TripletLoc: One-Shot Global Localization Using Semantic Triplet in Urban Environments

Weixin Ma, *Graduate Student Member, IEEE*, Huan Yin, *Member, IEEE*, Patricia J. Y. Wong, Danwei Wang, *Life Fellow, IEEE*, Yuxiang Sun, *Member, IEEE*, and Zhongqing Su

***Abstract*—This study presents a system, TripletLoc, for fast and robust global registration of a single LiDAR scan to a large-scale reference map. In contrast to conventional methods using place recognition and point cloud registration, TripletLoc directly generates correspondences on lightweight semantics, which is close to how humans perceive the world. Specifically, TripletLoc first respectively extracts instances from the single query scan and the large-scale reference map to construct two semantic graphs. Then, a novel semantic triplet-based histogram descriptor is designed to achieve instance-level matching between the query scan and the reference map. Graph-theoretic outlier pruning is leveraged to obtain inlier correspondences from raw instance-to-instance correspondences for robust 6-DoF pose estimation. In addition, a novel Road Surface Normal (RSN) map is proposed to provide a prior rotation constraint to further enhance pose estimation. We evaluate TripletLoc extensively on a large-scale public dataset, HeliPR, which covers diverse and complex scenarios in urban environments. Experimental results demonstrate that TripletLoc could achieve fast and robust global localization under diverse and challenging environments, with high memory efficiency.**

***Index Terms*—Global localization, pose estimation, graph theory, semantic triplet, autonomous vehicles.**

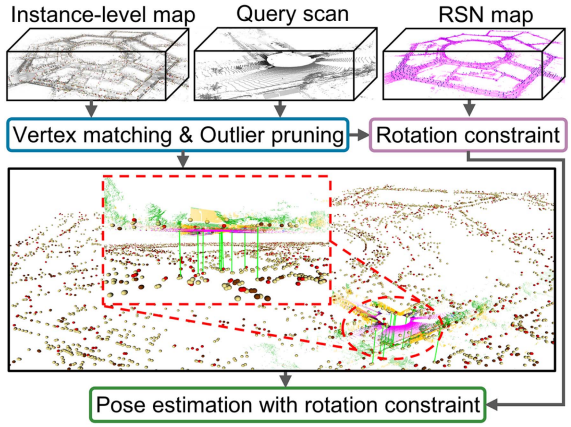

Fig. 1. TripletLoc Framework. Semantic graphs are constructed to represent both the single query scan and the global reference map. Vertex descriptors are then extracted from the graphs to build instance-to-instance correspondences. Graph-theoretic outlier pruning and rotation constraint from the RSN map are integrated for the 6-DoF pose estimation.

I. Introduction

GLOBAL localization refers to localizing a robot in a prior database or map with less or without initial guess. Global Navigation Satellite System (GNSS) is one of the most popular solutions. The performance of GNSS might be degraded because of signal occlusions and multi-path effects in urban canyons or undergrounds. Researchers tend to use onboard measurements to locate the robot within a prior database or map to release the need for GNSS in such environments. LiDAR-based methods have demonstrated good robustness and accuracy under diverse conditions, such as changes in illumination or weather, showing great potential for global localization [1], [2].

A popular global localization framework is based on a scan-to-scan loop closure scheme, which achieves global localization by comparing similarities of descriptors (i.e., place recognition) and estimating relative poses between the query scan and reference scans (i.e., scan-wise registration) [3], [4], [5], [6], [7]. However, some of them might not be efficient in memory when the scale of the reference map becomes larger, where thousands of discrete and dense point clouds are involved. Such map formulation is also hard to maintain in long-term localization, considering the discrete characteristics of reference scans.

Recent works [8], [9], [10] use a scan-to-map scheme based on lightweight semantics in environments. These methods directly register the single query scan to the large-scale reference map, where semantic instances are used to reduce memory and computational complexity of pose estimation. All-to-all [8], [10] and condition-meeting [9] strategies have been proposed to build instance-to-instance correspondences. However, they may still be inefficient at times when addressing the registration

Received 21 September 2024; accepted 8 December 2024. Date of publication 26 December 2024; date of current version 6 January 2025. This article was recommended for publication by Associate Editor Y. Zhou and Editor A. Banerjee upon evaluation of the reviewers' comments. This work was supported in part by Hong Kong Research Grants Council under Grant 15222523, in part by Hong Kong Innovation and Technology Commission under Grant K-BBY1, in part by the National Research Foundation (NRF), Singapore, under the NRF Medium Sized Centre scheme (CARTIN), ASTAR under Grant M22NBK0109, in part by the NRF, Singapore and Maritime and Port Authority of Singapore under Grant SMI-2022-MTP-04, and in part by the City University of Hong Kong under Grant 9610675. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not reflect the views of National Research Foundation, Singapore. *(Corresponding authors: Yuxiang Sun; Zhongqing Su.)*

Weixin Ma and Zhongqing Su are with the Department of Mechanical Engineering, The Hong Kong Polytechnic University, Hong Kong (e-mail: weixin.ma@connect.polyu.hk; zhongqing.su@polyu.edu.hk).

Huan Yin is with the Department of Electronic and Computer Engineering, The Hong Kong University of Science and Technology, Hong Kong (e-mail: eehyin@ust.hk).

Patricia J. Y. Wong and Danwei Wang are with the School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore 639798 (e-mail: ejywong@ntu.edu.sg; edwwang@ntu.edu.sg).

Yuxiang Sun is with the Department of Mechanical Engineering, City University of Hong Kong, Hong Kong (e-mail: yx.sun@cityu.edu.hk).

Digital Object Identifier 10.1109/LRA.2024.3523228

problem, particularly with a large number of correspondences in large-scale environments.

To address the above issues, we propose TripletLoc in this study, which is a fast, efficient, and robust solution for LiDAR-based global localization. Similarly, TripletLoc also converts the single query scan and the entire large-scale reference map to a compact instance-level semantic graph, which has been used in the current methods [7], [9], [11], [12] to embed environment layouts. Instance-to-instance correspondences are then generated based on a novel semantic triplet-based histogram descriptor, which embeds semantic, topological, and geometric cues from the scene. Based on this descriptive descriptor, a simple yet effective top-k matching strategy is used to guarantee running speed and computational feasibility of the scan-to-map based scheme. At the back end, graph-theoretic pruning is used to improve the pose estimation robustness against outlier correspondences. In addition, a RSN map is proposed to provide a prior rotation constraint for better pose estimation. The framework of TripletLoc is shown in Fig. 1. Our code is open-sourced.[1] Overall, our contributions are summarized as follows:

1) We develop a system that could efficiently and robustly localize a robot globally in large-scale urban environments using only one query scan without requiring a sequence of onboard scans and odometry.
2) We extend and improve the semantic triplet-based histogram descriptor [7] to obtain robust and efficient instance-to-instance correspondences.
3) We propose a novel RSN map to provide a prior rotation constraint for pose estimation to further enhance registration performance.

II. Related Work

LiDAR-based global localization can be generally categorized into two branches: one-shot-based and sequence-based methods [1]. Sequence-based methods can be further divided into retrieval and filtering methods, where relative poses between consecutive frames are needed. By accumulating consecutive frames as a submap, retrieval methods achieve global localization by identifying whether the current submap has been visited in a prior submap-based database. Filtering methods usually locate a robot by iteratively estimating the pose of the robot, commonly known as Monte Carlo Localization [13]. Differently, one-shot global localization only uses one single frame without requiring relative poses. This paper focuses on one-shot methods. We review existing related works in two streams: place recognition (PR)-based methods and registration-based methods.

A. Place Recognition-Based Methods

PR-based methods usually first achieve coarse localization by identifying whether the current query frame has been visited in a prior database. The relative metric pose is then obtained by registering the query frame to the retrieved frame using raw point clouds or local point features [1]. Global descriptors are usually used for the retrieval process in PR-based methods, including handcrafted feature-based and data-driven-based descriptors. In early works, handcrafted local features are usually aggregated into global descriptors for PR [14]. Instead of using local features, global descriptors extracted directly from LiDAR points have become another popular solution for PR in recent years [1]. Scan Context [3] embeds geometric information from a 3D point cloud as a 2D matrix global descriptor using bird-eye-view (BEV) projection. Similarly, SSC [5], Intensity Scan Context [4], RING++ [15], and LiDAR-Iris [6] all use BEV projection to extract their global descriptors. More recently, graph structure has been used in many methods to embed environment layouts [7], [11], [12], [16]. Most of them rely on the clustered semantic instances to build semantic graphs [7], [11], [12]. Similarities between semantic graphs are then used for place recognition. Differently, key points are first extracted from the point cloud and then be used to construct triangle-based graphs [16]. A voting strategy is proposed to select matched frames from a prior database for the query frame. In stead of using handcrafted descriptors, data-driven-based methods extract global descriptors from point clouds using deep neural networks [17], [18], [19].

B. Registration-Based Methods

Unlike PR-based methods, which use key-frame-based retrieval, registration-based methods directly register the query scan to the reference map. However, in global localization, the scale of the reference map is much larger than that for a single point-cloud scan, which usually covers thousands of square meters. It is computationally intractable to register a query LiDAR scan directly to a reference map using 3D point-level correspondences. Alternatively, several researchers extract instances from environments to build correspondences [8], [9], [10]. All-to-all strategy is used to build correspondences in [10], that is, all the possible pairs of an instance in the query scan and an instance in the reference map. Similarly, all-to-all correspondences are used in [8]. However, an instance from the query scan and an instance from the reference map will be paired only when their classes are the same. In [9], correspondences are generated when triangle descriptors for query scan and global map satisfy some designed conditions (i.e., same hash key, semantic classes, and similar variance matrices). Based on the built correspondences, all the methods mentioned above use the graph-theoretic outlier pruning to select inlier correspondences for pose estimation. These methods might build a large number of correspondences in large-scale environments, making it slow or even computationally intractable for outlier pruning.

C. Differences From Previous Studies

Our previous work, Triplet-Graph [7], is a typical scan-to-scan based approach. Each scan includes its surrounding instances, which might redundantly save some instances multiple times across different keyframes. To improve memory efficiency, TripletLoc employs a scan-to-map framework, where each instance is saved only once in the prior map. Unlike the RANSAC-based pose estimation in Triplet-Graph, TripletLoc enhances robustness to outliers using graph-theoretic pruning and Truncated Least Squares (TLS) based registration. Compared to other registration methods, TripletLoc ensures computational feasibility for robust pose estimation, regardless of the reference map's scale. To achieve this, we design an informative vertex descriptor that integrates semantic, topological, and geometric information, rather than relying solely on clustered instances. Additionally, we propose a novel RSN map to provide a prior

[1] [Online]. Available: https://github.com/Weixin-Ma/TripletLoc

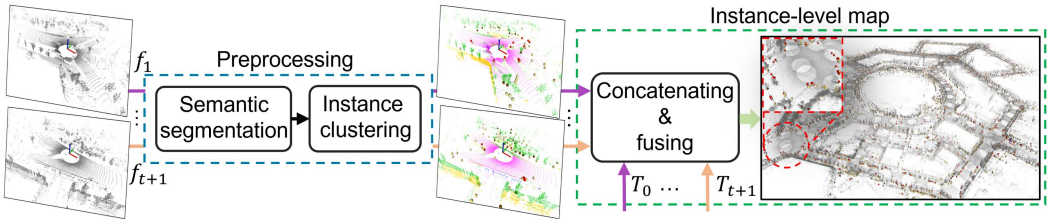

Fig. 2. The construction pipeline of the instance-level map. Semantic segmentation is first performed on each LiDAR scan, followed by instance clustering. The clustered instances of all scans are then concatenated and fused.

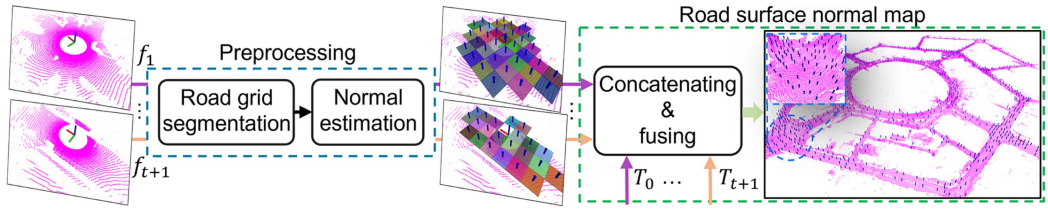

Fig. 3. The construction pipeline of the RSN map. Road grid segmentation is first performed on road points, followed by road surface normal estimation. The road-grid-based surface normal of all scans are then concatenated and fused.

rotation constraint to improve registration robustness. Using RSN overcomes the limitation of previous methods that discard road information, showing significant potential for urban global localization.

III. The Proposed Method

Our proposed one-shot global localization method, TripletLoc, is generally divided into three parts: instance-level map and RSN map construction, vertex descriptor extraction and matching, and robust 6-DoF pose estimation, as shown in Fig. 1.

A. Instance-Level Map and RSN Map

Due to the large scale of reference maps, directly localizing the vehicle using 3D point-level correspondences is computationally impractical. Therefore, lightweight and compact maps are essential. As shown in Fig. 3, we represent the scene at the instance level for better efficiency. To enhance localization robustness, we also propose a novel RSN map for prior rotation constraints in 6-DoF pose estimation.

1) Instance-Level Map: Given a sequence of LiDAR scans and their relative poses, we first perform semantic segmentation on each scan using the pre-trained SPVNAS [20] without fine-tuning. Instead of using all object classes, we only use trunk, pole, and traffic sign, because they are more stable in long-term localization. 3D points with the same class label are then clustered into different object groups for each scan using the Euclidean Cluster algorithm from the Point Cloud Library [21]. Each clustered group has its own ID, class label, geometric centroid (i.e., *xyz*-coordinate), and corresponding point number. All the clustered instances from different scans are then transformed and concatenated into the same coordinate system based on relative poses between different scans. Here, we directly use the ground truth of robot poses provided by the datasets. Since some instances might be observed across different scans, it would lead to redundancy and duplication of instances in the map. So, we further fuse instances whose distances are less than 0.5 m between each other into a new single instance. The geometric centroid for the new instance is the average of the geometric centroids of the fused instances. The class label for the new instance is set the same as the instance of which the corresponding point number is the largest across the fused instances.

2) RSN Map: Based on the semantic segmentation results from SPVNAS, we divide 3D road points into discrete road grids (10 m$\times$ 10m). Within each grid, if the number of 3D points exceeds a pre-defined threshold (1000in our case), we apply RANSAC to segment the road plane from these points. We then use the normal of this segmented road plane to approximate the road surface normal. This approximation is reasonable because roads are typically flat in local area in modern urban environments. The center of each road grid serves as the starting point for this normal. Similar to the instance-level map, we concatenate the road-grid-based surface normals from different scans. We then fuse normals with starting points less than a threshold distance (3.5m in the experiments) into a single normal. To achieve the normal fusion, we calculate the average of the normal vectors element-wise and normalize the result, noted as $\underrightarrow{n} \in \mathbb{R}^3$. The starting point of the new normal is the average of the starting points of the fused normals. We also compute the relative angle between each fused normal and the new normal. These angles (differences) are used to determine the standard deviation, $\sigma_{\underrightarrow{n}} \in \mathbb{R}$, which reflects the uncertainty of the road surface normal estimation ($\underrightarrow{n}$) of the local area.

B. Vertex Descriptor Extraction and Matching

Reliable correspondences between the query scan and the reference map are fundamental to accurate pose estimation but become increasingly challenging as larger maps introduce repeated elements and sub-structures. In the work, we extend the vertex descriptor proposed in our previous work [7] to establish

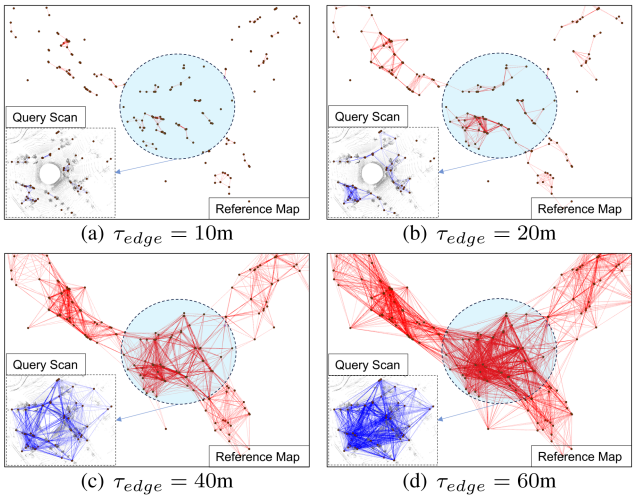

Fig. 4. Semantic graphs of a query scan and the reference map with varying τ_{edge}. Only part of the map is displayed. The blue circle refers to the corresponding area in the map at the same location as the query scan.

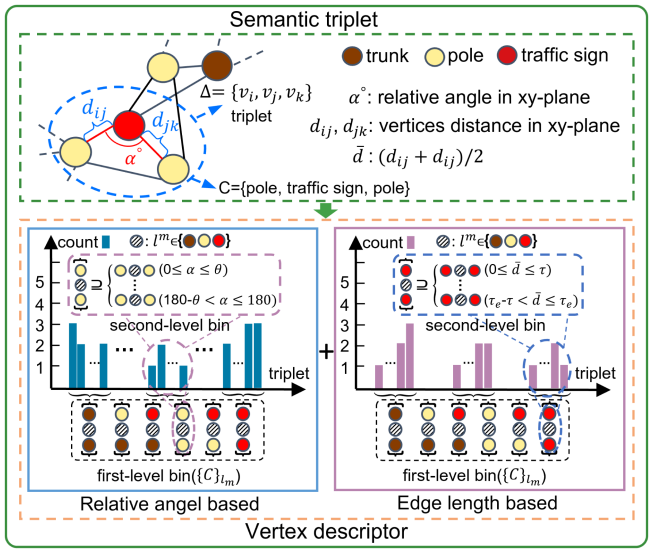

Fig. 5. Vertex descriptor extraction. A semantic graph is first constructed to represent the input LiDAR point cloud. The blue dotted ellipse shows a sample triplet. The triplet semantic histogram-based descriptor is then employed to represent the vertices in the graph.

more robust correspondences between the query scan and the instance-level map.

1) Semantic Graph Construction: Given a set of clustered instances $\{I_i\}$, we represent them as an undirected graph $\boldsymbol{G} = \langle \boldsymbol{V}, \boldsymbol{E} \rangle$. $\boldsymbol{V} = \{v_i\}$ is the set of vertices, in which v_i refers to an individual instance I_i. $\boldsymbol{E} = \{e_{ij}\}$ is the set of edges that connect two different vertices, in which $e_{ij} = \langle v_i, v_j \rangle$ is the edge connecting vertices v_i and v_j. Two instances are connected when their distance is less than a pre-defined threshold τ_{edge}. Intuitively, only partial instances in the reference map can be observed by the query scan. When τ_{edge} is larger, edges that connect instances distributed along the margin of the query scan, are more different from edges that connect the same instances in the reference map (see Fig. 4). Therefore, we use a smaller threshold (i.e., 20m) to reduce such differences.

2) Vertex Descriptor Extraction: In our previous work, Triplet-Graph [7], a triplet-based semantic histogram descriptor is proposed to describe vertices in a semantic graph. Specifically, given a vertex v_j, Triplet-Graph embeds topological (i.e., semantic combination) and geometric (i.e., relative angle) information of $\{\Delta\}_{v_j}$ from the graph. All the information is encoded as a histogram descriptor, as shown in the blue box in Fig. 5. The histogram can be easily converted as a $N_1 \times N_2$ matrix, denoted as $Des^{\alpha}_{v_j}$. We recommend reading our previous study [7] for the notations and the details of extraction process.

Only relative angles between instances are embedded in the second-level bin of $Des^{\alpha}_{v_j}$. When two triplets have the same class combination and close relative angle, they belong to the same bin of $Des^{\alpha}_{v_j}$. However, in practical situations, these two triplets might be very different in the length of the edges. Therefore, to better document these difference, we extend $Des^{\alpha}_{v_j}$ by explicitly embedding lengths of edges from triplets at the same time. Specifically, given a triplet Δ_{ijk} (as shown in the blue dotted ellipse at the top of Fig. 5, we first calculate distance d_{ij}/d_{jk} between vertex v_i/v_j and v_j/v_k in xy-plane (i.e., only x and y coordinates are used). Then we calculate the average edge length of Δ_{ijk}, denoted as $\bar{d} = (d_{ij} + d_{jk})/2$. Since two vertices are connected only when their distance is less than τ_{edge}, we have $\bar{d} \le \tau_{edge}$. Similar to $Des^{\alpha}_{v_j}$, a histogram based on two-level bins is used to embed the edge length information (i.e., $\bar{d}$) from all the triplets that use vertex v_j as the middle vertex in the graph. An example of the histogram descriptor can be found in the purple box in Fig. 5. The N_1 first-level bins are exactly the same as those for $Des^{\alpha}_{v_j}$. For each first-level bin $C \in \{C\}_{l_m}$ (please refer to [7] for more details about C and $\{C\}_{l_m}$), we further divide it into N_3 second-level bins. Each second-level bin has the same combination as C, but with a different range of average edge length. As shown in the blue dotted box in the right sub-figure of Fig. 5, we have $N_3 = \tau_{edge}/\tau$, in which τ ($\tau = 0.5$ m) is the interval of the range of average edge length. Note that τ should be divisible by τ_{edge}. The histogram can be also easily converted to a $N_1 \times N_3$ matrix, denoted as $Des^{d}_{v_j}$.

The final vertex descriptor for vertex v_j is the combination of $Des^{\alpha}_{v_j}$ and $Des^{d}_{v_j}$. To simplify the notation, we concatenate $Des^{\alpha}_{v_j}$ and $Des^{d}_{v_j}$ into a single $N_1 \times (N_2 + N_3)$ matrix by appending $Des^{d}_{v_j}$ on the right of $Des^{\alpha}_{v_j}$, denoted as Des_{v_j}.

3) Vertex Matching: Given semantic graph $\boldsymbol{G_{que}}$ of a query LiDAR scan and $\boldsymbol{G_{ref}}$ of the reference map, we use cosine similarity to measure the similarity between vertices [7]. Only vertices with the same class label are compared, for example, trunks in $\boldsymbol{G_{que}}$ are only compared with trunks in $\boldsymbol{G_{ref}}$. The cosine similarity can be calculated as:

$$\mathrm{Sim}\left(v_j^{que}, v_t^{ref}\right) = \frac{\sum Des_{v_j^{que}} \cdot Des_{v_t^{ref}}}{\left\| Des_{v_j^{que}} \right\|_F \times \left\| Des_{v_t^{ref}} \right\|_F}, \quad (1)$$

in which Sim is short for similarity, the dot $\cdot$ is the element-wise multiplication on two matrices, $\| \cdot \|_F$ is the Frobenius norm of a matrix, $\sum$ is the summation of all elements of a matrix, v_j^{que} and v_t^{ref} are respectively arbitrary vertices in $\boldsymbol{G_{que}}$ and $\boldsymbol{G_{ref}}$ with the same class label. For a vertex v_j^{que} in $\boldsymbol{G_{que}}$, we chose the top-k (25 in our implementation) vertices in $\boldsymbol{G_{ref}}$ that have highest similarities between v_j^{que} as the matching result for v_j^{que}. We then repeat the top-k matching for every

vertex in $\boldsymbol{G_{que}}$ to get a raw set of instance-wise correspondences $\mathcal{A}_{raw}$. K-D tree is created to accelerate the matching process. Specifically, given two normalized vectors $\mathbf{a}$ and $\mathbf{b}$, their cosine similarity $cos(\mathbf{a}, \mathbf{b})$ and euclidean distance $dis(\mathbf{a}, \mathbf{b})$ follows: $cos(\mathbf{a}, \mathbf{b}) = 1 - 0.5dis(\mathbf{a}, \mathbf{b})^2$. Therefore, for normalized $Des_{v_j^{que}}$ and $Des_{v_t^{ref}}$, the top-k matches with highest cosine similarities are the same as the top-k matches with smallest euclidean distance.

C. Pose Estimation

In scan-to-map-based global localization with large maps, achieving outlier-free correspondences is nearly unattainable, which poses significant challenges for accurate pose estimation. In this work, we employ graph-theoretic outlier pruning to minimize outliers and introduce a novel prior rotation constraint to enhance the robustness of pose estimation.

1) Graph-Theoretic Outliers Pruning: Theoretically, every vertex in $\boldsymbol{G_{que}}$ has at most one corresponding vertex in $\boldsymbol{G_{ref}}$. However, redundant correspondences can usually involve more inliers for better 6-DoF pose estimation, so we set k larger than 1. As a result, there are many outliers in $\mathcal{A}_{raw}$, especially when k comes larger. These outliers greatly increase the complexity of the 6-DoF pose estimation. To solve this problem, we prune outliers in $\mathcal{A}_{raw}$ by searching the Maximum Clique (MCQ) of the consistency graph constructed from $\mathcal{A}_{raw}$ [22], and the PMC library [23] is used to solve the searching problem. The filtered correspondence set is denoted as $\mathcal{A}$. Noted that different types of semantics instances are treated uniformly during the MCQ searching.

2) Point-to-Point Registration: After the outliers pruning, we estimate the unknown transformation between the query LiDAR scan and the instance-level map. Let $\boldsymbol{T} = [\boldsymbol{R}, \boldsymbol{t}]$ be the ground truth of the unknown transformation, in which $\boldsymbol{R} \in SO(3)$ and $\boldsymbol{t} \in \mathbb{R}^3$. Based on $\mathcal{A}$, we can easily get a set of vertex pairs $\mathcal{B}$, $\{(v_j^{que}, v_t^{ref})\}$. Each correct pair can be associated by $\overline{v}_t^{ref} = \boldsymbol{R}\overline{v}_j^{que} + \boldsymbol{t} + \boldsymbol{\epsilon}_{jt}$. $\boldsymbol{\epsilon}_{jt}$ models the measurement noise, $\overline{v}_t^{ref}$ and $\overline{v}_j^{que}$ are coordinates for vertex v_t^{ref} and v_j^{que}, respectively.

Although we have conducted outliers pruning already, there are inevitably still some potential outliers left in $\mathcal{A}$. To further improve the robustness to outliers, we follow [24] to calculate the estimation of $\boldsymbol{T}$, $\hat{\boldsymbol{T}} = [\hat{\boldsymbol{R}}, \hat{\boldsymbol{t}}]$, as a TSL registration problem:

$$\hat{\boldsymbol{R}}, \hat{\boldsymbol{t}} = \underset{\mathbf{R}\in SO(3), \mathbf{t}\in\mathbb{R}^3}{\arg\min} \sum_{j,t\in\mathcal{B}} \min\left(\left\|\overline{v}_t^{ref} - \boldsymbol{R}\overline{v}_j^{que} - \boldsymbol{t}\right\|_2, c_{jt}\right), \tag{2}$$

where c_{jt} is the truncation parameter [22].

3) Rotation Constraint Using RSN Map: Equation (2) generally provides stable 6-DoF pose estimation, but its performance could be degraded when point-to-point associations fail to provide effective rotation constraints. For example, if vertices in $\mathcal{B}$ are distributed along a straight line, potentially due to only one side of a road being visible, the accuracy of rotational estimation may suffer. To address this issue, we propose a prior rotation constraint from the RSN map when solving (2). This constraint is based on two assumptions: 1) road surface normals in urban environments remain consistent in a local area over time; and 2) yaw rotation is the dominant component in the relative rotation motion of ground robots and vehicles [25]. Therefore, the z-axis of the vehicle is approximately parallel to the local road surface normal, providing an additional constraint for rotation to improve registration performance. Specifically, we first calculate the average of geometric centroids of all $v_t^{ref} \in \mathcal{B}$, noted as anchor p_a. The $\underrightarrow{n}$ in the RSN map, with its starting point closest to p_a, is used to constrain the z-axis of the vehicle. This normal is denoted as $\underrightarrow{\hat{n}} \in \mathbb{R}^3$, with a standard deviation of $\sigma_{\underrightarrow{\hat{n}}}$. Let $\boldsymbol{R} = [\underrightarrow{r_1}\ \underrightarrow{r_2}\ \underrightarrow{r_3}]$, where $\underrightarrow{r_1}$, $\underrightarrow{r_2}$, and $\underrightarrow{r_3} \in \mathbb{R}^3$ is a vector that represents the direction of x, y, and z axes of $\boldsymbol{R}$, respectively. We then have the following equation:

$$\underrightarrow{\hat{n}} \cdot \underrightarrow{r_3} = 1 + \epsilon_n. \tag{3}$$

where $\cdot$ denotes the dot product, and ϵ_n represents the measurement noise. To account for road surface normal variance due to the position difference between the anchor p_a and the actual position $\boldsymbol{t}$, we add an additional perturbation δ (rad) of 5°, i.e., $\epsilon_n = \sigma_{\underrightarrow{n}} + \delta$. Equation (3) is then used as a prior constraint for $\hat{\boldsymbol{R}}$.

4) Solving Using Graduated Non-Convexity: The non-convex optimization problem (2) with the prior rotation constraint (3) are solved using Graduated Non-Convexity (GNC) [24] implemented in GTSAM [26]. Specifically, we use `PriorFactor` from GTSAM to implement the prior rotation constraints in (3).

IV. Experimental Results and Analyses

A. Dataset and Experimental Setup

1) Baselines: We compare the proposed TripletLoc with several state-of-the-art PR-based and registration-based methods. PR-based baselines include Scan Context (`SC`) [3], `STD` [16], `GOSMatch` [11], and `Triplet-Graph` [7]. We use `GLOSOM` [8] and `Outram` [9] as the registration-based baselines, which both follow one-shot based scan-to-map mechanism. All the mentioned methods are implemented in C++ and evaluated on a PC with an Intel i7-12700F CPU and 64 GB RAM.

2) Dataset and Setup: We evaluate TripletLoc and all the other baselines using the HeLiPR Dataset [27], which covers long-term data collections in diverse scenarios, from urban cityscapes to high-dynamic freeways. For each scenario in HeLiPR, different sequences are collected on different dates along a similar route, allowing the long-term performance of different localization methods to be assessed. The sequence collected earliest is selected as the reference, while the others serve as query sequences. LiDAR scans from the Spinning Ouster LiDAR are used for evaluation. Following the evaluation setup for PR methods in [27], we sample query scans at 10 m intervals and reference scans at 5 m intervals. It should be noted that `STD` originally uses accumulated sub-map of 10 consecutive scans [16]. So, here we provide two versions of `STD`, i.e., a single frame version (`STD-1`) and a sub-map version (`STD-10`) which accumulates 10 consecutive frames for each sampled scan. As for `Triplet-Graph`, we first use the global descriptor without the selection operation to obtain candidates, and then use the global descriptor with the selection operation to obtain the final loop closure result from candidates [7]. For registration-based methods, since `GLOSOM` and `Outram` are not open-sourced for their instance-level map generation, we use the same instance clustering method and the generated map for all the registration-based methods. In addition, we follow the setup for the KITTI dataset in [8], only parking and traffic signs are used in `GLOSOM`.

TABLE I
THE DETAILS ABOUT THE EVALUATED SEQUENCES FROM THE HELIPR DATASET

Reference Sequence	Query Sequence	Time Span
DCC04 (5.5km)	DCC05 (5.3km)	10 hours
	DCC06 (4.6km)	138 days
KAIST04 (6.3km)	KAIST05 (6.9km)	11 hours
	KAIST06 (6.7km)	138 days
Roundabout01 (9.0km)	Roundabout02 (7.4km)	16 days
	Roundabout03 (9.3km)	28 days
Town01 (7.8km)	Town02 (8.2km)	13 days
	Town03 (8.9km)	27 days
Riverside04 (6.5km)	Riverside05 (6.4km)	11 hours
	Riverside06 (7.2km)	138 days
Bridge01 (23.1km)	Bridge02 (14.6km)	14 days
	Bridge03 (19.4km)	28 days

Time span refers to the collection time span between the query sequence and the reference sequence.

We use default parameters for `Outram` [9] for scenarios KAIST and Town. For the other scenarios, we fine-tune the parameters to limit association numbers to avoid memory exhaustion when searching MCQ. The details about the evaluated sequences are displayed in Table I.

3) Evaluation Metrics: We use Relative Translation Error (RTE) and Relative Rotation Error (RRE) [7], [9] to evaluate the translation and rotation accuracy for global localization, respectively. RTE is calculated as $RTE = \|\hat{\boldsymbol{t}} - \boldsymbol{t}\|_2$. RRE is calculated as $RRE = \cos^{-1}(\frac{\mathrm{Tr}(\boldsymbol{R}^{-1}\hat{\boldsymbol{R}})-1}{2})$. Rotation estimations between query scans and retrieved scans are all available in `SC` (1-DoF, i.e., yaw), `STD` (3-DoF), `GOSMatch` (3-DoF), and `Triplet-Graph` (3-DoF). So, successful place recognition is defined by identifying a candidate with $RTE < 7.5$ m and $RRE < 10^\circ$, termed a true positive [27]. Similarly, we consider a localization result with $RTE < 7.5$ m and $RRE < 10^\circ$ as a successful case for all registration-based methods. Success rate $P = n_s/n$ is used to evaluate the overall performance across a query sequence. n_s is the number of successful place recognition or localization cases. n is the number of query scans. We also report the average runtime t_{ave} for a single query frame, excluding the time to load point clouds from binary files. Only the time for descriptor extraction and localization (i.e., retrieval for PR methods or vertex matching and pose estimation for registration methods) is considered.

B. Global Localization Performance

1) Success Rate: As shown in Table II, TripletLoc achieves a higher overall success rate. Especially in the DCC scenario, TripletLoc can successfully locate the vehicle for more than 93% of the query scans. As expected, significant degradation can be observed in the sequences Bridge02 and Bridge03. Basically, there are about $1/3$ of LiDAR scans are collected on a bridge, making these scans highly similar and repetitive in appearances and geometric structures. Meanwhile, there are multiple lane-level changes in translation between scans from sequence Bridge01 and sequence Bridge02/Bridge03. We guess these factors are the main reasons for the performance degradation. Compared to all the other methods, TripletLoc can still achieve a higher P at about 30%. In the sequences Twon02 and Town03, both `SC` and `Triplet-Graph` outperform TripletLoc. We suspect this is due to the lack of enough clustered instances in the query scans. The average number of clustered instances per scan in Town02/03 (32/34) and Bridge02/03 (27/29) is much lower than that in the other scenarios (e.g., 93 in DCC06 and 69 in Riverside06). When few clustered instances are present, TripletLoc's performance degrades. While instance numbers also affect `Triplet-Graph` [7], the impact is smaller due to its scan-to-scan mechanism. Notably, the Roundabout01, 02, and 03 sequences feature many moving vehicles and pedestrians, making semantic segmentation and instance clustering more challenging. Despite this, TripletLoc achieves the best results, with a success rate over 70%, demonstrating good robustness in highly dynamic scenarios. `Outram` achieves the highest success rate in the KAIST06 and Riverside06 sequences, while TripletLoc also performs reasonably well. Examples of the success rate of TripletLoc under different RTE and RRE thresholds are shown in Fig. 6. The results indicate that TripletLoc maintains a reasonable success rate even under more strict threshold settings.

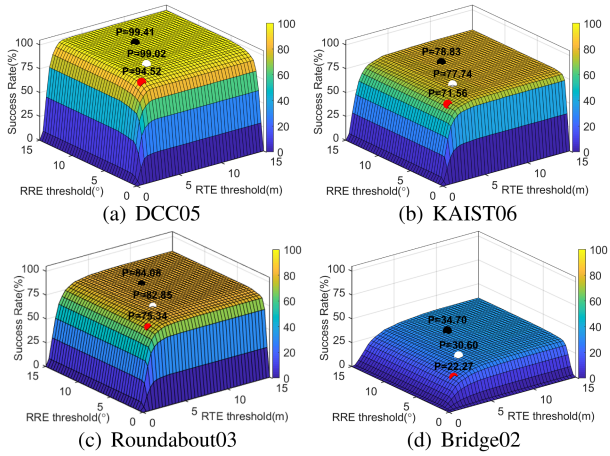

Fig. 6. Examples of success rate under different thresholds of RTE and RRE. Black, white, and red dots represent success rates for thresholds of (7.5 m, 10°), (5m, 5°), and (2.5m, 2.5°) for RTE and RRE, respectively.

2) Localization Accuracy: Only successfully localized scans (i.e., $RTE < 7.5$ m and $RRE < 10^\circ$) are used to evaluate the localization accuracy. As shown in Table II, registration-based methods outperform PR-based methods. `GLOSOM` shows the best RTE performance, with values under 1m in all query sequences except Riverside05 and 06. In most sequences, TripletLoc's RTE is very close to `GLOSOM`. Regarding RRE, TripletLoc has the best overall performance, with values under 2° in most of the sequences. Indeed, TripletLoc achieves a higher overall success rate. So, more scans are considered to calculate RTE and RRE, which might make values of RTE and RRE larger by encountering some difficult cases (i.e., successfully located scans with high RTE and RRE). In general, TripletLoc shows competitive performance in localization accuracy, with $RTE < 1$m and $RRE < 2^\circ$ in most cases.

3) Runtime Cost: Compared to registration-based methods, PR-based methods generally achieve global localization more efficiently using a retrieval strategy, often accelerated by K-Dimensional trees and hash functions. As shown in Table II, PR-based methods generally have better efficiency. Especially, only around 2 ms is needed to extract global descriptor and conduct retrieval for `SC`. In contrast, registration-based methods involve instance-to-instance matching and optimization-based pose estimation, which could be slowed down as the number of correspondences and outliers increases. `GLOSOM` uses an all-to-all

TABLE II
THE GLOBAL LOCALIZATION PERFORMANCE

Metrics	Method	Query Sequence											
		DCC05	DCC06	KAIST05	KAIST06	Roundabout02	Roundabout03	Town02	Town03	Riverside05	Riverside06	Bridge02	Bridge03
P (%)	⋆ SC	90.02	59.33	**92.05**	68.78	59.31	68.39	**73.27**	74.94	2.97	3.20	4.31	18.27
	⋆ GOSMatch	39.33	5.56	17.99	6.49	10.28	17.38	10.72	6.38	10.07	1.75	1.01	7.69
	⋆ STD-1	28.96	21.56	25.49	26.89	6.53	10.76	4.04	7.54	5.28	3.64	1.15	5.45
	⋆ STD-10	54.21	49.78	43.03	50.23	8.47	25.11	9.46	12.41	20.46	18.63	2.30	9.05
	⋆ Triplet-Graph	96.87	66.44	84.86	57.03	61.81	73.99	72.64	**75.41**	42.41	23.29	16.88	32.01
	† GLOSOM	56.36	50.22	9.60	9.27	28.06	30.94	24.72	25.41	19.31	14.56	6.32	9.21
	† Outram	55.77	61.33	77.81	**91.34**	45.83	51.23	38.08	43.74	71.29	**76.86**	19.90	24.21
	† TripletLoc	**99.41**	**93.33**	76.76	78.83	**74.58**	**84.08**	50.69	53.36	**75.58**	73.36	**34.70**	**38.00**
	† TripletLoc(no RSN)	**99.41**	92.89	74.81	78.52	74.17	83.86	48.55	51.97	74.42	71.76	30.82	35.44
	† TripletLoc($Des^{\alpha}_{v_j}$)	98.63	91.11	70.31	77.13	68.75	80.72	44.39	46.52	69.97	69.58	24.78	29.28
RTE (m)	⋆ SC	-	-	-	-	-	-	-	-	-	-	-	-
	⋆ GOSMatch	2.56±1.54	2.61±1.60	2.70±1.68	2.85±2.01	3.02±1.81	2.57±1.76	2.72±1.73	2.61±1.76	2.21±1.49	1.62±1.07	3.35±2.00	2.70±2.01
	⋆ STD-1	1.20±0.90	0.86±0.58	0.73±0.52	0.83±0.87	0.94±0.83	1.02±1.01	0.88±0.73	0.87±0.69	1.53±0.98	1.63±0.98	1.69±1.35	1.64±1.51
	⋆ STD-10	0.99±0.82	0.84±0.62	0.82±0.58	0.81±0.88	1.17±1.19	0.99±0.99	0.94±0.75	1.33±1.06	1.51±0.82	1.32±0.96	1.64±1.38	1.75±1.51
	⋆ Triplet-Graph	**0.78±0.80**	0.79±0.63	**0.46±0.33**	0.49±0.40	0.72±0.55	**0.52±0.50**	0.56±0.60	**0.63±0.52**	1.21±0.51	1.23±0.58	1.15±1.00	1.03±0.82
	† GLOSOM	0.88±0.80	**0.57±0.29**	0.56±0.33	0.48±0.35	0.70±0.45	0.54±0.40	**0.56±0.41**	0.66±0.38	**1.03±0.45**	**1.00±0.40**	**0.69±0.38**	**0.84±0.43**
	† Outram	0.79±0.68	0.87±0.54	0.56±0.48	0.46±0.29	0.92±0.71	0.71±0.55	0.86±0.65	0.95±0.69	1.15±0.37	1.14±0.41	1.46±0.85	1.58±0.90
	† TripletLoc	0.82±0.72	0.70±0.36	0.58±0.51	**0.46±0.28**	**0.70±0.40**	0.57±0.43	0.66±0.60	0.73±0.44	1.13±0.43	1.16±0.51	1.21±0.81	1.24±0.74
	† TripletLoc(no RSN)	0.82±0.72	0.71±0.38	0.58±0.46	0.48±0.33	0.71±0.43	0.56±0.43	0.65±0.52	0.75±0.50	1.14±0.40	1.15±0.50	1.20±0.72	1.26±0.76
	† TripletLoc($Des^{\alpha}_{v_j}$)	0.82±0.72	0.70±0.35	0.58±0.46	0.51±0.38	0.79±0.61	0.59±0.45	0.62±0.49	0.77±0.50	1.21±0.52	1.17±0.52	1.16±0.65	1.25±0.71
RRE (°)	⋆ SC	1.17±1.15	1.69±1.60	1.45±1.21	1.59±1.26	2.61±1.32	1.81±1.55	1.95±1.39	1.91±1.40	5.36±1.91	5.61±1.94	**1.14±0.76**	**1.43±1.64**
	⋆ GOSMatch	3.59±2.56	3.57±2.37	4.01±2.37	4.86±2.56	4.98±2.46	4.22±2.61	4.47±2.71	4.50±2.50	4.76±2.64	3.94±2.38	4.95±2.83	3.90±2.44
	⋆ STD-1	2.57±1.92	2.79±2.19	2.03±1.58	2.61±2.06	3.30±1.90	2.64±2.03	3.25±2.45	2.79±2.02	3.20±2.23	4.04±2.14	2.75±1.48	3.13±2.23
	⋆ STD-10	1.92±1.67	2.02±1.75	1.96±1.72	1.74±1.42	3.69±2.40	2.36±1.82	3.12±2.06	3.73±2.56	3.18±2.02	3.05±2.00	2.49±1.72	2.47±1.75
	⋆ Triplet-Graph	**0.69±0.88**	1.55±1.63	**1.02±1.24**	1.82±1.84	2.46±1.64	1.35±1.45	**1.62±1.80**	**1.62±1.75**	1.94±1.54	2.75±2.11	2.74±2.23	1.82±1.79
	† GLOSOM	1.49±1.30	1.44±1.20	1.52±1.49	1.34±1.64	2.39±1.56	1.36±1.42	1.73±1.65	1.70±1.60	1.80±1.56	2.17±1.77	1.56±1.69	1.44±1.23
	† Outram	2.03±1.77	2.15±1.84	1.56±1.49	1.66±1.47	3.22±2.09	2.30±1.81	2.78±2.07	2.85±2.23	1.95±1.63	2.03±1.66	3.70±2.26	3.55±2.28
	† TripletLoc	0.96±0.62	**1.10±0.91**	1.48±1.29	**1.31±1.07**	**2.36±1.52**	**1.31±1.06**	1.74±1.52	1.74±1.43	**1.70±1.34**	**1.66±1.42**	2.46±1.92	2.23±1.82
	† TripletLoc(no RSN)	0.97±0.66	1.13±1.04	1.54±1.49	1.40±1.17	2.46±1.64	1.34±1.11	1.91±1.81	1.96±1.90	1.74±1.44	1.69±1.48	2.76±2.19	2.51±2.18
	† TripletLoc($Des^{\alpha}_{v_j}$)	1.07±0.72	1.15±0.71	1.56±1.33	1.44±1.29	2.55±1.64	1.51±1.26	1.80±1.43	1.81±1.51	1.91±1.49	1.77±1.39	2.50±1.69	2.30±1.84
t_{ave} (ms)	⋆ SC	**2.35**	**2.27**	**2.19**	**2.26**	**2.25**	**2.28**	**2.34**	**2.36**	**2.23**	**2.10**	**2.17**	**2.26**
	⋆ GOSMatch	29.10	32.71	35.11	31.80	27.43	27.85	21.69	16.49	21.61	25.35	26.39	12.75
	⋆ STD-1	79.33	76.17	64.03	69.94	70.56	77.70	51.81	52.92	51.04	52.01	64.66	70.50
	⋆ STD-10	320.26	327.59	287.23	337.01	366.88	323.03	192.87	196.71	256.85	270.30	316.48	364.91
	⋆ Triplet-Graph	377.34	442.80	268.10	284.95	354.92	458.31	342.98	348.97	326.70	433.14	302.60	360.44
	† GLOSOM	1046.17	1430.92	73.50	84.86	1522.14	1801.99	616.37	628.17	119.81	146.14	651.07	789.88
	† Outram	234.44	336.115	527.64	744.47	527.64	596.05	757.45	827.57	365.01	680.76	619.47	521.55
	† TripletLoc	82.63	121.90	19.00	33.12	69.41	82.37	30.13	30.83	47.05	72.13	50.46	54.12
	† TripletLoc(no RSN)	81.02	118.38	17.79	31.43	67.12	79.22	28.41	29.18	46.65	72.55	46.37	51.73
	† TripletLoc($Des^{\alpha}_{v_j}$)	63.77	93.46	15.47	27.39	44.15	53.21	23.33	24.13	40.33	62.99	27.20	29.66

The best results are highlighted in bold font. The second best results are highlighted with an underline. "-" refers to not available. "⋆" refers to place recognition-based method. "†" refers to one-shot registration based method."No RSN" refers to omitting rotation constraint from RSN map. "TripletLoc($Des^{\alpha}_{v_j}$)" refers to only using $Des^{\alpha}_{v_j}$ for vertex matching. Tested on intel i7-12700f CPU.

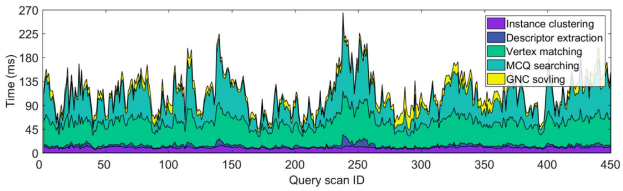

Fig. 7. Time cost breakdown for sequence DCC06. Tested on Intel i7-12700F CPU and 64 GB RAM.

matching strategy, leading to a large number of correspondences. Outram employs a triangle-based descriptor for substructure to build correspondences, while the condition-meeting mechanism can still result in many correspondences sometimes. As a result, graph-theoretic outlier pruning may reduce computational efficiency when the number of correspondences is too large in large-scale environments. The average runtime cost for GLOSOM and Outram is larger than that for the other methods. GLOSOM operates in real-time only when the number of instances in the reference map is relatively small (e.g., 536 for KAIST04 and 551 for Riverside04, versus 2011 for Bridge01 and 1719 for Roundabout01, considering parking and traffic signs). Differently, TripletLoc uses simple yet efficient top-k matches to limit correspondences, allowing it to run much faster in real-time. Fig. 7 shows the cost time breakdown per frame for TripletLoc on the DCC06 query sequence (the slowest query sequence). Vertex matching and MCQ searching (including consistency graph construction) dominate, with average times of 45.50 ms and 45.44 ms, respectively. The descriptor extraction is fast, averaging 3.75 ms per scan, while instance clustering and GNC solving take 9.85 ms and 11.03 ms on average, respectively.

C. Ablation Study

1) Effectiveness of $Des^{d}_{v_j}$: To demonstrate the effectiveness of integrating $Des^{d}_{v_j}$, we perform global localization using only $Des^{\alpha}_{v_j}$ for vertex matching. The results, shown in Table II as TripletLoc($Des^{\alpha}_{v_j}$), indicate that the complete version of TripletLoc achieves higher success rates across all query sequences compared to using only $Des^{\alpha}_{v_j}$. Localization accuracy shows a slight decline in RTE and RRE when using only $Des^{\alpha}_{v_j}$. As expected, when using Des_{v_j}, it takes more time for vertex descriptor extraction and matching, leading to a larger t_{ave}. However, such an increase is acceptable, and the system can still achieve a good real-time performance. In conclusion, explicitly embedding both relative angle and edge length from triplets can enhance the descriptive capability of vertex descriptors, improving the overall performance for global localization.

2) Effectiveness of RSN-Based Rotation Constraint: To demonstrate the effectiveness of the RSN-based rotation constraint, we present results for TripletLoc without this constraint, labeled as TripletLoc (no RSN) in Table II. A decline in success rates is observed across all sequences except DCC05 when the RSN-based rotation constraint is omitted from pose estimation. This decline is most pronounced in the Bridge scenario, where instances are primarily linearly located along both sides of the road, and point-to-point associations may not provide sufficient rotation constraint when limited instances are visible. As for localization accuracy, RTE remains largely unchanged, indicating limited impact of rotation constraint on translation. A slight decline in RRE can be also observed without rotation constraint. These results confirm that the RSN-based rotation constraint can enhance global localization.

TABLE III
THE MEMORY CONSUMPTION FOR PRIOR REFERENCE MAP

SC	GOSMatch	STD-1	STD-10	Triplet-Graph	GLOSOM	Outram	TripletLoc
16.27MB	68.85MB	393.37MB	597.24MB	631.25.24MB	**27.50KB**	561.60KB	245.63KB

D. Memory Consumption for Reference Map

To show the memory efficiency of TripletLoc, we measure the total memory consumption for storing prior reference maps on sequence Roudnabout01. For PR-based methods, vectorized descriptors for each reference frame are stored. For GLOSOM, only instance information is needed (i.e., instance label and geometric centroids). In Outram, the covariance matrix for clustered point cloud is also needed. To allow map updates, we also record the point number for each instance. The road surface normal, its starting point, and the standard deviation need to be stored for the RSN map (28.75 KB). As shown in Table III, the memory cost for GLOSOM is the lowest. TripletLoc also shows competitive efficiency in memory consumption.

V. CONCLUSIONS AND FUTURE WORK

In this study, we present TripletLoc, a fast and robust one-shot LiDAR-based global localization method. Semantic graphs are used to compactly represent both the query scans and the entire reference map. A novel semantic triplet-based histogram descriptor is proposed to embed semantic, geometric, and topological information from the semantic graphs for each vertex. Based on the proposed vertex descriptor, instance-to-instance correspondences are generated. Graph-theoretic outlier pruning is used to select inlier correspondences for robust 6-DoF pose estimation. A novel RSN map is proposed to provide a prior rotation constraint to further enhance pose estimation. Extensive experiments in diverse and large-scale urban environments demonstrate that our TripletLoc is highly competitive to state-of-the-art methods. Quantitative and qualitative results show the robustness, accuracy, as well as memory and real-time efficiency of TripletLoc. However, our RSN map is currently only applicable for ground robots in outdoor environments with flat road surface. In the future, we plan to introduce more information of instances (e.g., size and orientation) to TripletLoc and extend it to indoor environments.

REFERENCES

[1] H. Yin et al., "A survey on global LiDAR localization: Challenges, advances and open problems," *Int. J. Comput. Vis.*, vol. 132, pp. 3139–3171, 2024.

[2] W. Ma, H. Yin, L. Yao, Y. Sun, and Z. Su, "Evaluation of range sensing-based place recognition for long-term urban localization," *IEEE Trans. Intell. Veh.*, vol. 9, no. 5, pp. 4905–4916, May 2024.

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

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

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

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

[7] W. Ma, S. Huang, and Y. Sun, "Triplet-Graph: Global metric localization based on semantic triplet graph for autonomous vehicles," *IEEE Robot. Automat. Lett.*, vol. 9, no. 4, pp. 3155–3162, Apr. 2024.

[8] J. Ankenbauer, P. C. Lusk, A. Thomas, and J. P. How, "Global localization in unstructured environments using semantic object maps built from various viewpoints," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2023, pp. 1358–1365.

[9] P. Yin et al., "Outram: One-shot global localization via triangulated scene graph and global outlier pruning," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2024, pp. 13717–13723.

[10] S. Matsuzaki, K. Koide, S. Oishi, M. Yokozuka, and A. Banno, "Single-shot global localization via graph-theoretic correspondence matching," *Adv. Robot.*, vol. 38, no. 3, pp. 168–181, 2024.

[11] Y. Zhu, Y. Ma, L. Chen, C. Liu, M. Ye, and L. Li, "GOSMatch: Graph-of-semantics matching for detecting loop closures in 3D LiDAR data," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2020, pp. 5151–5157.

[12] G. Pramatarov, D. De Martini, M. Gadd, and P. Newman, "BoxGraph: Semantic place recognition and pose estimation from 3D LiDAR," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2022, pp. 7004–7011.

[13] F. Dellaert, D. Fox, W. Burgard, and S. Thrun, "Monte Carlo localization for mobile robots," in *Proc. IEEE Int. Conf. Robot. Automat.*, 1999, pp. 1322–1328.

[14] 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," *J. Field Robot.*, vol. 26, no. 11/12, pp. 892–914, 2009.

[15] X. Xu et al., "RING++: Roto-translation-invariant gram for global localization on a sparse scan map," *IEEE Trans. Robot.*, vol. 39, no. 6, pp. 4616–4635, Dec. 2023.

[16] C. Yuan, J. Lin, Z. Zou, X. Hong, and F. Zhang, "STD: Stable triangle descriptor for 3D place recognition," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2023, pp. 1897–1903.

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

[18] J. Komorowski, "MinkLoc3D: Point cloud based large-scale place recognition," in *Proc. IEEE Winter Conf. Appl. Comput. Vis.*, 2021, pp. 1790–1799.

[19] K. Żywanowski, A. Banaszczyk, M. R. Nowicki, and J. Komorowski, "MinkLoc3D-SI: 3D LiDAR place recognition with sparse convolutions, spherical coordinates, and intensity," *IEEE Robot. Automat. Lett*, vol. 7, no. 2, pp. 1079–1086, Apr. 2022.

[20] H. Tang et al., "Searching efficient 3D architectures with sparse point-voxel convolution," in *Proc. Eur. Conf. Comput. Vis.*, 2020, pp. 685–702.

[21] R. B. Rusu and S. Cousins, "3D is here: Point cloud library (PCL)," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2011, pp. 1–4.

[22] H. Yang, J. Shi, and L. Carlone, "TEASER: Fast and certifiable point cloud registration," *IEEE Trans. Robot.*, vol. 37, no. 2, pp. 314–333, Apr. 2021.

[23] R. A. Rossi, D. F. Gleich, and A. H. Gebremedhin, "Parallel maximum clique algorithms with applications to network analysis," *SIAM J. Sci. Comput.*, vol. 37, no. 5, pp. C589–C616, 2015.

[24] H. Yang, P. Antonante, V. Tzoumas, and L. Carlone, "Graduated non-convexity for robust spatial perception: From non-minimal solvers to global outlier rejection," *IEEE Robot. Automat. Lett*, vol. 5, no. 2, pp. 1127–1134, Apr. 2020.

[25] H. Lim, B. Kim, D. Kim, E. Mason Lee, and H. Myung, "Quatro++: Robust global registration exploiting ground segmentation for loop closing in LiDAR slam," *Int. J. Robot. Res.*, vol. 43, no. 5, pp. 685–715, 2024.

[26] F. Dellaert and G. Contributors, "borglab/gtsam," May 2022. [Online]. Available: https://github.com/borglab/gtsam

[27] M. Jung, W. Yang, D. Lee, H. Gil, G. Kim, and A. Kim, "HeLiPR: Heterogeneous LiDAR dataset for inter-LiDAR place recognition under spatiotemporal variations," *Int. J. Robot. Res.*, vol. 43, no. 12, pp. 1867–1883, 2024.