)$$

Notice that is the value of \mathbf{n} will not affect the domain of \mathbf{n} , hence solving \mathbf{c} is equivalent to finding a point that minimizes the sum of squares of the distances from other points to it, which happens to be the definition of the centroid. Then, to solve \mathbf{n} , let $\mathbf{y}_i = \mathbf{x}_i - \mathbf{c}$, the expressions becomes:

$$\min_{n, \|n\|=1} \sum_{i=1}^N (y_i^T n)^2 = \min_{n, \|n\|=1} n^T (YY^T) n \quad (16)$$

Where Y is a $3 \times N$ matrix from the concatenation of vector y_i . By the definition of scatter matrix, YY^T here is the scatter matrix of points x_i . Denote YY^T as S and with constraint $n^T n = 1$, apply Lagrange multiplier to find the local minimum of the Lagrange function:

$$\mathcal{L}(n, \lambda) \equiv f(n) - \lambda(n^T n - 1) \quad (17)$$

Finally, when the eigenvector and eigenvalues of the matrix S are assigned to n and λ respectively, the function is able to satisfy:

$$Sn = \lambda n \quad (18)$$

Where $f(n)$ is able to reach its local extremum.

From the foregoing processes, we calculate the point scatter matrix S for each voxel:

$$\bar{x} = \frac{1}{N} \sum_{i=1}^N x_i; S = \sum_{i=1}^N (x_i - \bar{x})(x_i - \bar{x})^T \quad (19)$$

Apply eigendecomposition to matrix S , and let λ_k denote the K -th largest eigenvalue. According to the plane criterion principle proposed by STD [9]:

$$\lambda_3 < \sigma_1 \text{ and } \lambda_2 > \sigma_2 \quad (20)$$

where σ_1 and σ_2 are pre-set hyperparameters. Using this criterion, we can determine if the points within a voxel can form a plane. Subsequently, we initialize a plane by selecting any such voxel and search for its neighboring voxels. If the neighboring voxels are on the same plane (has the same plane normal direction within a threshold distance), they are included into the expanding plane.

3.2.2. BEV Image Generation. Our BEV Image Generation method is inspired by Scan Context [8], which azimuthally and radially partitions the point cloud and transforms geometric shape of the point cloud around a local key point into an pixel image. Since Scan Context assumes that all LiDAR points lay on a plane perpendicular to the z -axis, its proposed descriptor is less resilient against disturbance in row and pitch angles. Our method effectively addresses this problem by selecting the best-fit plane as a reference. Given a frame of point cloud $P = \{p_1, p_2, \dots, p_N\}$ and its best-fit plane π with crosspoint c and a normal vector n , we first projects the various points in the point cloud onto the ground plane by calculating the Euclidean distance d between the point and the plane:

$$D(p_i) = n^T(p_i - c), \text{ where } i = 1, 2, \dots, N \quad (21)$$

To project the points onto the plane, we rotate the point cloud frame until the z -axis of its local coordinate coincides with the normal vector of the plane. After such transformation, the z -coordinate of every point within the frame will become the calculated point-to-plane distance. Using the key point $c = (\bar{x}, \bar{y})$, each projected point (x_i, y_i) is transformed into polar coordinates (r_i, θ_i) with respect to the key point and the plane by:

$$r_i = \sqrt{(x_i - \bar{x})^2 + (y_i - \bar{y})^2}, \theta_i = \text{atan2}(y_i - \bar{y}, x_i - \bar{x}) \quad (22)$$

After projection, our goal is to partition the scan into N_a angular intervals and N_r radial segments. A starting vector $\mathbf{v}_{\text{start}}$ parallel to the plane and centered at c is defined. Since we have to split the scan into N_a angular intervals, it is then rotated in increments of angles $d\theta = \frac{2\pi}{N_a}$ to cover a complete angular range. For each rotation, the angular interval is then split into radial segments with a constant width $\frac{R_{\text{max}}}{N_r}$, where R_{max} is the maximum LiDAR range. The points on the plane are categorized into the bins formed by these angular intervals and radial segments, as shown in Figure 3.(a).

After scan partitioning, a single value is assigned to each bin. For the cell that belongs to the radial segment i and angular interval j , its value is calculated by:

$$\phi(B(i, j)) = \max_{\mathbf{p} \in B(i, j)} D(\mathbf{p}) \quad (23)$$

where \mathbf{p} is a point that falls into the bin $B(i, j)$ after partitioning and $D(\mathbf{p})$ outputs the Euclidean distance between the point \mathbf{p} before projection and the ground plane. As for empty bins, we assign a zero to them.

The bins are further visualized on a 2D pixel image, where each pixel represents a bin in the scan. The column and row of the pixels are categorized by the partitioned angular intervals and radial segments. As illustrated in Figure 3.(b), the assigned value of each bin is used as the corresponding pixel value, and each pixel is further assigned a color by its value, according to a preset color bar.

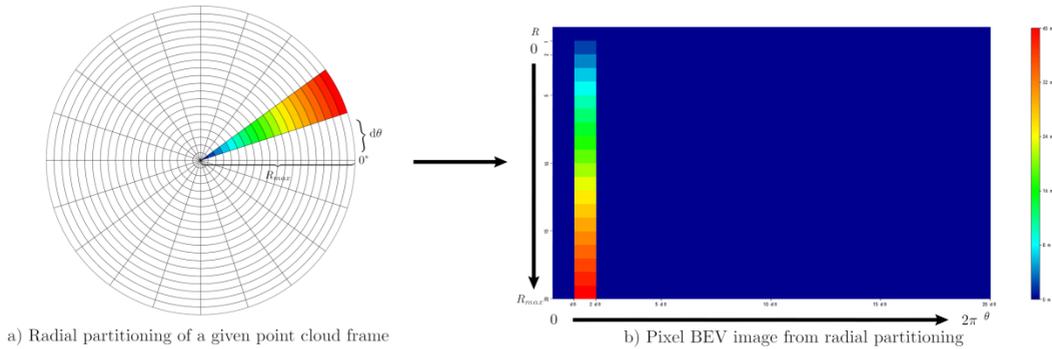


Figure 3. Process of BEV image generation. The partitioning of a point cloud is shown in (a), and the pixel image derived from the radial partitioning is shown in (b). Color of each pixel corresponds to a maximum point-to-plane distance according to the color bar on the right.

3.3. Retrieval

3.3.1. Database Construction and Update. The BEV pixel images generated from point cloud key-frames are used to search for possible candidates of loop closure. The pixel image, with N rows, M columns, and specific pixel values, corresponds to a $N \times M$ matrix. We here apply L_1 normalization row-wise to the matrix, which yields a $N \times 1$ matrix. The $N \times 1$ matrix derived from each key-frame is then treated as a point in a N -dimensional space, which allows us to construct A k-D tree from these N -dimensional points. Using such data structure to conduct candidate searches can largely reduce computational time.

The construction process involves these key steps:

Choosing Splitting Axis: At each level of tree construction, the axis along which the data is split is determined. This decision can be based on criteria like alternating dimensions or selecting the axis with the highest variance.

Finding Splitting Value: After selecting the splitting axis, a splitting value is determined. This value is typically the median of the points projected onto the chosen axis.

Creating Subtrees: The data is divided into two subsets based on the splitting value. Subtrees are then created for each subset, and the construction process recurs on these subtrees.

To ensure the database's relevance, a regular update mechanism is employed. Every 50 seconds, the k-D Tree database is refreshed by reconstructing the k-D Tree using the latest point cloud frame. The newly obtained $N \times 1$ matrix is treated as a new N -dimensional point, and the k-D Tree is modified to incorporate this point while preserving its hierarchical structure.

The update procedure encompasses these main steps:

Finding Insertion Point: The existing k-D Tree is traversed to identify the appropriate leaf node where the new point should be inserted.

Expanding the Tree: Once the target leaf node is located, the k-D Tree is expanded to accommodate the new point. This ensures that the k-D Tree's balance and structure are maintained.

3.3.2. Similarity Calculation. As each point cloud key-frame was previously encoded into a N-dimensional point from the k-D tree database, we search k (e.g., k = 50) near neighbor point from the tree, each of which representing a possible loop closure candidate for each key-frame. We then calculate the similarity score between each candidate and the target key-frame. Here, we use the N×M matrix before encoding rather than the N×1 matrix for calculation on both candidate and target to ensure the accuracy of loop closure detection. Since angular partitioning is applied to each point cloud key-frame, our proposed BEV descriptor is extremely sensitive to azimuthal changes. To ensure a robust detection method, we must have to apply column-wise shifting to the candidate matrix in consideration of azimuthal angle variations. Since the matrix is N by M, we decide to shift the matrix for M times. B_o and B_k represent the matrix before and after each shift. To shift the matrix:

$$B_s(i, j) = B_o(i, j + k), \text{ where } k = 0, 1, 2, \dots, M - 1 \quad (24)$$

For each shift, we calculate the similarity score between the shifted candidate matrix and the target matrix for once. For the calculation of the similarity score, we decide that a higher similarity score indicates a higher similarity between two BEV descriptors. Thus, the similarity function is:

$$\mathbf{S}(B_t, B_c^k) = 1 - \sum_{i=1, j=1}^{i=N, j=M} \frac{\|B_t(i, j) - B_c^k(i, j)\|}{N \times M \times R_{max}} \quad (25)$$

where B_t represents the target descriptor and B_c^k represents the candidate descriptor after k shifts. Here, we mathematically derive the deviation of the shifted candidate descriptor from the target descriptor, and subtract the result from 1 to obtain a similarity score less than 1. After calculating similarity scores for all the shifted candidate descriptors, the highest similarity score will be assigned to the candidate. This assigned score will be used to compare with that of the other k candidates, and the final candidate C_L for loop closure is selected by highest assigned similarity score:

$$C_L = \arg \max \mathbf{S}(B_t, B_c) \quad (26)$$

3.3.3. ICP-based Geometric Verification. We use Iterative Closest Point (ICP) algorithm, a commonly-used algorithm for geometric alignment of two 3D point clouds, to geometrically verify whether a true loop closure has occur. The algorithm determines an optimal transformation that minimizes the disparities between corresponding points. However, unlike ICP used as an odometer, ICP used in our loop closure geometric verification finds the best transformation between the target key-frame and the candidate key-frame with the highest similarity score, rather than a target and a source point cloud. Apart from finding the translation vector and rotational matrix, we here compare the percentage of overlapping regions between two candidate point clouds (N_o) after transformation:

$$N_o = \frac{N_{overlap}}{N_{sum}} \times 100\% \quad (27)$$

where $N_{overlap}$ is number of overlapped points and N_{sum} is the number of total points, to a pre-set parameter σ (e.g., $\sigma = 50\%$) to determine whether a loop closure has occur. The percentage is obtained from dividing the numbers of overlapping points after transformation over the number of total points. If the percentage exceeds σ , a valid loop closure has occur between the two candidate point cloud frames.

3.3.4. Pose Graph Optimization. The pose graph consists of nodes that represent individual lidar point cloud scans. Two consecutive nodes, representing consecutive time steps, are linked by a transformation based on odometry information. This transformation includes a translation vector \mathbf{t} and a rotational matrix \mathbf{R} , typically obtained from LiDAR odometry.

The pose graph optimization process begins with an initial estimation of the poses of each LiDAR point cloud scan. Constraints are established between consecutive nodes based on odometry

transformations. Given consecutive nodes i and j , the transformation \mathbf{T}_{ij} from node i to node j is represented by the translation vector \mathbf{t}_{ij} and the rotational matrix \mathbf{R}_{ij} . The constraint equation is given by:

$$\mathbf{T}_{ij} = \begin{bmatrix} \mathbf{R}_{ij} & \mathbf{t}_{ij} \\ \mathbf{0}^T & 1 \end{bmatrix} \quad (28)$$

The goal of pose graph optimization is to minimize the discrepancy between the estimated poses and the constraints. The optimization objective can be defined as minimizing the sum of squared error terms between the estimated transformations and the constraint transformations:

$$\min \sum_{i,j} \|\mathbf{T}_{ij} - \mathbf{T}_{ij}^{\text{est}}\|_F^2 \quad (29)$$

where $\mathbf{T}_{ij}^{\text{est}}$ is the estimated transformation between nodes i and j .

Once a loop closure is detected between nodes i and k , a new constraint is added between these nodes. The transformation \mathbf{T}_{ik} is obtained using ICP, yielding translation \mathbf{t}_{ik} and rotation \mathbf{R}_{ik} .

$$\mathbf{T}_{ik} = \begin{bmatrix} \mathbf{R}_{ik} & \mathbf{t}_{ik} \\ \mathbf{0}^T & 1 \end{bmatrix} \quad (30)$$

After adding loop closure constraints, the optimization objective is updated to include these new constraints. The optimization problem is solved again to refine the poses of all nodes, considering both odometer and loop closure constraints.

4. Experiment

Table 1. Experimental parameters

Parameter	Value	Description
v	1 (m)	Plane voxel size
R_{max}	40 (m)	Maximum LiDAR range
N_a	20	Number of angular intervals
N_r	20	Number of radial segments
σ^1	0.01	Plane judgement threshold
σ^2	0.05	Plane judgement threshold
k	50	Number of loop closure candidates

Table 2. LiDAR comparison from each dataset

	Type	Price	FOV	Scan rate (pts/s)	Range (~0.80 reflectivity)	Point cloud density	Weight
Livox mid-360	Solid State	\$749	360° * 59°	200k	70 m	6 lines	265 g
Velodyne HDL-64E	Mechanical Spinning	\$75,000	360° * 26.8°	1.3M	120 m	64-lines	< 13000 g

Table 3. Selected dataset sequences for experiment

	KITTI				Private			
Sequence Index	00	02	05	08	ID2	OD2	OD3	OD4
Total Length (m)	3714	4268	2223	3225	621	957	1733	2333
# of Nodes	4541	4661	2761	4071	528	802	1196	1923
# of True Loops	790	309	493	332	275	84	559	426
Route Dir. on revisit	Same	Same	Same	Reverse	Reverse	Same	Same	Same

In this section, we evaluate our loop closure detection method on both open and private datasets with different scenarios (indoor and outdoor unstructured environment) to verify the effectiveness and adaptability of the method. In each experiment, we compare our method with state-of-the-art counterparts Scan Context [8] and VoxelMap proposed [19] by Yuan et al. on their default parameters. All experiments are carried out on the same system with an Intel i5-12600kf @ 3.7 GHz with 32 GB memory.

4.1. Dataset Preparation

We use the KITTI odometry dataset and a private dataset for the validation of our method. The two datasets are chosen considering both LiDAR and scenario diversity to ensure the method's adaptability under varying circumstances. Basic parameters of the two LiDARs used in both datasets are shown in Table 2. Characteristics of both datasets are summarized in Table 3.

1) KITTI odometry dataset: Developed by the Karlsruhe Institute of Technology (KIT) and the Toyota Technological Institute at Chicago (TTIC), KITTI odometry dataset provides a collection of synchronized sensor data in open urban environment, including grayscale stereo images, mechanical LiDAR scans, and accurate GPS/INS ground truth information. We evaluate our method on four sequences with the highest number of loop occurrence from the eleven sequences having the ground truth of pose (from 00 to 10), namely KITTI00, KITTI02, KITTI05, and KITTI08. While the other three sequences have loops in the same direction, only KITTI08 has reverse loops. LiDAR data from the dataset is obtained from a Velodyne 64-ray mechanical spinning LiDAR.

2) Private Dataset: We select four sequences from our dataset: one indoor and three outdoor sequences. The dataset was collected around summer of the year 2023, originating from a typical residential quarter within Minhang district of Shanghai. LiDAR and IMU data from the dataset are collected on a pre-built robotic vehicle as shown in Figure 4.(b), equipped with a HEXMAN MARK-2 chassis, a Livox Mid-360 solid state LiDAR and a built-in IMU. The indoor sequence, ID2, has a total length of 621 m, is recorded at the underground parking lot of the residential quarter. The outdoor sequence, OD2, has a total length of 957 m, is recorded at the interior of the residential quarter; the rest two outdoor sequences, having a total length of 1423 m and 1955 m respectively, are both recorded at the peripheral area of the residential quarter. The trajectories of all sequences are shown in Figure 5.

We set the experimental parameters as shown in Table 1. For KITTI dataset, we evaluate the performance of our proposed method on loop closure detection by the precision-recall curve, compared with Scan Context, another state-of-art point cloud descriptor. Then, given the constraints from detected loop closures, accuracies of the trajectories before and after pose graph optimization will be compared. Our private dataset will not be evaluated through the above two measures due to the lack of ground truth; it will rather serve as an additional validation of the adaptability of our proposed method under varying LiDAR types and environments.



Figure 4. Recording platform with sensors (a) from KITTI dataset [20] and (b) our private dataset

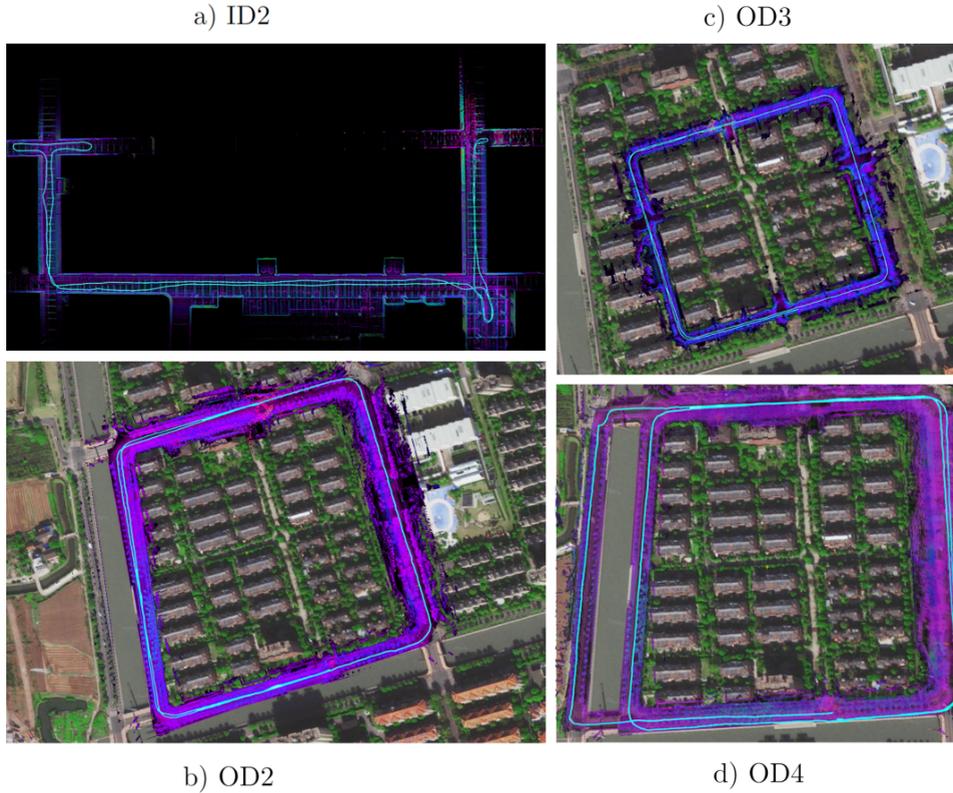


Figure 5. Trajectories of the four sequences, aligned with their corresponding point cloud and satellite image. In (a), the path of sequence ID2 is a semi-loop with its reverse loop. In (b) and (c), the paths of sequences OD2 and OD3 are all full loops. In (d), the path of sequence OD4 consists of a smaller loop and a larger loop.

4.2. Precision Recall Evaluation

The precision-recall curves for both our proposed BEV descriptor and Scan Context on KITTI odometry sequences are illustrated in Figure 7. From the results, our method outperforms Scan Context in most sequences. Both methods have a similar curve shape, yet our method has a slight increase in recall rate in all sequences except KITTI02. Like our method, Scan Context is sensitive to azimuthal variances; however, it mainly relies on vertical structure information as its bin encoding uses a maximum height, limiting its performance when the vertical height in the surrounding varies a little. In comparison, our approach relies on plane projection and replaces maximum height with maximum point-to-plane distance. This allows more variances in vertical structure and increases robustness of the descriptor against changes in pitch angle, ultimately enhancing its overall performance. According to Figure 6, the performance of our method on sequence KITTI02 is poorer than on any other sequence, probably due to sparse structures or occlusion issue of the loop closure in the scene [9] [21]. An example scene in the sequence is shown in Figure 6.



Figure 6. An example scene in KITTI02 sequence. Such highway scenes usually contain less geometric or structural information than urban scenes.

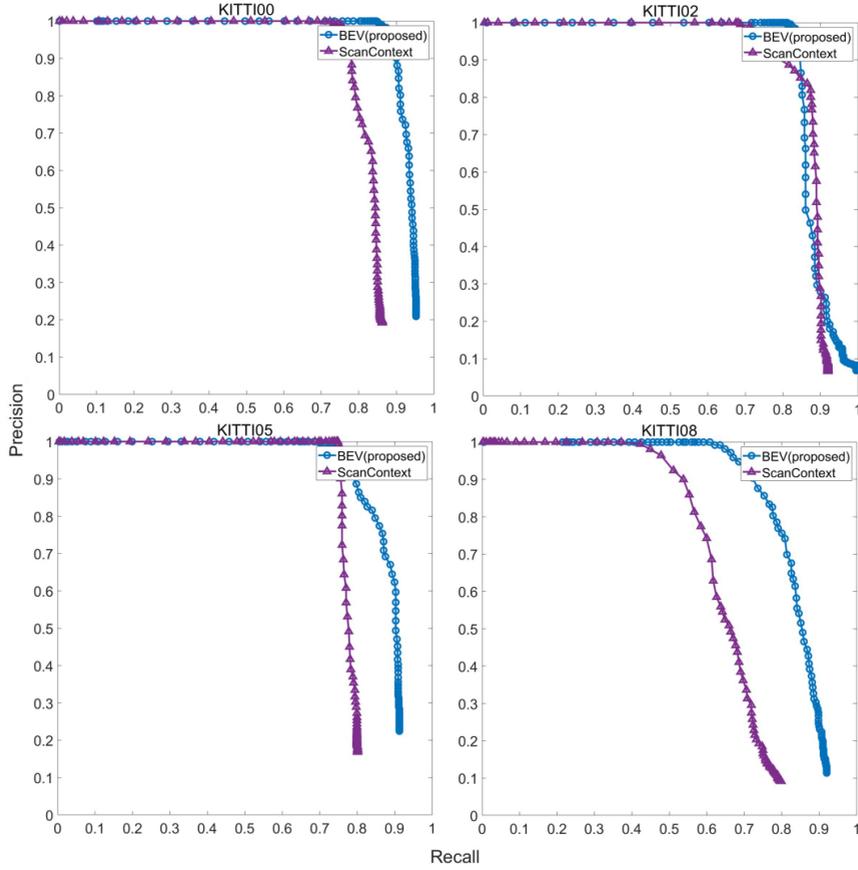


Figure 7. Precision-recall curves on four KITTI odometry sequences.

To further compare the detection performance, we calculate the F1 scores on both methods over four sequences. The results in Table 4, consistent with the precision-recall curves, shows a higher score on our method than on Scan Context. In addition, our method improves the most over sequence KITTI08, indicating our method can effectively detect reverse loop closures with higher robustness to azimuthal changes.

Table 4. F1 scores over four KITTI odometry sequences

Method	KITTI00	KITTI02	KITTI05	KITTI08
Scan Context	0.8567	0.8509	0.8563	0.6731
BEV(proposed)	0.9247	0.9024	0.8479	0.8057

4.3. Trajectory Accuracy Evaluation

To demonstrate the improvement of mapping quality through our loop closure detection method, we use VoxelMap [19], a high accuracy odometer, to generate the trajectories of the four KITTI sequences. In contrast, we further use the constraints from detected loop closures in the sequences to correct the trajectories. As shown in Figure 8, we compare the original trajectories of four sequences mapped by VoxelMap, with the trajectories after correction by our method. The distance error between each trajectory and the ground truth is then calculated. From the results, the accuracies of the trajectories after correction by our method has notably improved. No signs of huge drifts or deviations from the original trajectories are shown in results, hence indicating that loop closures in the four sequences can be accurately detected through our proposed method and utilized to improve the quality and precision of long-term mapping. It is worth noticing that in sequence KITTI02, the start of the trajectory after

correction had experience a comparably large drift, probably due to the sparse structures on the highway scene as key points extracted in such scenes are scarce. This finding of a relatively poor performance on KITTI02 is consistent with the results in previous section.

The average Root Mean Square Error (RMSE) of each trajectories is summarized in Table 5. The trajectories after correction by our approach produce a reduced average RMSE. The average RMSE of KITTI02 trajectory after correction decreases less significantly than that of the other three sequences. Therefore, our loop closure detection method can be integrated into SLAM context to provide accurate loop closure detection under varying conditions.

Table 5. Average RMSE of trajectories before and after correction Unit: m

Method	KITTI00	KITTI02	KITTI05	KITTI08
VoxelMap	3.2293	7.7210	1.7562	4.5893
BEV(proposed)	0.8247	5.2420	0.5246	2.8239



Figure 8. Trajectories (left) generated by VoxelMap compared to trajectories (right) after correction by our approach. The color bar on the right indicates distance error between the generated trajectory and the ground truth. Unit: m

4.4. Runtime Evaluation

We record the computation time on KITTI05 for our method and Scan Context. For Scan Context, we test with its provided Python API with the default parameter. As shown in Figure 9, both methods' computational time increases with the number of key-frames, possibly due to the construction of k-d tree for retrieval in both methods. Overall, our method has a more consistent computational time over time and a less computational time per frame than Scan Context. The difference on how we test the two methods (generically on ours and using an API for Scan Context) might account for our superiority, but an average computational time of less than 100 ms per frame is sufficient to verify our proposed descriptor as an efficient descriptor for loop closure detection.

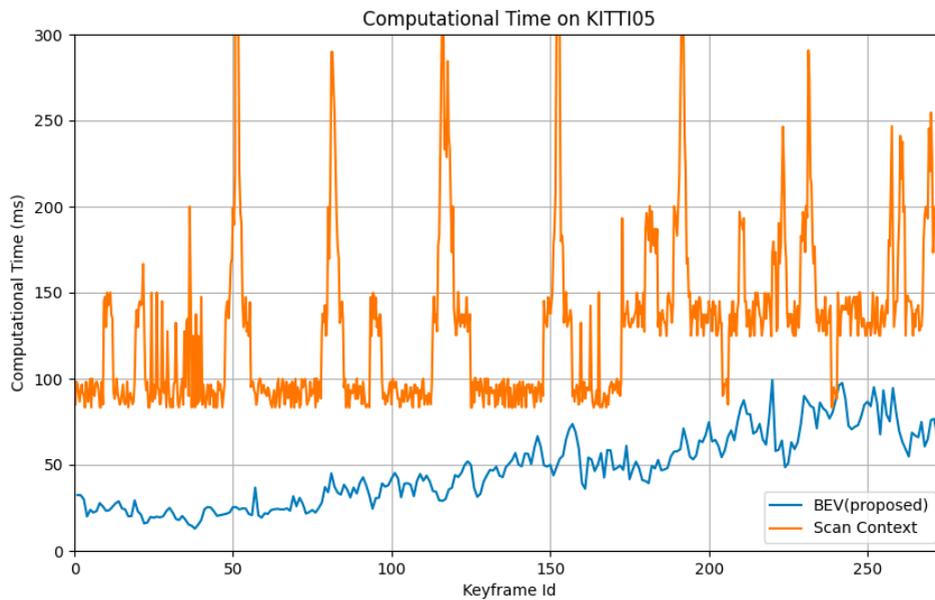


Figure 9. Computational time on KITTI05 of our proposed BEV method

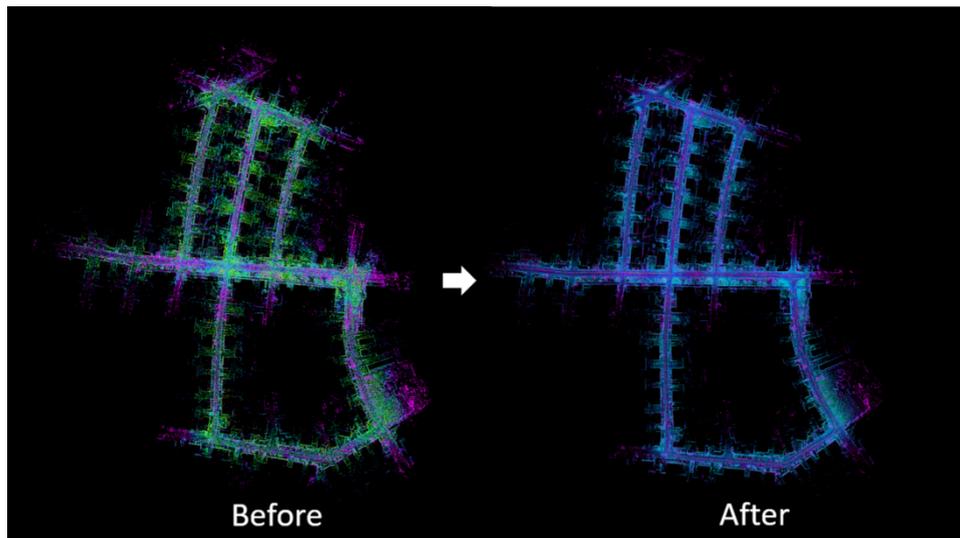


Figure 10. KITTI05 point cloud map before and after correction by our approach

5. Conclusion

This paper proposes a adaptive and robust BEV point cloud descriptor. An effective BEV image generation process based on plane detection and projection and space partition is proposed to extract and preserve geometric information from each point cloud key-frames. Through radial and azimuthal partition, our descriptor exhibits resilience against changes in azimuthal and pitch angles. To speed up the querying and retrieval process, we further construct a k-d Tree database for searching possible loop closure candidates and compute similarity scores between the candidates. Compared with Scan Context, a state-of-art global descriptors, our method not only outperforms Scan Context in KITTI dataset but also demonstrate a great robustness and adaptability to both outdoor and indoor environment and to different, less costly LiDAR types.

References

- [1] Mathieu Labbe and François Michaud. Online global loop closure detection for large-scale multi-session graph-based slam. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2661–2666. IEEE, 2014.
- [2] Saba Arshad and Gon-Woo Kim. Role of deep learning in loop closure detection for visual and lidar slam: A survey. *Sensors*, 21(4):1243, 2021.
- [3] Mathieu Labbe and François Michaud. Online global loop closure detection for large-scale multi-session graph-based slam. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2661–2666. IEEE, 2014.
- [4] Ben Glocker, Jamie Shotton, Antonio Criminisi, and Shahram Izadi. Real-time rgb-d camera relocalization via randomized ferns for keyframe encoding. *IEEE transactions on visualization and computer graphics*, 21(5):571–583, 2014.
- [5] Andrés C Jiménez, Vicente García-Díaz, and Sandro Bolaños. A decentralized framework for multi-agent robotic systems. *Sensors*, 18(2):417, 2018.
- [6] Stephanie Lowry, Niko Sünderhauf, Paul Newman, John J Leonard, David Cox, Peter Corke, and Michael J Milford. Visual place recognition: A survey. *IEEE transactions on robotics*, 32(1):1–19, 2015.
- [7] Wei Xu, Yixi Cai, Dongjiao He, Jiarong Lin, and Fu Zhang. Fast-lid2: Fast direct lidar-inertial odometry. *IEEE Transactions on Robotics*, 38(4):2053–2073, 2022.
- [8] Giseop Kim and Ayoung 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)*, pages 4802–4809. IEEE, 2018.
- [9] Chongjian Yuan, Jiarong Lin, Zuhao Zou, Xiaoping Hong, and Fu Zhang. Std: Stable triangle descriptor for 3d place recognition. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1897–1903. IEEE, 2023.
- [10] Huan Yin, Xuecheng Xu, Sha Lu, Xieyuanli Chen, Rong Xiong, Shaojie Shen, Cyrill Stachniss, and Yue Wang. A survey on global lidar localization. arXiv preprint arXiv:2302.07433, 2023.
- [11] Mark Cummins and Paul Newman. Fab-map: Probabilistic localization and mapping in the space of appearance. *The International journal of robotics research*, 27(6):647–665, 2008.
- [12] Adrien Angeli, David Filliat, Stéphane Doncieux, and Jean-Arcady Meyer. Fast and incremental method for loop-closure detection using bags of visual words. *IEEE transactions on robotics*, 24(5):1027–1037, 2008.
- [13] Dorian Gálvez-López and Juan D Tardos. Bags of binary words for fast place recognition in image sequences. *IEEE Transactions on Robotics*, 28(5):1188–1197, 2012.
- [14] Christoffer Valgren and Achim J Lilienthal. Sift, surf and seasons: Long-term outdoor localization using local features. In *3rd European conference on mobile robots, ECMR’07, Freiburg, Germany, September 19-21, 2007*, pages 253–258, 2007.
- [15] Michael J Milford and Gordon F Wyeth. Seqslam: Visual route-based navigation for sunny summer days and stormy winter nights. In *2012 IEEE international conference on robotics and automation*, pages 1643–1649. IEEE, 2012.
- [16] Fei Han, Xue Yang, Yiming Deng, Mark Rentschler, Dejun Yang, and Hao Zhang. Sral: Shared representative appearance learning for long-term visual place recognition. *IEEE Robotics and Automation Letters*, 2(2):1172–1179, 2017.
- [17] Renaud Dubé, Daniel Dugas, Elena Stumm, Juan Nieto, Roland Siegwart, and Cesar Cadena. Segmatch: Segment based place recognition in 3d point clouds. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5266–5272. IEEE, 2017.
- [18] John Crassidis and F Markley. Attitude estimation using modified rodrigues parameters. In *NASA Conference Publication*, pages 71–86. NASA, 1996.
- [19] Chongjian Yuan, Wei Xu, Xiyuan Liu, Xiaoping Hong, and Fu Zhang. Efficient and probabilistic adaptive voxel mapping for accurate online lidar odometry. *IEEE Robotics and Automation Letters*, 7(3):8518–8525, 2022.

- [20] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. *In 2012 IEEE conference on computer vision and pattern recognition* pages 3354–3361. IEEE, 2012.
- [21] Gang Wang, Xiaomeng Wei, Yu Chen, Tongzhou Zhang, Minghui Hou, and Zhaohan Liu. A multi-channel descriptor for lidar-based loop closure detection and its application. *Remote Sensing*, 14(22):5877, 2

Acknowledgments

I would like to thank for the support from my school Shanghai High School International Division, and from my advisor, Mr. Li. Note that the topic of this essay was inspired by a project on autonomous driving I've completed in last year, which primarily focuses on point cloud-based localization and mapping in hospitals. The previous experiences on such topics motivate me to conduct similar researches, and after discussion with my advisor, who is also my Computer Science teacher, Mr. Li, we finally decided the topic. The accumulation of experience from the last project enables me to complete the whole research independently. Devices involved in the process of data collection, including LiDAR sensors and the experimental vehicle, are all self-purchased. Data are collected and analyzed solely by myself. And finally, I would like to give my special thanks to my adviser for his charge-free guidance.