GNSS-FREE GLOBAL LOCALIZATION IN URBAN ENVIRONMENTS

WEIXIN MA

PhD

The Hong Kong Polytechnic University

2025

The Hong Kong Polytechnic University
Department of Mechanical Engineering

GNSS-free Global Localization in Urban Environments

Weixin Ma

A thesis submitted in partial fulfillment of the requirements for
the degree of Doctor of Philosophy
Nov 2024

CERTIFICATE OF ORIGINALITY

I hereby declare that this thesis is my own work and that, to the best of my knowledge and belief, it reproduces no material previously published or written, nor material that has been accepted for the award of any other degree or diploma, except where due acknowledgment has been made in the text.

Signature: _____

Name of Student: Weixin Ma

Abstract

Global localization provides robots with their current positions relative to the environment, enabling them to perform more complex downstream tasks such as path planning and navigation. One of the most widely adopted solutions, the Global Navigation Satellite System (GNSS), has demonstrated significant advantages over the past decades. Its robustness and accuracy may be compromised by weak signals, multipath effects, or signal obstruction, especially in densely built urban areas with tall buildings. We therefore identify a need to advance techniques on global localization in GNSS-denied environments. The goal of this thesis is to develop real-time solutions with high memory efficiency for GNSS-free global localization in urban environments, with a focus on autonomous vehicles.

Our first contribution focuses on global localization in high-rise environments using a publicly available map, i.e., OpenStreetMap. We observe that the building roof outline captured by a sky-looking fish-eye camera shares similarities with the building outline featured in the OpenStreetMap. Based on this observation, we propose a descriptor that incorporates both topological (i.e., junction types) and geometric (i.e., building outline) information to bridge fish-eye camera images with OpenStreetMap data. To handle the challenge of similar or repeated building outlines across different images, we formulate our method as a Bayesian Filtering problem using Monte Carlo localization, which leverages multiple consecutive fish-eye images for robust localization. The sky-looking fish-eye camera also naturally avoids disturbances from

dynamic objects such as vehicles and pedestrians, making it particularly suitable for urban environments.

While our first solution enables real-time localization in high-rise environments without the need for manually collecting and maintaining reference maps, its long-term reliability can be impacted by the quality of captured images, which are sensitive to changes in illumination, weather, and seasons. In contrast, range sensors such as LiDAR and radar are more resilient to these environmental factors. To investigate the influence of these factors on range sensing based global localization methods, we conduct a comprehensive evaluation of current techniques in long-term scenarios with significant seasonal variations and adverse weather conditions. In addition, we design a novel metric to evaluate the influence of matching thresholds on place recognition performance for long term localization.

Based on our previous evaluation study, LiDAR-based place recognition shows good robustness under long-term conditions. However, it typically only identifies whether the current place has been visited before. Determining the vehicle's pose relative to the environment requires additional point cloud or feature point based registration, which might demand significant memory to store these data. Our third contribution addresses this challenge by integrating pose estimation with place recognition to improve LiDAR-based global localization performance. Rather than relying on abstract environmental representations like 3D points, we use lightweight semantic features (e.g., traffic signs, trees, poles, and so on) to represent both on-vehicle LiDAR scans and reference scans in the databases, significantly reducing memory requirements. We propose a novel semantic histogram descriptor to represent each semantic instance, which is used for instance association in pose estimation and aggregated into a global descriptor for place recognition.

While our previous contribution show competitive localization accuracy and memory efficiency, the scan-to-scan framework still redundantly stores some of semantic instances multiple times across keyframes. Meanwhile, its RANSAC-based pose

estimation might fail when outliers dominate. Our fourth contribution shifts to a scan-to-map localization manner, where each semantic instance is stored only once in the reference map, further improving the memory efficiency. We improve the semantic histogram descriptor in our previous work to achieve more robust and effective instance-to-instance correspondences. In addition, we propose a novel Road Surface Normal (RSN) map to provide a prior rotational constraint, enhancing pose estimation. We then apply graph-theoretic outlier pruning to extract inlier correspondences for robust 6-DoF pose estimation.

Finally, we develop a multi-robot localization system, building upon our previous scan-to-map localization approach. Unlike single-robot localization, multi-robot systems typically lack a prior map, and the initial relative poses between local reference frames of robots are unknown. To this end, we formulate the multi-robot localization as an optimization problem and incorporate one-shot registration-based localization into the optimization process. Specifically, lightweight semantic instances from different robots are transmitted to a central server, which constructs instance maps for inter-robot localization. We propose a dual-metric validation strategy to confirm the validity of pairwise localization results from our previous scan-to-map localization solution to reduce the risk of involving incorrect localization results. A pose averaging based optimization method is used to obtain reliable alignment estimations between local reference frames of different robots. A shortest transformation chain searching method is used to align all robots into a shared reference frame. Our preliminary results demonstrate the feasibility and effectiveness of the proposed system, as well as its promising potential in communication bandwidth-limited conditions.

Publications Arising from the Thesis

1. Weixin Ma, Shoudong Huang, and Yuxiang Sun, "SkyLoc: Cross-modal Global Localization with a Sky-looking Fish-eye Camera and OpenStreetMap in GNSS-degraded Dynamic Environments", accepted by *IEEE Transactions on Intelligent Transportation Systems (TITS)*.

2. Weixin Ma, Huan Yin, Patricia J. Y. Wong, Danwei Wang, Yuxiang Sun, and Zhongqing Su, "TripletLoc: One-Shot Global Localization using Semantic Triplet in Urban Environments", in *IEEE Robotics and Automation Letters (RA-L)*, vol. 10, no. 2, pp. 1569-1576, 2025..

3. Weixin Ma, Shoudong Huang, and Yuxiang Sun, "Triplet-Graph: Global Metric Localization based on Semantic Triplet Graph for Autonomous Vehicles", in *IEEE Robotics and Automation Letters (RA-L)*, vol. 9, no. 4, pp. 3155-3162, 2024.

4. Weixin Ma, Huan Yin, Lei Yao, Yuxiang Sun, and Zhongqing Su, "Evaluation of Range Sensing-based Place Recognition for Long-term Urban Localization", in *IEEE Transactions on Intelligent Vehicles (TIV)*, vol. 9, no. 5, pp. 4905-4916, 2024.

Acknowledgments

First and foremost, I wish to express my deepest gratitude to my chief supervisor Prof. Dr. Zhongqing Su, for giving me the opportunity to be a member of his research group. I sincerely appreciate his unwavering support, trust, and the academic freedom he has provided throughout my research journey. He has always been generous and friendly to share his life experience and invaluable suggestions on my study and life. He has constantly encouraged me to overcome challenges and seize every possible opportunity for self-improvement. Studying under his mentorship has been an exceptional honor and privilege, one that I will cherish for a lifetime.

I would also like to extend my heartfelt thanks to Prof. Yuxiang Sun (CityU), Prof. David NAVARRO-ALARCON, and Prof. Huan Yin (HKUST) for accepting to co-supervise my research projects and for contributing to the techniques which represent the foundations of this thesis. Working alongside them has been a truly enriching experience, one that has deepened my understanding and passion for robotics. Their mentorship has not only guided my research but also profoundly shaped my academic journey, and I am sincerely grateful for the opportunity to collaborate with such distinguished scholars. Meanwhile, I am profoundly thankful to Prof. Danwei Wang and Prof. Patricia J. Y. Wong at the Nanyang Technological University (NTU), Singapore, for their support and guidance during my four-month visit, where I conducted part of the experiments for this PhD thesis. I am also sincerely grateful to my Master's supervisors, Prof. Youping Chen, Prof. Jinming Xie, and Prof. Gang Zhang at

Huazhong University of Science and Technology, China, for their encouragement and support in recommending me to pursue a higher academic degree.

Throughout this PhD journey, it was an immense pleasure to collaborate with many friendly and nice people. I would like to express my special thanks to Dr. Xuzhan Chen (HUAWEI) and Dr. Renzhi Yuan (BUPT), who gave me invaluable suggestions during this journey. Along with Wanglinhan Zhang, Lei Yao, Jianyuan Ruan, Xinze Guo, Lei Fan, Shiyu Meng, Guangzhi Tian, Jun Wang, KeYu Chen, Zhuoyuan Liu, Yibo Wang, Xuyang Tang, Yi He, Jiangang Xu, Yafeng Chen, He Gao, Guojie Luo, Qi Yuan, Ming Ma, Qingqing Wang, Yanfeng Lang, Huaiyuan Xu, Haotian Li, Shuang Gao, Yuchao Feng, Zhen Feng, Jie Huang, Pengfei Li, Xin Gong, Jianbin Li, Jiaxin Tian, and Zaiwei Liu, they have enriched this journey with friendship and joy. I am also indebted to Dr. Guohao Peng, Dr. Jun Zhang, Mr. Pengyu Yin, Mr. Heshan Li, Mr. Kaimin Mao, Mr. Yichen Zhou, Mr. Yanpu Yun for their support and help during my visit at NTU, which made me feel truly at home. The discussion and exchanges of knowledge with them greatly benefited my academic research and career development.

I am also deeply thankful to Dr. Kwai-Wa Tse and Ms. Lily Tam for their assistance and guidance in addressing many of the challenges I encountered during my time at PolyU.

Finally, my heartfelt gratitude to my family, especially my parents, for their constant love, unconditional support, and encouragement in chasing my dreams over all these years.

Table of Contents

List of Figures

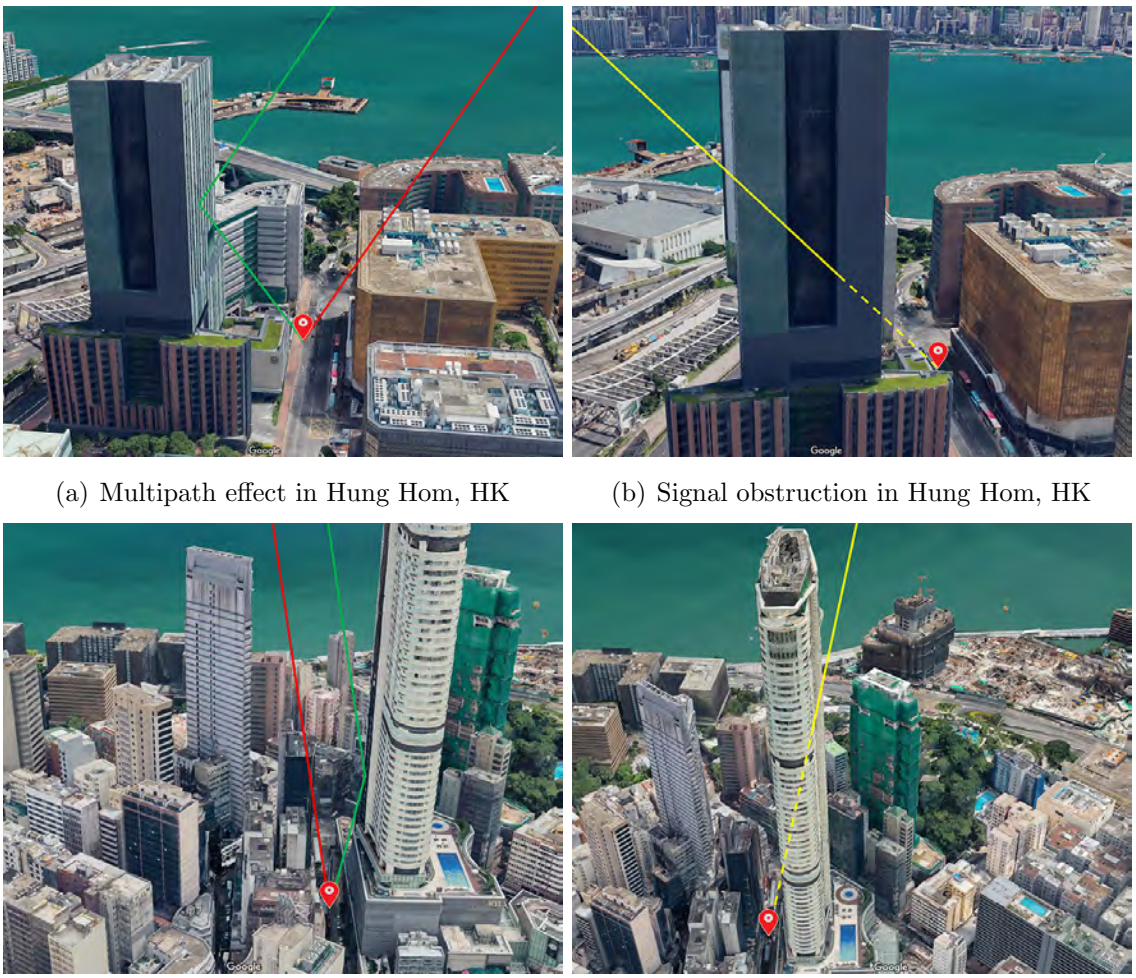
List of Tables

Chapter 1

Introduction

1.1 Research Background

Autonomous driving has gained significant attention in recent years, driven by its potential to revolutionize the transportation industry. This surge in popularity is driven by the promising benefits of safer transportation, reduced traffic congestion, increased efficiency, less stress for drivers in traffic, and enhanced mobility for people with disabilities [1]. With the rapid advancements in sensor technology, in-vehicle computing platform, and artificial intelligence, autonomous vehicles (AVs) are no longer a futuristic concept but are being actively tested and deployed in real-world scenarios [2]. The ultimate goal of autonomous driving technology is to achieve full automation—where AVs can perform all driving tasks under any conditions without human intervention. Although this level of autonomy has yet to be realized, it remains an area of active research and development. To approach high-level or full autonomy, an AV must possess several core capabilities: 1) global localization-determining the vehicle's position relative to its involved environment; 2) perception-recognizing and interpreting the surrounding environment using on-board sensors; 3) path planning-calculating the optimal route to a destination based on road condition, traffic, and



(a) Multipath effect in Hung Hom, HK

(b) Signal obstruction in Hung Hom, HK

(c) Multipath effect in Tsim Sha Tsui, HK

(d) Signal obstruction in Tsim Sha Tsui, HK

Figure 1.1: Examples of signal reception issues due to multipath effects and obstruction. Green lines represent satellite signals reflected off building surfaces, while yellow dotted lines indicate satellite signals obstructed by tall buildings.

obstacles; 4) motion control-executing smooth and safe vehicle movements, including steering, acceleration, and braking; and 5) decision making-making informed decisions based on dynamic traffic conditions and adherence traffic rules [3].

Among the various capabilities required for autonomous driving, global localization stands out as a particularly important component. Precise localization provides the vehicle with a continuous and reliable understanding of its position relative to its surroundings. This understanding is essential for performing a range of tasks, from

basic navigation to complex decision-making processes, such as avoiding obstacles and path re-planing under city construction. However, achieving robust and precise localization is particularly challenging in urban environments, where the Global Navigation Satellite System (GNSS) faces significant limitations. These typically include signal obstructions caused by tall buildings, as well as the multipath effect, where satellite signals reflect off surfaces before reaching the receiver, leading to inaccurate positioning, as shown in Fig. 1.1. Consequently, in densely built-up urban environments, GNSS localization accuracy can drop to as much as 30–50 meters, making it difficult for vehicles to maintain reliable localization[4]. This degradation presents a significant challenge to autonomous driving systems, which might result in incorrect navigation decisions and potential safety risks.

In recent years, there has been growing interest in alternative approaches to global localization that rely on onboard sensors, eliminating the need for GNSS [5, 6]. One of the earliest approaches in this domain is visual global localization, which takes advantage of the rich texture and appearance information captured by camera images to estimate a vehicle's position [7, 8]. However, the reliability of visual localization is heavily influenced by image quality, which can degrade due to changes in illumination, weather conditions, or seasonal variations. These environmental sensitivities might limit the consistency of performance of vision-based localization in long-term scenarios. In contrast, range sensors such as LiDAR and radar are less affected by changes in lighting or weather and can perform reliably across a wide range of environmental conditions, showing promising potential for robust global localization [5, 9]. However, range sensors typically provide pure geometric structure information of the surroundings, making data association challenging. Typically, these onboard sensor-based global localization methods require a prior reference map or database to eliminate the need for GNSS. These maps or databases provide a pre-recorded model of the environment, allowing the vehicle to match its current sensor readings with stored data for localization. However, several key challenges arise when utilizing

such reference maps or databases. First, urban environments are dynamic and often experience frequent changes in appearance and geometric structure, such as city construction or road alterations, leading to discrepancies between real-time sensor data and the information stored in reference maps. These differences complicate the process of associating on-vehicle sensor readings with the stored map data. Second, reference maps and databases often contain vast large volumes of detailed sensor data, such as high-resolution 3D maps. This demands efficient storage and processing techniques to ensure that memory consumption remains manageable without significantly compromising performance, especially as maps expand to cover larger areas. Finally, outlier associations between sensor data and reference data are common, especially in cluttered or dynamic environments. As a result, the development of robust feature representations and effective outlier rejection methods is essential to enhance the reliability of localization.

This thesis aims to contribute to the development of GNSS-free global localization solutions that operate in real-time while maintaining high memory efficiency, with a particular focus on urban environments. Dense and complex infrastructures in urban areas presents significant challenges for traditional localization methods, necessitating innovative alternatives. In response, we explore the use of publicly available maps and lightweight semantics as promising approaches to compactly represent the environment with less memory usage. These compact and efficient representations serve as the core foundation for the solutions developed and proposed throughout this thesis.

1.2 Motivation and Objectives

"Where am I?" This is a question that we frequently ask ourselves in our daily lives. Determining our current location is essential for executing subsequent tasks such as navigation or path planning. I vividly remember the first time I traveled from accommodation to the PolyU campus. The journey took me over half an hour due to the

(a) Mong Kok, HK (b) Wan Chai, HK

Figure 1.2: Examples of building and environmental layouts in Hong Kong.

complex surroundings and degraded GNSS performance, even though Google Maps indicated that it should take less than 20 minutes. Fig. 1.2 illustrates the building density and environmental layout in Mong Kok and Wan Chai, Hong Kong. In these areas with tall, dense buildings, GNSS signal obstruction and multipath effects are common challenges. These problems are not unique to Hong Kong; similar challenges arise in cities like Tokyo and New York. The reliability and accuracy of GNSS in such environments might not be sufficient to support autonomous vehicles effectively. As a result, there is a growing need for alternative approaches that do not rely on GNSS but can still provide reliable localization in complex urban settings. Current sensor-based methods, while promising, face several challenges. These include maintaining localization robustness in long-term scenarios, reducing memory demands in large-scale environment, and handling incorrect data associations caused by frequent changes in urban settings. These challenges reveal a critical gap in existing systems' ability to ensure robust, real-time localization with high memory efficiency.

In the scope of this thesis we want to overcome these challenges and contribute to the development of GNSS-free global localization solutions for urban autonomous driving. Specifically, we wish to address the following objectives:

1. **Develop GNSS-free localization solutions that ensure real-time performance.** For autonomous driving, path planning and navigation depend

on the vehicle's current position relative to its surroundings for effective re-planning and status checking. In complex urban environments, more frequent position updates are often necessary, making real-time performance essential for localization solutions.

2. **Explore the use of publicly available maps and lightweight semantic representations to reduce memory usage in large-scale environment.** As autonomous vehicles operate over vast areas, more sensor data needs to be processed and stored for reference maps or databases. Storing raw sensor data, such as high-resolution images or 3D LiDAR scans, can be memory-intensive. In contrast, publicly available maps (such as OpenStreetMap) and lightweight semantic representations offer promising solutions to reduce memory costs.

3. **Improve the robustness of localization against changes in the environment, enabling long-term applicability.** In urban environments, changes like new construction or infrastructure modifications can quickly render reference maps outdated. Updating these reference maps or databases frequently incurs significant costs, both in terms of time and resources. As a result, discrepancies between the on-vehicle sensor data and the reference data often arise. Such discrepancies will result in outlier data associations, complicating global localization. Consistent feature representation and effective outlier filtering method are therefore necessary to address these challenges.

Chapter 2

Monte Carlo Localization using Fish-eye Camera and OpenStreetMap

Global localization can estimate geo-referenced locations (e.g., longitude and latitude), which is a fundamental capability for autonomous vehicles. Most existing solutions rely on the Global Navigation Satellite Systems (GNSS). Their accuracy could be degraded by the multi-path effects or occlusions of GNSS signals in urban environments. Some GNSS-free methods could achieve global localization by comparing the current on-line sensory data with pre-built databases/maps. However, they require tedious human efforts to drive a vehicle to collect and maintain the databases/maps. Moreover, most of these methods use front-looking cameras or LiDARs, so the captured data could be easily contaminated by dynamic objects (e.g., moving vehicles and pedestrians). To provide a solution to these problems, this chapter proposes a novel global localization method by comparing an image from a sky-looking fish-eye camera with the publicly available OpenStreetMap (OSM), and using particle filter to achieve real-time metric localization in dynamic traffic environments. To evaluate

our method, we extend a public dataset with OSM data, which are retrieved through the given geo-referenced location information. Experimental results demonstrate the effectiveness and efficiency of our method.

2.1 Introduction

Localization is a fundamental capability for autonomous vehicles [10]. It can provide location information for downstream tasks, such as decision making, path planning, and autonomous navigation [11], etc. There are mainly two types of localization: global localization and local localization. Global localization aims to localize a vehicle against a geo-referenced database or map without initial guess[12]. Local localization aims to estimate relative poses with respect to previous poses or a small-size local map. For global localization, most of existing methods rely on the Global Navigation Satellite Systems (GNSS). However, GNSS is not always reliable or even sometimes unavailable due to occlusions or multi-path effects of GNSS signals [13], especially in dense urban environments, such as urban canyons. To improve the GNSS localization accuracy, 3-D city models and digital maps with environmental knowledge (e.g., street layouts and building heights) have been used to identify and use "Non-Line-Of-Sight" signals in some early works [14, 15].

To relieve the need of GNSS, place recognition and re-localization with visual cameras or LiDARs have been extensively studied in the research community [7, 16, 17, 18]. These methods generally consist of an off-line stage and an on-line stage. During the off-line stage, a data-collection vehicle usually equipped with expensive global localization equipment is employed to build a geo-referenced image or point-cloud database/map. During the on-line stage, images or point clouds captured from vehicle-mounted sensors are compared with the geo-referenced database/map to determine the current location of the vehicle.

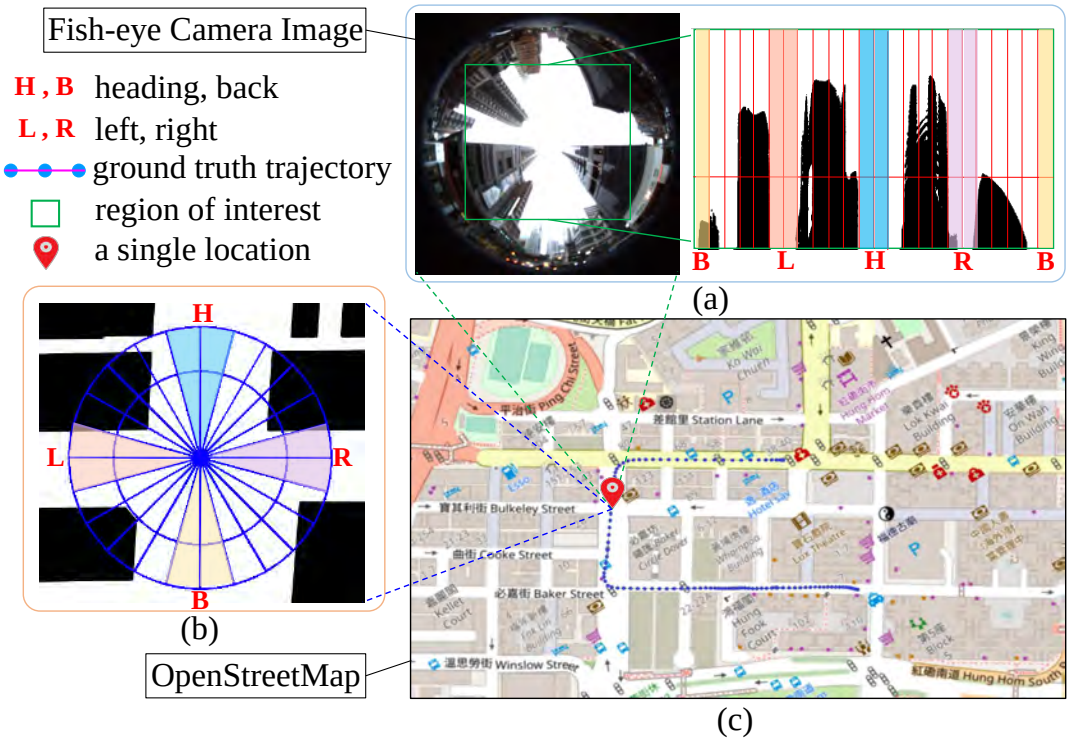


Figure 2.1: A hybrid descriptor is used to extract topological and geometric information from both input fish-eye image and OSM at the same location. The ROI of fish-eye image is transformed to the polar coordinate as shown in (a), while operation for the ROI on OSM is based on the cylinder coordinate (b). (c) is the OSM with the trajectory of the vehicle.

Despite the effectiveness of these methods, they still suffer from several issues. Firstly, building and maintaining the geo-referenced databases/maps is tedious, expensive, and time-consuming. Secondly, dynamic objects (e.g., vehicles and pedestrians) may cause discrepancies between the sensor data captured on-line and those stored in the pre-built databases/maps. Thirdly, in most existing vision-based methods, due to the limited field-of-view, the overlap between the image captured on-line and those stored in the database may not be large enough to accurately determine the locations.

To provide a solution to above issues, we proposes a novel cross-modal method, Sky-Loc, using a sky-looking fish-eye camera as on-line observation and OSM as pre-built

map, followed by a filter to achieve global localization in GNSS-degraded dynamic and complex urban traffic environments. Due to its free and open nature, OSM has been used in many applications [19, 20]. One novelty of our method is that we innovatively use a sky-looking fish-eye camera for localization, which has the advantage of avoiding disturbances from dynamic objects and can get a large field-of-view. Moreover, we adopt the combination of on-vehicle visual information and OSM to relieve the need for GNSS to achieve global localization. Since we use OSM to build our database, the tedious database building work using a data-collection vehicle can be alleviated.

To the best of our knowledge, this is the first solution using a sky-looking fish-eye camera and OSM to achieve global localization. Our method can be easily integrated into existing localization systems (e.g., LiDAR, radar, or front-looking camera based methods) to enhance their robustness by providing redundancy from different sensory sources (i.e., sky-looking Fish-eye camera). To evaluate our method, we extend an existing public dataset [21] with OSM data retrieved through the given geo-referenced location information. The contributions of this work are summarized as follows:

1. We propose a novel cross-modal global localization method with a sky-looking fish-eye camera and OSM data for dynamic and complex urban environments.

2. We design a novel topology-geometry based hybrid (TGH) descriptor to represent both the fish-eye image and OSM data to narrow the modality gap.

3. We test our method on different computing platforms including PC and embedded devices to demonstrate the real-time efficiency of our method.

This chapter is structured as follows. Section II reviews the related work. Section III describes our descriptor extraction and weighting model in detail. Experiments and results are presented in section IV. Conclusions and future work are drawn in the last section.

2.2 Related Work

This section reviews two streams of cross-modal global localization methods using public aerial data. The first stream adopts the *Ground-to-Aerial* matching framework. *Ground* means that the on-line data are captured by an on-vehicle sensor (e.g., camera or LiDAR). *Aerial* means that the off-line database is built with geo-referenced aerial or satellite images. The second stream adopts the *Ground-to-OSM* framework. The key problem in cross-modal localization lies in how to bridge the modality gap.

2.2.1 Ground-to-Aerial Matching

Vision-based Methods

Some early methods use hand-crafted image features, such as key points, lines, and planes, to match images captured by an on-vehicle camera with the images from a geo-referenced aerial-image database [22, 23, 24, 25]. These methods rely on geometric information and features to build their descriptors for matching. Differently, Noha *et al.* [26] used semantic-level textual information (i.e., shop, restaurant, or street names) to match camera images with a Google Map. Instead of using hand-crafted features, Kim *et al.* [27] used a Siamese network to learn embeddings from a ground-image sequence and satellite images, which are then used to update particle weights during filtering. Similarly, Hu *et al.* [28] also used a Siamese network to measure the similarity between ground and aerial images. The location of the query on-vehicle image is provided by retrieving aerial images stored in a database. Then, the authors extended [28] with a Markov localization framework to ensure the temporal consistency of the matching results [29]. Different from CNN-based methods, TransGeo [30] is a pure transformer-based approach, eliminating the need for aligned image pairs during training. TransGeo exhibits good flexibility and generalization but, as a retrieval-based method, requires additional registration to determine the

vehicle's relative pose to the aerial image.

Point cloud-based Methods

Range sensors, compared to visual sensors, are more robust to illumination and weather changes [9]. They can also provide accurate depth measurements. However, the modality gap between range sensor data and aerial images is much larger. To narrow such modality gap, Kummerle *et al.* [31] extracted edge points for both a satellite image and point clouds provided by a 2-D laser scanner, which are then used to estimate the position of a ground robot. RSLNet [32] and its extension [33] all use synthetic radar images generated from satellite images as the intermediate modality for matching and network training. Self-supervised learning technique is used in [33] to release the need of ground truth required in [32]. Unlike the above methods that directly use point clouds, some researchers extract high-level representations for matching. Hussein *et al.* [34] used a LiDAR to scan tree stems and matched them with tree crowns captured in an aerial image to localize a ground robot. Miller *et al.* [35] extracted semantic information for both a ground LiDAR point cloud and a satellite image to calculate their similarity, which is then used to update particle weights during filtering.

2.2.2 Ground-to-OSM Matching

Vision-based Methods

Some early works achieve global localization by comparing past trajectories of odometry with road routes in OSM. For example, Floros *et al.* [36] combined trajectory of an odometry with road information of OSM by using champfer matching, which shows a 5m average localization error in their results. Similarly, Bruaker *et al.* [37] also localized a vehicle by matching the trajectory of an odometry with road topological

information from a OSM, while a probabilistic framework is proposed for matching. Instead of directly using road information, Panphattarasap *et al.* [38] proposed a binary semantic descriptor to represent road junctions and gaps between surrounding buildings for both images and OMS. Multiple descriptors of consecutive input images are then combined together to improve the retrieval performance. Based on [38], Noe *et al.* [39] used a triplet loss to learn embeddings for both surrounding images and OSM instead of extracting binary semantic descriptors. Similarly, Zhou *et al.* [40] used the same approach to extract descriptors for both vehicle-captured images and OSM, followed by a particle filter for global localization instead of the retrieval method used in [39]. Recently, Sarlin *et al.* [41] propose OrienterNet, a deep neural network for sub-meter image localization within OSM. It requires a coarse GPS prior to build a local map from OSM and may struggle when the query image includes many unregistered elements in OSM, e.g., pedestrians or vehicles, limiting its suitability for some highly dynamic urban scenarios.

Point cloud-based Methods

Similar to some vision-based methods, Ruchti *et al.* [42] and Suger *et al.* [43] both match LiDAR odometry trajectories to OSM road information, focusing on urban environments and outer-urban environments, respectively. Unlike [42] and [43] which focus on geometric information, [44] and [45] first extract semantic information of the surroundings. A 4-bit semantic descriptor and a scan-context based OSM descriptor were proposed in [44] and [45] for matching, respectively. Global localization is then achieved using particle filter and data retrieval in [44] and [45], respectively. A localization error at about 20m on KITTI sequence-00 was reported in [44].

2.2.3 Difference from Previous Work

Similarly, our method also uses OSM as the reference map. The major difference is that we use a low-cost sky-looking camera for the observation model, which can avoid the interference from dynamic objects on roads, while the other methods ([44] uses a LiDAR, [30] and [40] a panoramic camera, [41] use front-looking cameras) may need to tackle the interference from dynamic objects on the street. In [44], a compact binary semantic descriptor (BSD) that captures the topological information (i.e., junction type) is used to bridge the modality gap between on-board measurement and OSM. Based on this compact representation, we simultaneously use geometric information (i.e., building outline) to further narrow the modality gap between sky-looking fish-eye camera images and OSM. Lastly, different from [44, 40, 35, 30, 41], which may require GPUs to achieve better efficiency, our method can directly run on a GPU-free computing platform.

2.3 Methodology

Given a sky-looking fish-eye camera image I_F at time t and a street block-sized geo-referenced OSM M_{osm}, our goal is to estimate the vehicle pose $\mathbf{x}_t$ with respect to M_{osm}. Note that M_{osm} is obtained by re-rendering from the original OSM file (see Fig. 2.1(c)) to retain only building areas, rendered in black, to facilitate descriptor extraction (see Fig. 2.4). We assume that the vehicle runs on a flat road, so the vehicle pose consists of three variables: 2-D coordinates (x, y) and orientation θ.

2.3.1 Problem Formulation

As aforementioned, we use particle filter to achieve metric localization. The key idea of particle filter is to use a number of particles to estimate the posterior probability of the

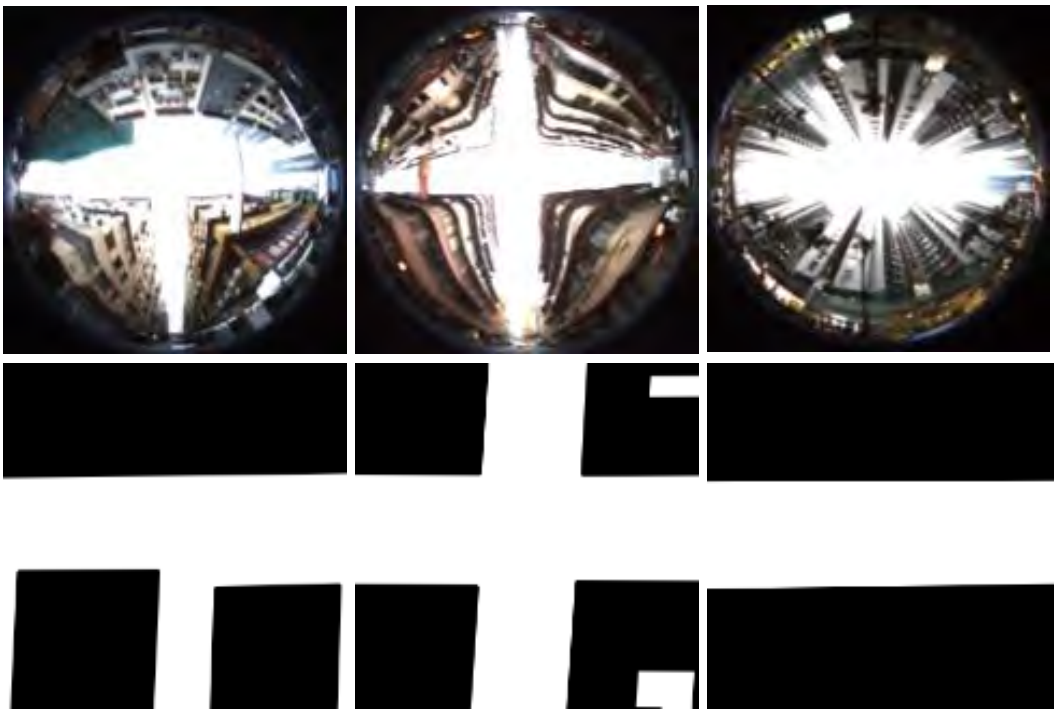


Figure 2.2: Fish-eye camera images with typical sky shapes (top row) and the OSM masks (bottom row) generated from the corresponding patches in M_{osm}.

vehicle pose. The optimal pose estimation can be achieved through the expectation of the posterior probability distribution. Mathematically, given all observations from time 1 to time t, $\mathbf{z}_{1:t}$, and all the motion control inputs, $\mathbf{u}_{1:t}$, the posterior probability $p\left(\mathbf{x}_t \mid \mathbf{z}_{1:t}, \mathbf{u}_{1:t}\right)$ can be calculated as:

$$\begin{aligned} p\left(\mathbf{x}_t \mid \mathbf{z}_{1:t}, \mathbf{u}_{1:t}\right) &= \eta p\left(\mathbf{z}_t \mid \mathbf{x}_t\right) \\ &\int p\left(\mathbf{x}_t \mid \mathbf{u}_t, \mathbf{x}_{t-1}\right) p\left(\mathbf{x}_{t-1} \mid \mathbf{z}_{1:t-1}, \mathbf{u}_{1:t-1}\right) d\mathbf{x}_{t-1}, \end{aligned} \tag{2.1}$$

where η is a normalization constant, $p\left(\mathbf{z}_t \mid \mathbf{x}_t\right)$ represents the likelihood from the observation at pose $\mathbf{x}_t$, the integral term is the prior probability of the pose [46]. Here we use a LiDAR odometry algorithm, LOAM [47] as our control input $\mathbf{u}_t$. Note that other motion estimation algorithms that can provide absolute scales can also be used as the control input here.

We use the KLD-sampling algorithm [48] to speed up the process of weight calculation

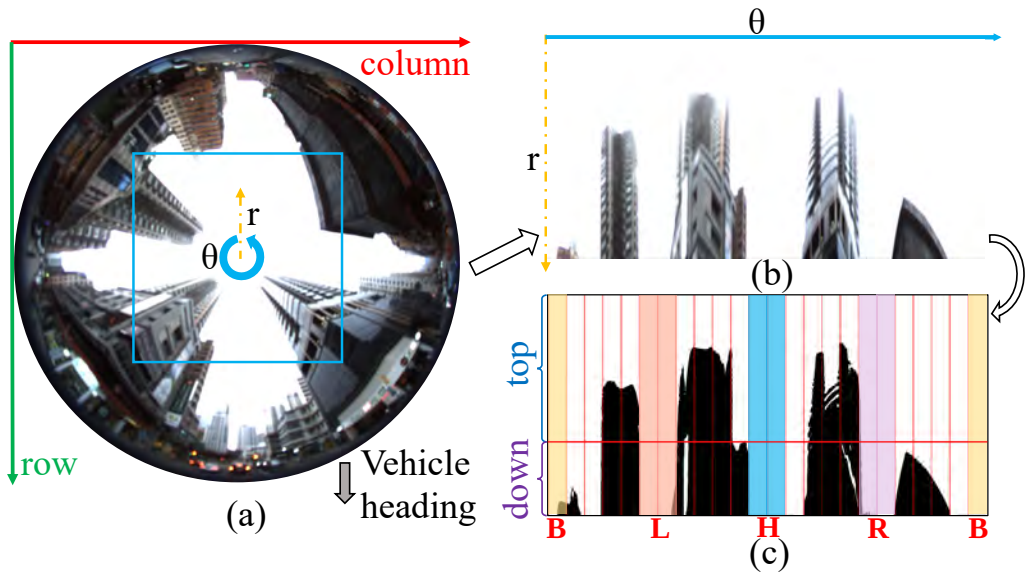


Figure 2.3: Descriptor extraction for input fish-eye image I_F. The ROI of I_F is expanded to polar coordinate and binarized to get the sky-looking mask. The final descriptor is extracted based on the sky-pixel ratios for the divided bins of the sky-looking mask.

for particles by adaptively reducing the number of particles during filtering. The key idea is using Kullback-Leibler divergence (KLD) to calculate how many particles are needed to approximate the distribution of the current vehicle pose:

$$\begin{aligned} n &= \frac{1}{2\epsilon}\chi^2_{k-1,1-\delta} \\ &= \frac{k-1}{2\epsilon}\left\{1 - \frac{2}{9(k-1)} + \sqrt{\frac{2}{9(k-1)}}z_{1-\delta}\right\}^3, \end{aligned} \tag{2.2}$$

where n is the number of the needed particles, $z_{1-\delta}$ is the upper $1-\delta$ quantile of the standard Gaussian distribution, ϵ is the upper error bound in KLD, and k is the number of bins that contain at least one particle. In our case, k represents the number of square-shape ($W_{bin} \times W_{bin}$, where W_{bin} is the length of sides in terms of pixel) bins that contain at least one particle on M_{osm}.

2.3.2 The Proposed TGH Descriptors

To associate I_F with M_{osm}, we measure the similarity between the on-line captured I_F and a particle-centered patch in M_{osm}. However, directly comparing I_F and M_{osm} is challenging due to the significant modality gap between fish-eye camera images and OSM. Interestingly, I_F and M_{osm} both encode the structural information about roads and buildings, as shown in Fig. 2.2. This shifts the problem to: *How can this information be embedded?* For road structures, binary semantic descriptor has shown its efficiency to embed junction types [44, 38]. For buildings, I_F and M_{osm} share similar geometric outlines, but describing such outlines with a vector-based descriptor is nontrivial. Instead, we represent this information implicitly using the ratio of non-building areas, which can be easily extracted from both I_F and M_{osm}. To keep the method intuitive and efficient, we adopt a handcrafted descriptor (i.e., TGH descriptor) rather than a neural network, which usually requires a large amount of paired training data and additional GPU resources for inference.

Sky-looking Mask and Descriptor Extraction

Given a I_F, we first extract a square-shaped region-of-interest (ROI) at the center of I_F instead of using all pixels. In this way, pixels from roads, pedestrians, and vehicles can be removed. Only the top parts of buildings and the sky are kept. This could make the structure appearance of I_F more close to M_{osm}. We empirically set the size of the ROI as 900×900 pixels, which can keep enough information while maintain a good efficiency. The ROI is then expanded horizontally to correct the image distortion, followed by grayscale conversion. Otsu's method [49] is then used to binarize the ROI with adaptive threshold, assigning a value of 255 to sky pixels and 0 to building pixels. A simple binarization operation is used to set the sky pixels value as 255 and the building pixels value as 0. In this way, we can get a binary sky-looking mask (see Fig. 2.3(c)).

Algorithm 1: TGH Descriptor Extraction

Input: Sky-looking fish-eye image I_F

Output: TGH Descriptor Des_{TGH}

begin

  Get the ROI of I_F and horizontally expand it, noted as I_H

  Classify pixels in I_H as sky or building via binarization operation, noted as I_B

  Divide I_B into 24 bins, $\{\text{Bin}_k\}$, $k \in \{1, 2..., 24\}$

  for each Bin_k **do**

    Count sky pixels in top part, get $\gamma(\text{Bin}_k^T)$

    Count sky pixels in down part, get $\gamma(\text{Bin}_k^D)$

  end

  $Des_{\text{geo\_T}} \leftarrow \{\gamma(\text{Bin}_k^T)\}, \quad Des_{\text{geo\_D}} \leftarrow \{\gamma(\text{Bin}_k^D)\}$

  $Des_{\text{geo}} \leftarrow \{Des_{\text{geo\_T}}, Des_{\text{geo\_D}}\}$

  for each bit_i **do**

    Find two corresponding bins, Bin_{k1} and Bin_{k2}

    if $(\gamma(\text{Bin}_{k1}^T) \geq \tau_1^T \ \& \ \gamma(\text{Bin}_{k1}^D) \geq \tau_1^D) \ \&$
    $(\gamma(\text{Bin}_{k2}^T) \geq \tau_1^T \ \& \ \gamma(\text{Bin}_{k2}^D) \geq \tau_1^D)$ **then**

      $\text{bit}_i = 1$

    end

    else if $(\gamma(\text{Bin}_{k1}^T) \geq \tau_2^T \ \& \ \gamma(\text{Bin}_{k1}^D) \geq \tau_2^D) \ \|$
    $(\gamma(\text{Bin}_{k2}^T) \geq \tau_2^T \ \& \ \gamma(\text{Bin}_{k2}^D) \geq \tau_2^D)$ **then**

      $\text{bit}_i = 1$

    end

    else

      $\text{bit}_i = 0$

    end

  end

  $Des_{\text{topo}} \leftarrow \{\text{bit}_i\}, i \in \{1, 2, 3, 4\}$

  return $Des_{\text{TGH}} \leftarrow \{Des_{\text{geo}}, Des_{\text{topo}}\}$

end

Similar to the binary semantic descriptors (BSD) [44], we design our descriptor by first dividing the sky-looking mask into 24 bins along the column direction, in which each bin covers 15° out of 360°. Each bin is then divided into 2 parts, that is, the top part (i.e., the first 2/3 rows of the mask) and the down part (i.e., the last 1/3 rows of the mask), see Fig. 2.3. Unlike [44] or [38] that use only topological information, such as road junctions and gaps between buildings, we use both topological information and geometric information to extract our descriptors. For the geometric information, we count the sky pixels in both top and down parts in each bin to respectively calculate the sky-pixel ratios, denoted as $\gamma(\text{Bin}_k^T)$ and $\gamma(\text{Bin}_k^D)$, respectively. Bin_k represents the k-th bin, T is short for the top part, and D is short for the down part. The descriptor vector of the geometric information can be obtained as Des_{geo}=$[\gamma(\text{Bin}_1^T), ..., \gamma(\text{Bin}_{24}^T), \gamma(\text{Bin}_1^D), ..., \gamma(\text{Bin}_{24}^D)]$, where Des and geo is short for descriptor, and geometric, respectively. For the topological information, we design a 4-bit binary descriptor Des_{topo} to indicate the presence of a road in the vehicle's heading direction(H), back direction(B), left direction(L), and right direction(R). Each bit covers two bins of the mask, as shown in Fig. 2.3(c). If the sky pixel ratios of such two bins satisfy conditions listed in Algorithm 1 (see thresholds setting in Tab. 2.1), the corresponding bit of Des_{topo} is set as 1 (i.e., a road exists in this direction), otherwise 0. Let bit_i denotes the i-th bit of Des_{topo}, where $i \in \{1, 2, 3, 4\}$ follows the *heading-back-left-right* order. The whole process of the TGH descriptor extraction is shown in Algorithm 1.

OSM Mask and Descriptor Extraction

Given a particle par^i with pose $(x_p^i, y_p^i, \theta_p^i)$, where i represents i-th particle, we first extract a circle-shaped ROI centered at (x_p^i, y_p^i) with a diameter of ϕ_p (see 2.4). We then orient the ROI according θ_p^i before the descriptor extraction (see Fig. 2.4). Similarly, we convert the OSM ROI as grayscale and then binarize it, where non-building pixels approximately correspond to the sky pixels in the sky-looking mask.

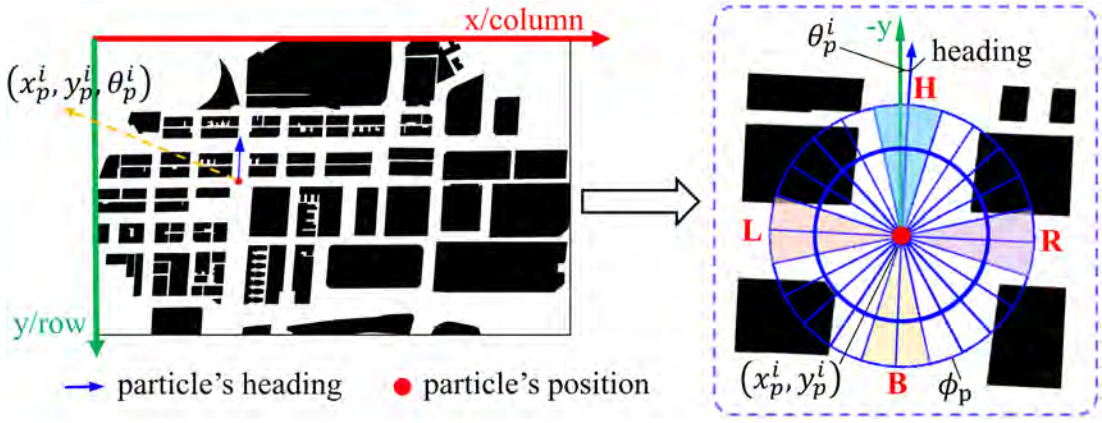


Figure 2.4: Example of descriptor extraction for a given particle on OSM. A cylinder coordinate is adapted on the OSM to get the ROI. Then the final descriptor can be obtained as the same way for fish-eye image.

Similar to the TGH descriptor for I_F, we divide the ROI of OSM mask into 24 bins based on a cylinder coordinate, where each bin covers 15° out of 360°. Each bin is then further divided into center part (inner 2/3 radius) and marginal part (outer 1/3 radius), as shown in Fig. 2.4. Here we do not expand the ROI horizontally, since there is not distortion in M_{osm} and circle-shaped ROI is much more close to the raw I_F. Then we count pixels belonged to the non-building area within both center part and marginal part in each bin to calculate the "sky"-pixel ratio, denoted as $\gamma(\text{Bin}_k^C)$ and $\gamma(\text{Bin}_k^M)$, respectively. C is short for center and M is short for marginal. The geometry part $Des_{\text{geo}}^{par^i}$ and topology part $Des_{\text{topo}}^{par^i}$ of $Des_{\text{TGH}}^{par^i}$ for particle par^i are extracted by following the same process in Algorithm 1 (i.e., line 5-26). However, different thresholds are chosen, considering the gap between polar and cylinder coordinate. More details can be found in Tab. 2.1.

2.3.3 Observation/Weighting Model

The observation, or weighting model in particle filter is used to update particle weights during the filtering process. A higher weight indicates that the live sensor input data

is much more likely captured when the robot is under a state as like the given particle. In our case, such similarity/weight is computed by the distance between descriptors extracted from I_F and M_{osm}.

Given a TGH descriptor $Des^F_{\text{TGH}} = \{Des^F_{\text{geo}}, Des^F_{\text{topo}}\}$ of I_F and $Des^{par^i}_{\text{TGH}} = \{Des^{par^i}_{\text{geo}}, Des^{par^i}_{\text{topo}}\}$ of particle par^i at time t, the final weight for par^i is the combination of two parts, including topology similarity $w^{par^i}_{topo}$ and geometry similarity $w^{par^i}_{geo}$. Similar to [44] and [40], we use Hamming distance d_{ham} to calculate the topology similarity as follow:

$$w^{par^i}_{topo} = 1 - 0.2 d_{ham}(Des^F_{\text{topo}}, Des^{par^i}_{\text{topo}}), \tag{2.3}$$

The reason for choosing 0.2 as the factor is to make sure those particles with higher Hamming distance can be still kept in a certain probability after re-sampling. As for the geometry similarity, we use *cos* distance d_{cos}, which is widely adapted to evaluate the similarity between two distributions. Des^F_{geo} and $Des^{par^i}_{\text{geo}}$ can be regarded as distributions of sky or building pixels. Specifically, we first calculate *cos* distance between the top part $Des^F_{\text{geo\_T}}$ of Des^F_{TGH} and the center part $Des^{par^i}_{\text{geo\_C}}$ of $Des^{par^i}_{\text{TGH}}$. Similarly, we then calculate the *cos* distance between the corresponding down part and marginal part. To reduce the gap caused by the difference of bin size and the bin shape used for extracting Des^F_{TGH} and $Des^{par^i}_{\text{TGH}}$, we combine these two *cos* distances by weighting factors. The final geometry similarity $w^{par^i}_{geo}$ can be obtained as follow:

$$w^{par^i}_{geo} = \lambda d_{cos}(Des^F_{\text{geo\_T}}, Des^{par^i}_{\text{geo\_C}}) + \omega d_{cos}(Des^F_{\text{geo\_D}}, Des^{par^i}_{\text{geo\_M}}), \tag{2.4}$$

where λ and ω are weights satisfying $\lambda + \omega = 1$. The final weight of particle par^i can be obtained as follow:

$$w^{par^i} = \alpha w^{par^i}_{topo} + \beta w^{par^i}_{geo}, \tag{2.5}$$

where α and β are weights satisfying $\alpha + \beta = 1$. More details about values of weights are shown in Tab. 2.1.

Table 2.1: Parameters used in TGH descriptor and KLD-sampling

Thresholds for Des^{F}_{topo} / $Des^{par^i}_{\text{topo}}$				Weights for Des_{TGH}				KLD-sampling		
τ_1^T	τ_1^D	τ_2^T	τ_2^D	λ	ω	α	β	ϵ	δ	W_{bin}
0.7 / 0.7	0.7 / 0.6	0.8 / 0.8	0.8 / 0.8	0.4	0.6	0.6	0.4	0.15	0.1	25

2.4 Experimental Results and Discussions

2.4.1 The Dataset

We build our dataset based on the UrbanLoco dataset [21] by supplementing the OSM data. The UrbanLoco dataset contains 13 sequences collected in San Francisco and Hong Kong, covering a total length of over 40 km. There are front-looking camera images, LiDAR point clouds, and RTK GNSS poses recorded in each sequence. However, only four sequences have sky-looking fish-eye camera images, including HK-Data20190426-1, HK-Data20190426-2, HK-Data20190316-1, and HK-Data20190316-2. These four sequences are collected in urban canyon areas (i.e., the Whampooa and Ma Tau Kok areas in Hong Kong), as shown in Fig. 2.5. We make our dataset using the aforementioned four sequences. We sample the fish-eye images at the GNSS collection rate (i.e., 1 Hz) to get the ground truth pose. Since the down-sampled sequences are too short, we reverse the sequence (i.e., seems like reversing the car) at the end of the original sequence. In this way, the sequence is expanded twice long, renamed as Seq-01, 02, 03, and 04, respectively, as shown in Fig. 2.7. Note that RTK-GNSS trajectories of Seq-03 and 04 are not accurate enough due to severe signal occlusions along the data collection route. So, we refine their ground truth trajectories based on the relative trajectories estimated by LOAM.

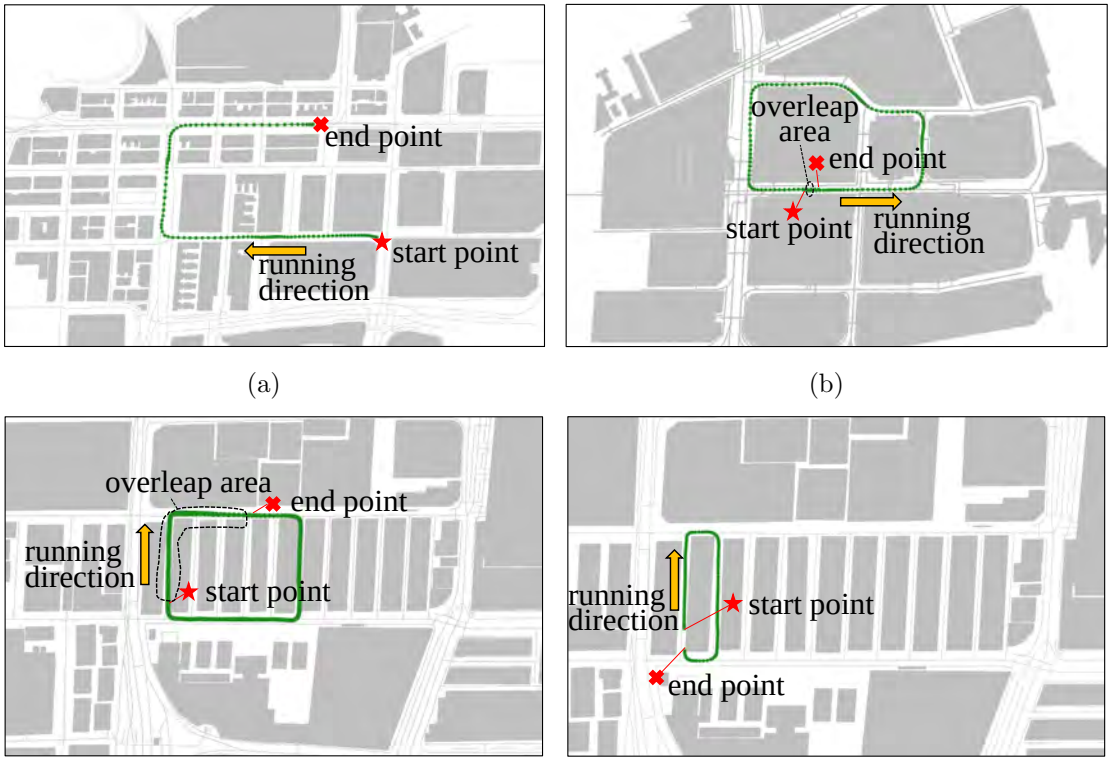


Figure 2.5: The M_{osm} with the trajectory of the vehicle. Figure (a), (b), (c), and (d) corresponds to Seq-01, 02, 03, and 04 without expansion, whose length is 562.5m, 723.9m, 657.0m, and 232.5m, respectively.

2.4.2 Experimental Setup

We assume that vehicles always run on roads and follow the traffic rules (e.g., cannot make u-turns in the middle of roads). Thus, we evenly distribute $40,000$ particles on the road area, and constrain the particle orientation θ_p within a range with a tolerance φ along the road orientation to reduce the number of particles that have invalid poses (e.g., particles in the middle of the road but with a 90° heading direction). So, given a road with $a°$ orientation, the orientations of particles within this road should satisfy the requirement: $\theta_p \in [a - \frac{\varphi}{2}, a + \frac{\varphi}{2}]$ or $[a + 180 - \frac{\varphi}{2}, a + 180 + \frac{\varphi}{2}]$. The a is calculated with the two junctions at the two ends of the road. In this work, we set φ as 30° and evenly distribute the θ_p with a resolution of 1°. We use a larger ROI of I_F for Seq-02,

i.e., 1300×1300 pixels. We set ϕ_p as around 50m for Seq-01 and 02, and around 25m for Seq-03 and 04 because the building density is much higher in these two sequences. Besides, $W_{bin} = 15$ is used for Seq-03 and 04 to better estimate the distribution of sampled particles in narrow street areas. To achieve real-time performance, we pre-extract descriptors from the given M_{osm} to build a database, which takes about 20s for $200,000$ particles on an intel core i7 computer. During filtering processes, $Des^{par^i}_{\text{TGH}}$ can be approximated by the descriptor that is extracted at the closest location to par^i in the database. A KD-search tree is used to accelerate searching. The descriptor for the closest point in the database might have a different orientation from the particle par^i. So, we further align the retrieved descriptor with par^i according to θ^i_p by sifting the column of the retrieved descriptor. The more data points in the database are sampled, the more accurate such approximation would be. Empirically, a larger map M_{osm} usually needs more data points to build such a database.

2.4.3 Performance Evaluation

Baselines

Since work [44, 40] are not open-sourced, we are unable to directly compare with them. So, we create some baselines for comparison. The work [44] extracts BSD descriptors for both M_{osm} and on-line LiDAR scans to update particle weights during filtering. Similarly, we use road junctions on the heading, back, right, and left of the vehicle to build the BSD descriptor. In the first baseline, LOAM is used to provide pose estimation to update the control input $\mathbf{u}_t$. A perfect BSD-based observation model is used. Specifically, we first find the corresponding coordinates (i.e., x, y, θ) on OSM according to the ground truth poses of the vehicle. Then BSD descriptors extracted at these positions from the OSM are used as on-line observations. So, the BSD descriptors for the on-line observations are exactly the same as those for the particles whose locations are identical to the on-line sensor in the OSM. We name

this baseline as the BSD-based method (`BSD`). We use this baseline as the upper limit in principle for the BSD-based method when using a practical motion model. As for the second baseline, we only set particle weights as 0 when particles are outside the road area, namely, the Road Net-based method (`RN`). Motion estimation from LOAM is used to update $\mathbf{u}_t$. This baseline only uses road information in its observation model like [36]. In the third baseline, we replace the original observation model in our method with a perfect observation model similar to the first baseline `BSD`. So, the TGH descriptors for the on-line observations are exactly the same as those for the particles whose locations are identical to the on-line sensor in the OSM. Such baseline is noted as ours with Perfect Observation model (`ours-PO`), which refers to the upper limit for the performance of the TGH-based method when using a practical motion mode.

We run our method and three baselines on Seq-01, 02, 03, 04, and their reversed versions, totally eight sequences. We run ten times of all the methods on each sequence. The filtering process is considered as converged when the standard deviation of all the current particles pose are less than a pre-defined threshold. Here, we set the standard deviation as 40 *pixel* (around 6m) for both x and y, as well as 10° for θ. Some localization results in one runtime are as shown in Fig. 2.7.

Localization Accuracy and Convergence

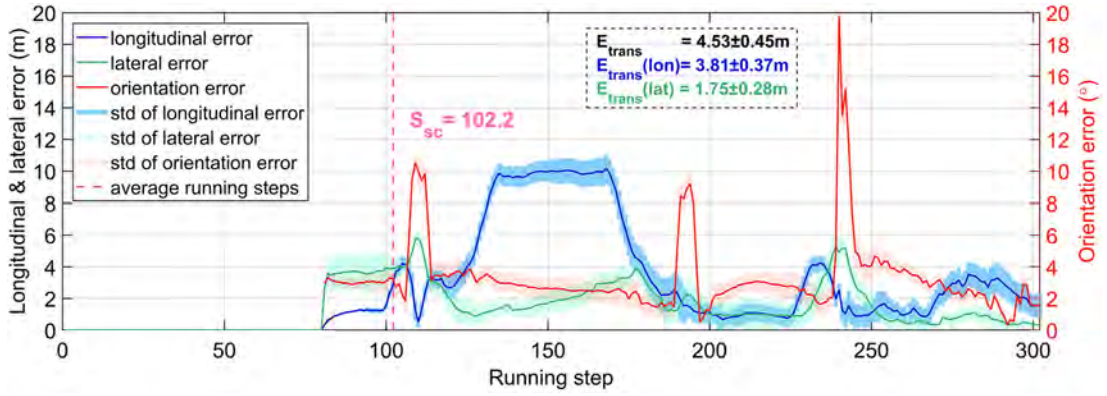
We use the successful convergence rate $P_{sc} = |L|/n_{test}$ and average running steps to achieve successful convergence, $S_{sc} = \frac{1}{|L|}\sum_{l\in L} s^l$, to evaluate the performance of convergence. $|L|$ is the cardinality of L, in which $L = \{l\}$ is the set of testing runs that can achieve successful convergence. n_{test} represents total testing times. s^l is the number of running steps for successful convergence in l-th testing run. Once a filtering process is successfully converged, we calculate the average error for translation and orientation as $E_{trans} = \frac{1}{N-s^l}\sum_{j=s^l}^{N}\sqrt{\left(\tilde{x}_j^l - x_j\right)^2 + \left(\tilde{y}_j^l - y_j\right)^2}$ and

Table 2.2: Average E_{trans} and average E_{ori} with standard deviation, average running steps for successful convergence S_{sc}, and successful convergence rate P_{sc} in UrbanLoco dataset.

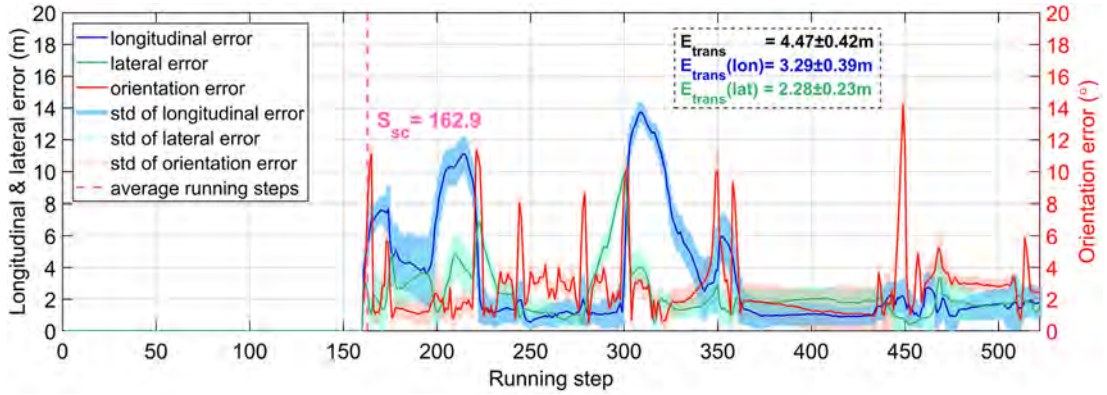
Metrics	Method	Sequence							
		01	01-reverse	02	02-reverse	03	03-reverse	04	04-reverse
E_{trans}	ours	4.53±0.45	3.98±0.38	4.47±0.42	3.66±0.66	2.84±0.23	2.84±0.26	N/A	N/A
	BSD	3.84±0.25	3.16±0.22	3.17±0.17	2.98±0.42	1.21±0.14	1.47±0.83	N/A	N/A
	RN	N/A	N/A	3.93±0.41	3.01±0.68	N/A	N/A	N/A	N/A
	ours-PO	**3.75±0.31**	**2.98±0.28**	**2.56±0.21**	**2.45±0.32**	**0.73±0.11**	**0.69±0.09**	**0.23±0.05**	N/A
E_{ori}	ours	**3.29±0.32**	**2.87±0.22**	**2.76±0.37**	**2.74±0.32**	1.40±0.19	1.32±0.24	N/A	N/A
	BSD	3.58±0.31	3.05±0.17	3.33±0.34	3.34±0.29	0.83±0.14	0.87±0.10	N/A	N/A
	RN	N/A	N/A	3.49±0.49	3.54±0.53	N/A	N/A	N/A	N/A
	ours-PO	3.37±0.30	3.00±0.23	2.91±0.23	3.03±0.32	**0.68±0.11**	**0.53±0.09**	**0.34±0.14**	N/A
P_{sc}	ours	**1.0**	**1.0**	**1.0**	**1.0**	0.9	0.8	0.0	0.0
	BSD	**1.0**	**1.0**	**1.0**	**1.0**	0.9	0.8	0.0	0.0
	RN	0.0	0.0	**1.0**	**1.0**	0.0	0.0	0.0	0.0
	ours-PO	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**0.9**	**0.2**	0.0
S_{sc}	ours	102.2	102.6	162.9	61.3	230.4	**253.6**	N/A	N/A
	BSD	104.8	104.1	163.2	55.9	241.2	263.9	N/A	N/A
	RN	N/A	N/A	207.0	84.3	N/A	N/A	N/A	N/A
	ours-PO	**101.4**	**95.0**	**135.8**	**47.0**	**189.4**	254.6	**287.0**	N/A

E_{trans}(unit: m) and E_{ori}(unit: °) refer to average E_{trans} and average E_{ori} here, respectively. "N/A" represents not-converged or failed-converged cases. The best results for different metrics on each sequence are highlighted in bold font.

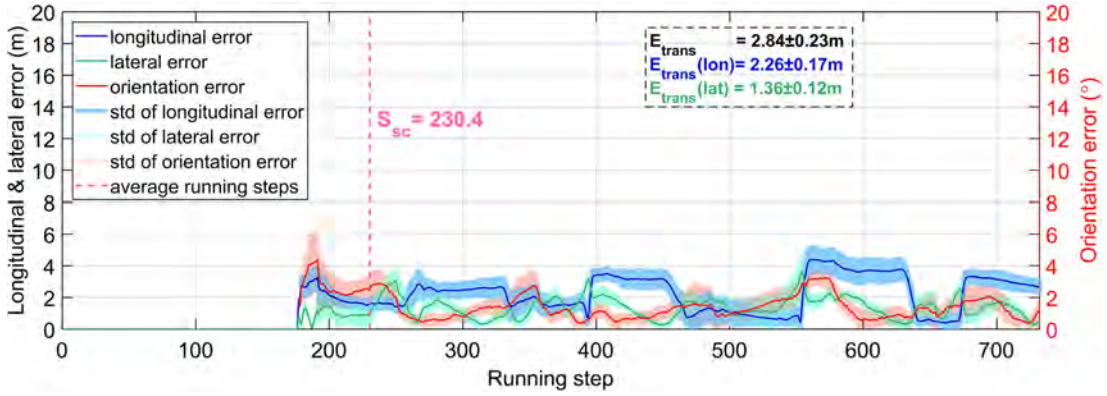
$E_{ori} = \frac{1}{N-s^l}\sum_{j=s^l}^{N}\left|\overleftarrow{\mathbb{R}}\left(\mathbb{R}(\tilde{\theta}_j^l)^{-1}\mathbb{R}(\theta_j)\right)\right|$. j refers to j-th frame. $(\tilde{x}_j^l, \tilde{y}_j^l, \tilde{\theta}_j^l)$ is the estimation pose of the vehicle at frame j in l-th testing run. (x_j, y_j, θ_j) is the ground-truth pose. N is the frame number of the testing sequence. $\mathbb{R}(\cdot) \in SO(2)$ is the rotation matrix of the given orientation. $\overleftarrow{\mathbb{R}}(\cdot)$ is the orientation of the given 2D rotation matrix. The average E_{trans} and average E_{ori} with standard deviations are then calculated based on all the successful convergence filtering processes, which are used to evaluate the localization accuracy. As shown in Tab. 2.2, our method can achieve similar localization accuracy as BSD. The values of average E_{trans} for all the successful converged sequences are less than 4.6m. The values of average E_{ori} are less than 3.3°.



(a)



(b)



(c)

Figure 2.6: The average absolute error with standard deviation for each single running step after successful convergence. The first, middle, and bottom row represents results for Seq-01, 02, and 03, respectively.

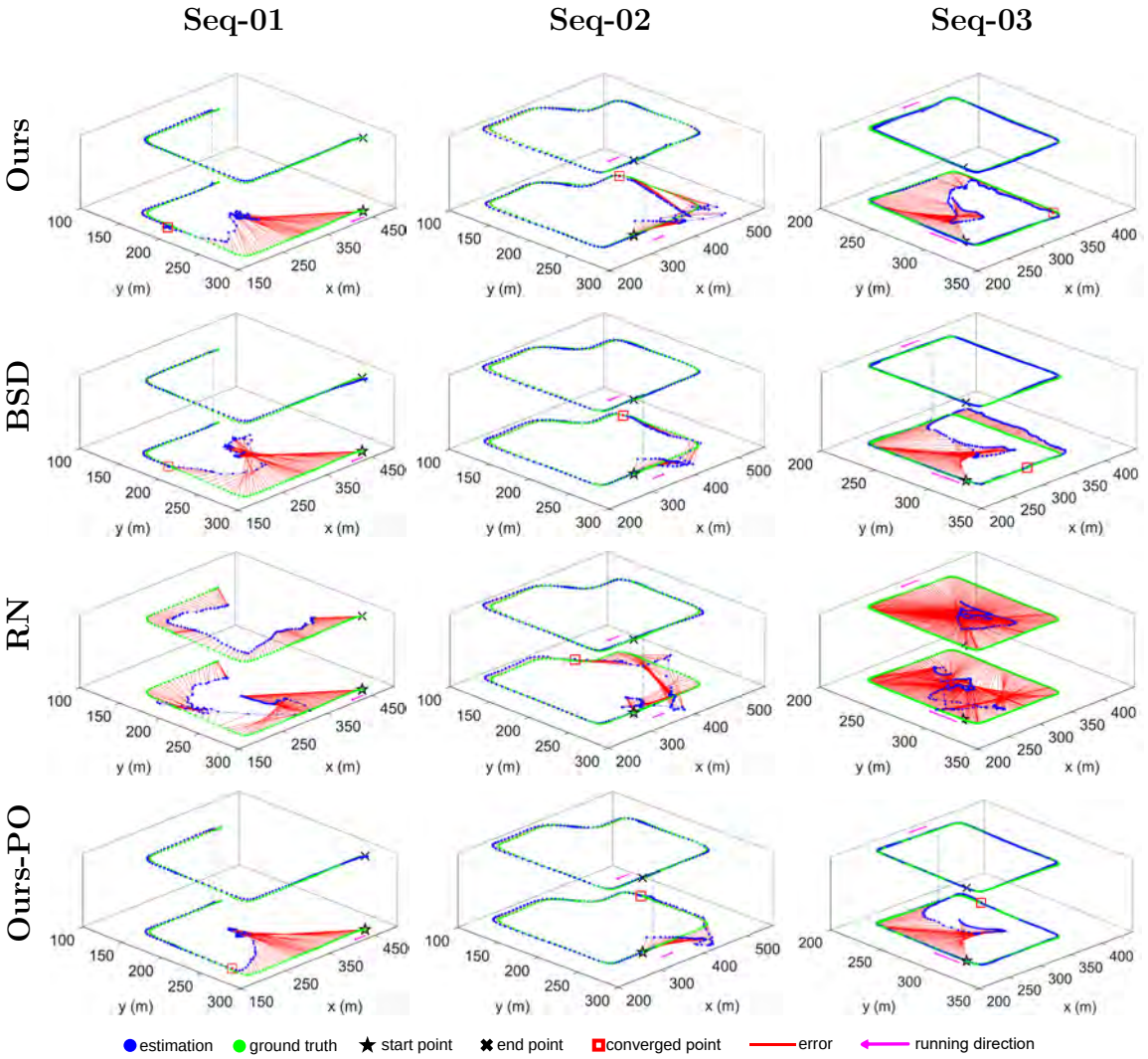


Figure 2.7: Examples for qualitative performance of localization. Figures in each row are based on the same method. In each picture, the top trajectory is the expanded part, and the dotted line connects the place where we reverse the sequence (see Part A of section IV).

When using our method on all sequences except sequence 02-reverse, the average running steps for successful convergence are smaller compared to `BSD`. This indicates that our method can successfully converge faster than `BSD`. Our method can also achieve a high successful convergence rate. Indeed, `BSD` has a perfect observation model without noise, which means such results are very difficult or even impossible to be obtained

when using practical sensors. It is worth noting that our method outperforms all the other baselines when we using a perfect TGH-based observation model from the perspective of convergence performance with high competitive localization accuracy. This shows that our TGH-based observation model has a higher upper limit than the BSD-based observation model. We interestingly find that `RN` baseline can only localize the vehicle on the Seq-02 and 02-reverse with larger S_{sc}. We consider RN-based method as a kind of "lazy" particle filter, which treats every particle equally. It only drops particles when they are not on roads. The reason why such "lazy" filter can successfully converge could be that the trajectory of the running vehicle is unique in the M_{osm}, while this is not common in urban environments. We also observe that in challenging scenarios, successful convergence can be difficult to achieve, leading to failures. For instance, on the Seq-04, only `ours-PO` manages to converge successfully with a P_{sc} value of 0.2. Moreover, our method and all the other baselines are unable to successfully converge on Seq-04-reverse. We conjecture the primary reason is the lack of enough consecutive and distinguished observations. The junction types, building layouts, structures, and even appearances are highly similar and repeated in Seq-03 and Seq-04. In cases with insufficient observations, such as Seq-04 and Seq-04-reverse, our method fails. While when more consecutive observations are available, as in Seq-03 and 03-reverse, our method still able to successfully converge even in such challenging environments. To enhance the performance in challenging scenarios with limited observations. We believe that incorporating features such as street signs and shop names captured in images and registered in OSM provide promising potentials.

We also visualize the average absolute errors of localization, including $\epsilon_j(\text{trans}) = \frac{1}{|L|}\sum_{l\in L}\sqrt{\left(\tilde{x}_j^l - x_j\right)^2 + \left(\tilde{y}_j^l - y_j\right)^2}$ and $\epsilon_j(\text{ori}) = \frac{1}{|L|}\sum_{l\in L}\left|\overleftarrow{\mathbb{R}}\left(\mathbb{R}(\tilde{\theta}_j^l)^{-1}\mathbb{R}(\theta_j)\right)\right|$ for each running step-j once the filtering is converged, as well as their standard deviations. The $\epsilon_j(\text{trans})$ is further split into longitudinal (i.e., along the heading of the vehicle) and lateral errors (i.e., perpendicular to the heading of the vehicle). As shown in Fig. 2.6, our method can provide a good estimation for both translation and orientation, where

the errors are respectively less than 10m and 20° in most of the single running step. We can also observe that the longitudinal errors are generally larger than the lateral errors. We conjecture that the primary reason lies in the road width being significantly smaller than its length. The field of view in the lateral direction of the road is more sensitive than in the longitudinal direction, making fish-eye images captured at different locations across the road width more distinguishable. Therefore, it would be better to set different uncertainties for longitudinal and lateral estimations when using our method in navigation or localization systems.

Dead-reckoning Distance Traveled before Successful Localization

Similar to [50], we calculate the probability of travelling a given distance x without successful localization in the target map, $P(x)$. In [50], it follows a key-frame-based place recognition manner, while ours follows a filtering manner. Therefore, successful localization in our case can only occur after the filtering process has converged. To determine whether successful localization has occurred, we calculate Relative Translation Error (RTE) and Relative Rotation Error (RRE) for each step after successful convergence (more details about RTE and RRE can be found in [12]). Localization is considered successful when RTE<7.5m and RRE<10°. For each test, a uniformly distributed random frame is selected as the starting point for the filtering process. We conduct 100 trials per sequence and record the travelled distance at the first instance of successful localization. Results for each scenario (e.g., Scene-01 refers to Seq-01 and Seq-01-reverse combined) are presented in Fig. 2.8. The vehicle successfully localizes within 600m and 700m in 95% of the time for Scene-01 and Scene-02, respectively. In Scene-03, due to more challenging environmental conditions, the vehicle localizes within 850m in 95% of the time.

$$P(x) = \frac{\begin{array}{c}\text{Sum of distance travelled without} \\ \text{localization for greater or equal to } x \text{ meters}\end{array}}{\text{Total distance travelled}} \tag{2.6}$$

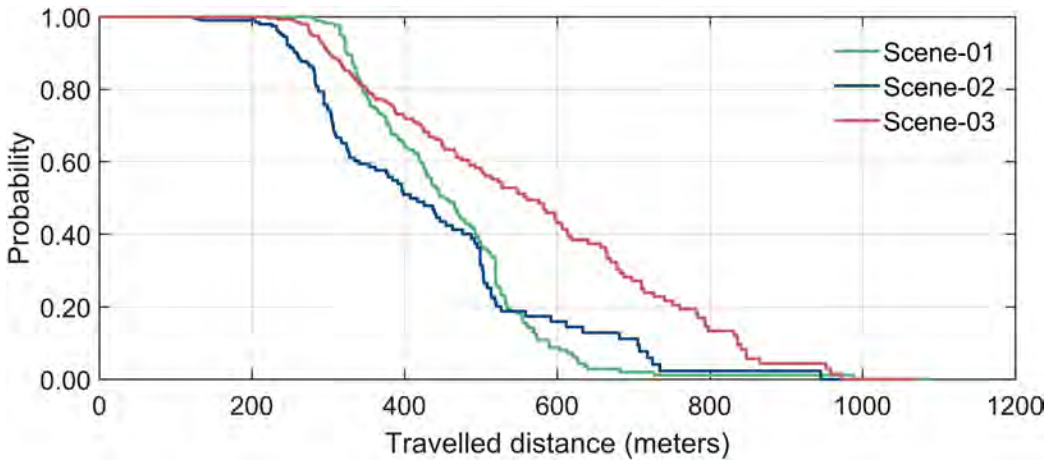


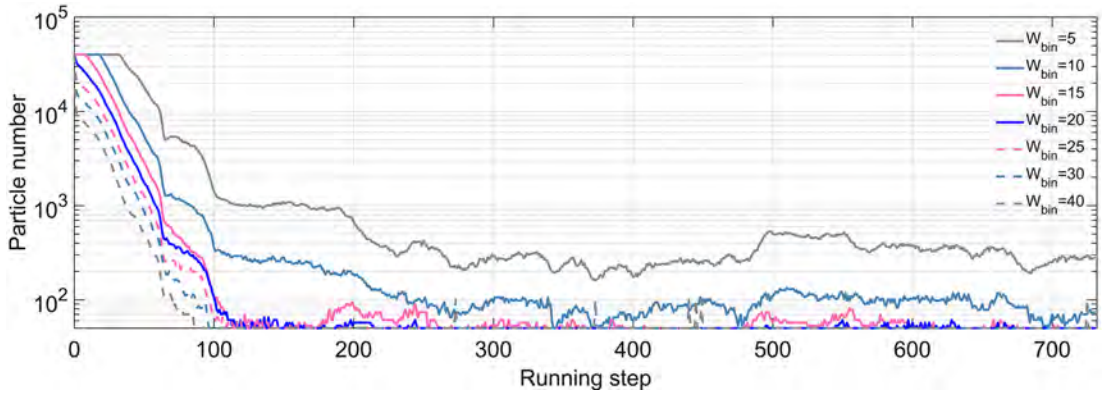
Figure 2.8: Probability of travelling a given distance before successfully localizing.

2.4.4 Ablation Study

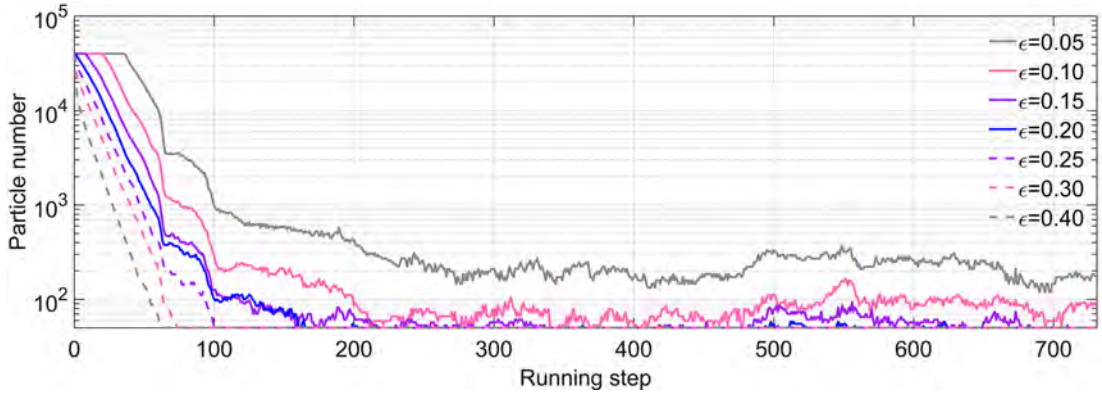
In this part, we investigate how much the topological information and geometric information contribute in TGH. We run our method with Des_{TGH}, Des_{topo}, or Des_{geo} on all the sequences except Seq-04 and 04-reverse for ten times, respectively. Results for P_{sc} and S_{sc} are displayed in Tab. 2.3. We can see that the combination of Des_{geo} and Des_{topo} is able to improve the robustness of global localization. For example, using Des_{TGH} can achieve higher values of P_{sc} on the sequences 02, 03, and 03-reverse compared to using only Des_{geo} or Des_{topo}. Meanwhile, using both topological information and geometric information allows a better convergence speed on the sequences 01-reverse and 03-reverse. In general, only using Des_{topo} and Des_{geo} is not very stable, they sometimes even lead to failures, while the proposed Des_{TGH} allows more robust performance.

2.4.5 Parameter Tuning for Re-sampling

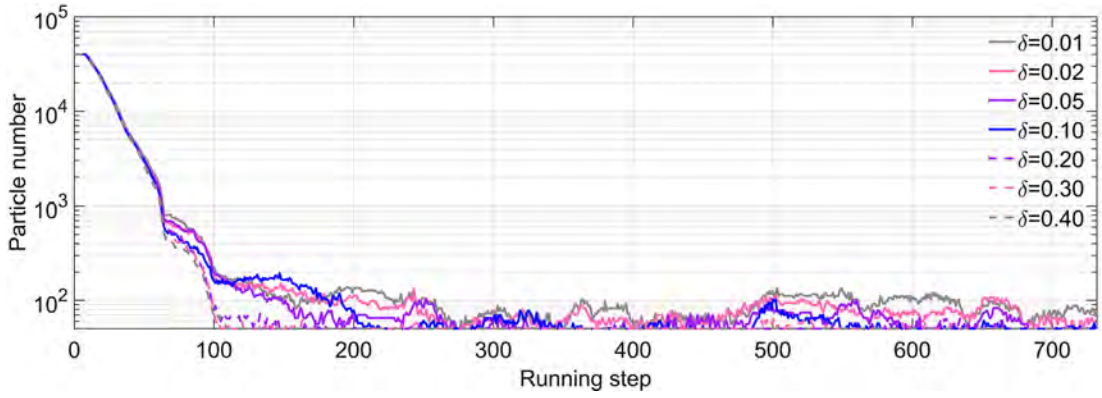
To have a better understanding of the influence of KLD-based re-sampling on our method, we investigate the influence of re-sampling parameters (i.e., ϵ, δ, and W_{bin}) on the number of particles, P_{sc}, and S_{sc}. For each parameter, we change its value



(a)



(b)



(c)

Figure 2.9: The influence of re-sampling parameters on the change of particles number. The sub-figures (a), (b), and (c) represent the influences of W_{bin}, ϵ, and δ on particles number, respectively.

Table 2.3: P_{sc} and S_{sc} for using descriptor Des_{TGH}, Des_{topo}, or Des_{geo}.

Sequence	Des_{TGH}		Des_{topo}		Des_{geo}	
	P_{sc}	S_{sc}	P_{sc}	S_{sc}	P_{sc}	S_{sc}
01	**1.0**	102.2	**1.0**	<u>94.1</u>	0.9	110.4
01-reverse	**1.0**	<u>102.6</u>	**1.0**	114.0	**1.0**	114.5
02	**1.0**	162.9	0.5	<u>162.3</u>	0.0	N/A
02-reverse	**1.0**	61.3	0.9	70.3	**1.0**	<u>51.5</u>
03	**0.9**	230.4	0.6	<u>220.0</u>	0.6	229.5
03-reverse	**0.8**	<u>253.6</u>	0.4	263.3	0.1	365.0

The best result for each row is highlighted. Different formats are used to represent the best results for different metrics (i.e., bold font for P_{sc} and underline for S_{sc}).

while keeping the others unchanged, and then we run our method ten times. The experiment is conducted on Seq-03. Examples of the change of particle numbers during the filtering process can be found in Fig. 2.9, where the y-axis uses the *log*-scale. It can be easily found that ϵ has the greatest influence on the number of particles. When ϵ becomes larger, the number of particles reduces rapidly. Differently, the relation between the number of particles and W_{bin} can not be easily determined by the partial derivative of (2.2). When W_{bin} is larger, the value of the statistic result k is generally smaller, leading to the fast reduction of the number of particles (see Fig. 2.9(a)). As shown in Fig. 2.9(c), the influence of δ on the number of particles is small during the filtering process. This is because $\sqrt{\frac{2}{9(k-1)}}$ is much smaller than $z_{1-\delta}$ with different values of δ, especially at the early stage of the filtering process. Intuitively, filtering processes usually converge faster when the number of particles decreases faster. As displayed in Tab. 2.4, we can find that the vehicle can localize itself faster with a larger ϵ or a larger W_{bin}. However, the filtering process becomes not very stable with larger value of ϵ and W_{bin}, resulting in a low successful convergence rate P_{sc}. Empirically, smaller ϵ and W_{bin} could greatly reduce the failure chance of

Table 2.4: P_{sc} and S_{sc} for different re-sampling parameters.

$\epsilon = 0.15,\ \delta = 0.1$			$W_{bin} = 15,\ \delta = 0.1$			$\epsilon = 0.15,\ W_{bin} = 15$		
W_{bin}	P_{sc}	S_{sc}	ϵ	P_{sc}	S_{sc}	δ	P_{sc}	S_{sc}
5	**1.0**	392.8	0.05	**1.0**	329.6	0.01	0.9	262.6
10	**1.0**	293.6	0.10	**1.0**	253.5	0.02	**1.0**	248.4
15	0.9	230.4	0.15	0.9	230.4	0.05	0.9	252.2
20	0.9	234.9	0.20	0.8	213.1	0.10	0.9	230.4
25	0.8	225.3	0.25	0.7	221.0	0.20	**1.0**	222.9
30	0.7	201.4	0.30	0.5	197.4	0.30	0.9	260.2
40	0.3	126.3	0.40	0.3	155.5	0.40	0.8	222.3

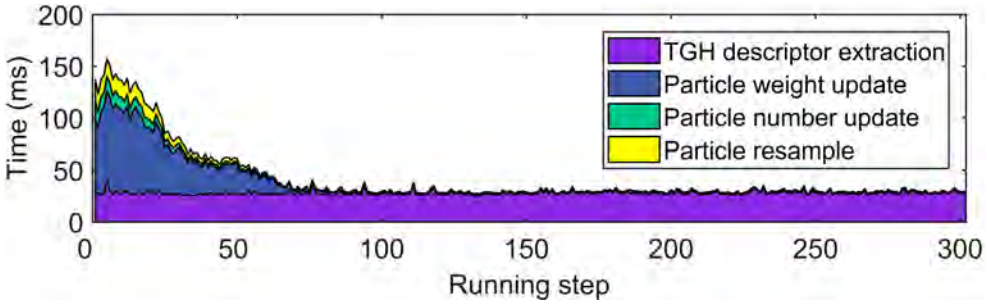


Figure 2.10: Computational cost of the filtering process for Seq-01 using a NVIDIA Jeston Xavier NX kit.

localization. However, it might cause too slow reduction for the number of particles, which usually requires more computational time. It might be better to use smaller values in more challenging environments, where usually have highly similar building structures, high building densities, and narrow streets.

2.4.6 Computational Cost

To show the efficiency of our method, we evaluate the computational costs of different parts during the filtering process in Tab. 2.5. They are measured on three different platforms: a PC with i7-11700KF CPU (3.6 GHz) and 32-GB RAM, an Intel NUC Kit with i5-1135G7 CPU (2.4 GHz) and 16-GB RAM, and a NVIDIA Jetson Xavier

Table 2.5: Computational cost of the filtering process implemented on different platforms.

Platform	i7-PC		Intel NUC		NVIDIA Xavier NX	
	μ	Λ	μ	Λ	μ	Λ
Dataset loading (s)	**3.57**	3.63	4.07	4.08	15.92	16.66
Filter initialization (ms)	**47.92**	48.06	58.40	58.61	212.06	214.78
TGH for I_f (ms)	**5.67**	6.87	7.47	8.38	27.58	40.22
Weight update (ms)	**1.05**	10.41	2.96	23.18	10.72	40.22
Particle number update (ms)	**0.39**	5.26	0.51	6.49	1.37	17.59
Particle re-sample (ms)	**0.40**	4.12	0.58	6.42	1.90	22.25
Total for single step (ms)	**7.51**	26.66	11.52	44.47	41.57	178.87

μ refers to the average time cost. Λ is the maximum time cost.

NX kit with 6-core Careml ARM CPU and 8-GB RAM. We run our method three times on Seq-01 on each platform. We then calculate the average time cost μ for each part of our method during filtering. Λ is the maximum time cost of a single running step during filtering.

Specifically, the dataset loading contains loading the M_{osm} descriptor dataset and constructing the KD tree, which takes the longest time. The particle filtering initialization can be finished within 220ms. We consider these two parts as the initial phase, which only needs to be conducted once. As for the other parts (i.e., Des_{TGH} extraction time, time for particle weights update, time for particle number update, and KLD-based particle re-sampling time), the total average time cost is less than 50ms on all the three platforms. Generally, the proposed method can run fast with real-time performance.

2.5 Conclusion

In this work, we present a novel method to globally localize a vehicle by using a sky-looking fish-eye camera and OSM in GNSS-degraded environments. The modality gap between fish-eye images and OSM is bridged by the similarities in building outlines present in both data sources. A KLD-sampling based particle filter is used to improve the efficiency and robustness of localization. We have shown quantitative and qualitative results for global localization performance in challenging scenarios, along with high runtime efficiency, even on embedded platforms.

Chapter 3

Range Sensing-based Place Recognition for Long-term Localization: An Evaluation Study

Place recognition is a critical capability for autonomous vehicles. It matches current sensor data with a pre-built database to provide coarse localization results. However, the effectiveness of long-term place recognition may be degraded by environment changes, such as seasonal or weather changes. To have a deep understanding of this issue, we conduct a comprehensive evaluation study on several state-of-the-art range sensing-based (i.e., LiDAR and radar) place recognition methods on the Borease dataset, which encapsulates long-term localization scenarios with stark seasonal variations and adverse weather conditions. In addition, we design a novel metric to evaluate the influence of matching thresholds on place recognition performance for long-term localization. Our results and findings provide fresh insights to the community and potential directions for future study.

3.1 Introduction

Place recognition (PR) refers to determining whether the current place has been visited previously against a pre-built keyframe-based database [5]. Such capability is critical for applications in mobile robots and autonomous vehicles, such as global localization and loop closure in Simultaneous Localization and Mapping (SLAM) [51, 52, 12, 53]. Significant progress has been made in the last two decades for both vision-based and range sensing-based methods [54, 55, 6]. However, reliable long-term PR still remains a challenge in complex road environments, where long-term variations frequently occur in both geometry and visual appearance.

Vision-based methods could be easily degraded due to dramatic changes in viewpoint, illumination, or weather conditions [56, 57]. Recently, range sensors, such as LiDAR and Frequency-Modulated Continuous Wave (FMCW) Radar have shown reasonable robustness to diverse weather conditions. Some methods have used them in SLAM and localization tasks under diverse weather conditions [58, 59, 60, 61, 62, 63, 64]. Radars are more robust to extreme weather conditions (e.g., rain or snow) than LiDARs. This is because radars work in GHz, which is lower than that of LiDARs (THz).

However, *to what extent do weather conditions influence radar- and LiDAR-based PR methods?* We find that this question has scarcely been investigated in existing literature, so we attempt to answer it by comparing state-of-the-art (SOTA) range sensing-based methods in this study. In addition to experiments, we also design a novel evaluation metric to assess influences caused by the matching thresholds [55] in long-term PR. The motivation for us to design the new metric is that existing works usually use the retrieval precision and recall [65], however, in long-term PR, the performance might be degraded when using the same matching threshold to determine whether a place has been visited. For example, a threshold that can achieve high precision and recall in summer might not provide satisfactory performance in winter.

We argue that a robust long-term PR method should be able to achieve performance with acceptable variations by using a general threshold under seasonal changes.

To the best of our knowledge, this is the first comprehensive evaluation that explores the impact of seasonal and weather variations on range sensing-based place recognition methods in long-term scenarios, considering both performance metrics and matching thresholds. Our contributions are as follows:

1. We design a novel metric to evaluate the influences of matching thresholds on long-term place recognition performance.

2. We conduct a comprehensive evaluation of SOTA range sensing-based place recognition methods on a dataset with long-term localization scenarios to explore the impact of season and weather conditions.

3. We open-source our evaluation code and make the experimental results publicly available, which could inspire further works in this area from the research community[1].

3.2 Related Work

This evaluation study mainly focuses on single-shot PR methods using 360° LiDARs and radars. For filtering- or aggregating-based approaches, as well as PR methods using Non-repetitive LiDAR, readers may refer to the survey papers [5, 6].

3.2.1 Range Sensing-based Place Recognition

LiDAR-based PR methods usually design global descriptors to compare the similarity between different LiDAR scans to retrieve places. They can be generally divided into

[1] `https://github.com/Weixin-Ma/PR_Evaluation_Project`

handcrafted feature-based methods and data-driven methods. Early works usually rely on handcrafted local features, such as mean surface curvature [66] and local Normal Distribution Transform (NDT) [67]. Differently, some researchers use semantic objects to build their global descriptors instead of using low-level geometric information. Fan *et al.* [68] and Zhu *et al.* [69] both used topological information of objects in environments to build global descriptors. Instead of building global descriptors on the extracted features, projection-based methods generate global descriptors based on the projection results of raw point clouds. M2DP [70] projects raw point cloud into multiple 2D planes to extract Histogram descriptors. Scan Context [71] is a typical bird-eye-view (BEV) projection-based method, where a 2D matrix descriptor is used to embed the geometirc information. Similarly, Scan Context++ [72], SSC[73], Intensity Scan Context [74], DiSCO[75], RING++[76], and LiDAR-Iris[77] all project point clouds into BEV, followed by different global descriptor extraction methods. Instead of using handcrafted features, data-driven methods extract features from point clouds using deep neural networks. Uy *et al.* proposed PointNetVLAD [78], which uses NetVLAD [79] to aggregate features extracted from PointNet [80] into a global descriptor. MinkLoc3D [81] and its extension [82] use generalized-mean pooling layer [83] to aggregate local features extracted from sparse voxelized point could into global descriptors.

Radar-based PR remains a challenge due to its low spatial resolution and noise. Gadd *et al.* [84] used sequence matching to reduce influences of noise clusters in a single radar scan, achieving a 30% boost in performance. The authors further introduced a temporal data augmentation method to obtain a more robust descriptor [85]. Suaftescu *et al.* [86] combined cylindrical convolutions, anti-aliasing blurring, and azimuth-wise max-pooling to extract more reliable features from polar radar scans. Different from all the data-driven methods above, Hong *et al.* [60] used M2DP [70] to extract global descriptors from filtered 2D radar point clouds.

For long-term range sensing-based PR, there are few works. Alijani *et al.* [87] eval-

uated a SOTA visual PR method GEM [83] on the Oxford RobotCar dataset [88]. Their results show a performance degradation of approximately 6% every 100 days. Peltomaki *et al.* [89] fed LiDAR depth images into an image retrieval method CNNRetr [90] to assess the performance of long-term LiDAR PR. Instead of using depth images, Zywanowski *et al.* [91] combined camera images and LiDAR intensity images, which benefits the PR performance across weather conditions. However, only one data-driven-based method and one metric are evaluated in [89, 91]. Cao *et al.* [92] developed a global descriptor from a cylindrical image representation of a 3-D point cloud, which enhances robustness with a sequence-based check. They later proposed a two-head classification network for end-to-end long-term localization [93]. However, all these methods [92, 93] require a sequence of LiDAR scans and odometry to build a submap.

3.2.2 Performance Evaluation for Place Recognition

The evaluation of PR methods typically focuses on their place retrieval performance. Machine learning metrics for classification tasks are widely adopted in PR. Popular metrics include Precision-recall curves [71], maximum F_1 score [73], Recall@100% [78], Extended Precision (EP) [65], AUC-PR [75], and Recall@N [55]. Given similarity values between every query frame and their retrieved frame, different values of Precision and Recall can be computed by varying the matching threshold. The Precision-recall curve can be obtained by plotting Precision against the Recall, which summarizes the trade-off between the true positive rate and the positive predictive value using different matching thresholds. Maximum F_1 score, EP, and AUC-PR are all computed from the Precision-recall curve, indicating the performance of a PR method with a single value between 0 and 1. Recall@100% represents the Recall value at which Precision drops from 100%, which shows the highest Recall that can be reached before the first false positive occurs. Recall@N is computed by dividing the number of query frames with correct matches among the top-N retrieved frames

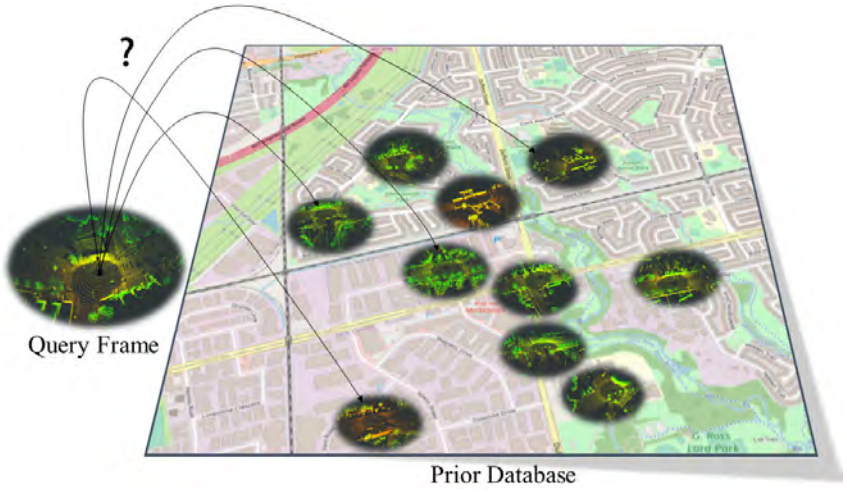


Figure 3.1: Problem formulation of place recognition. Given a query frame from the current sensor data, place recognition methods determine whether it has been previously visited by matching it with a pre-built database.

by the total number of query frames.

3.2.3 Research Gap

Existing literature scarcely investigate and evaluate influences of seasonal changes on the range sensing-based (i.e., both LiDAR and radar) PR methods. Meanwhile, existing evaluation metrics almost focus on the performance from the perspective of Precision and Recall. Influences of matching thresholds on the performance in long-term conditions have also not been studied.

3.3 Preliminaries

3.3.1 Problem Formulation of Place Recognition

Given a query LiDAR frame Q and a prior database D_{ref} as shown in Fig. 3.1, place recognition aims to determine whether the frame Q has been visited previously in D_{ref}

or not. This is determined by the similarity between Q and its most similar frame in D_{ref}. Once the similarity exceeds a predefined matching threshold, the frame Q is determined as a positive match, indicating its corresponding place has been visited previously in D_{ref}. Otherwise, the PR method considers the frame Q as a negative match.

3.3.2 Precision, Recall, and F-score

In PR, positive matches are fewer than negative matches. Precision-recall curve has been widely used to evaluate this imbalanced matching problem. Based on the matching results and the ground-truth information, correct positive matches are regarded as True-Positives (TP) whereas incorrect positive matches are regarded as False-Positives (FP). Similarly, True-Negatives (TN) and False-Negatives (FN) represent correct negative matches and incorrect negative matches, respectively. Precision and Recall are computed by Precision $= \frac{\text{TP}}{\text{TP+FP}}$ and Recall $= \frac{\text{TP}}{\text{TP+FN}}$, respectively.

Precision is the ratio of correctly identified positive matches, while Recall is the ratio of TP to actual positives. By adjusting the matching threshold, we can compute corresponding precision and recall values. The threshold typically ranges from the lowest to the highest similarity (or distance). A Precision-recall curve, plotting precision against recall, illustrates the trade-off between them under different thresholds.

F-score, F_β, is another widely used evaluation metric for PR, especially F_1 score. F_β considers both precision and recall. It is calculated by:
$F_\beta = (1+\beta^2) \cdot \frac{\text{Precision} \cdot \text{Recall}}{(\beta^2 \cdot \text{Precision}) + \text{Recall}}$, where $\beta \in R^+$ is chosen to make Recall β times as important as Precision. When $\beta = 1$, we have F_1 score, which is the harmonic mean of Precision and Recall.

3.4 The Proposed Evaluation Metric

The Precision-recall curve retains no information about the matching threshold. Other common evaluation metrics, like maximum F_1 score, EP, and AUC-PR, all summarize a Precision-recall curve into a single value between 0 and 1 to indicate the performance of a PR method. Therefore, the matching threshold information is missed. This information is important for long-term PR since a reliable long-term PR method is expected to achieve similar performance with a general matching threshold.

To assess such ability of a PR method, an intuitive idea is to keep the matching threshold unchanged and measure the performance variations for a PR method. For example, given a PR method, we can set a constant matching threshold and calculate metrics like Precision, Recall, or F-score for different sequences. The variations of the values of each metric can show the performance variations of a PR method. However, only the performance variation at a single point (i.e., the matching threshold) is evaluated. Meanwhile, it is not easy to select a specific value for the matching threshold. First, a higher/lower matching threshold will result in lower/higher Recall and higher/lower Precision across all the testing sequences, creating an illusion that the matching threshold has little influence on the performance variations. Second, the range of similarity values of the query results for different PR methods might be very different even using the same testing sequence. So it may be very difficult to find a specific threshold to fairly compare the influences of the matching threshold on the performance of different PR methods.

To solve these problems, we use the statistical result of the performance variations of a PR method instead of the performance variations at a single point. Considering the performance metrics, we choose to use the F-score since it provides a more comprehensive evaluation of a PR method by considering both Precision and Recall. Note that it is also acceptable to use other metrics like Precision and Recall here. Specifically, we introduce the metric $AwC\text{-}FT$, Area within Curves for FT-curves. For a given

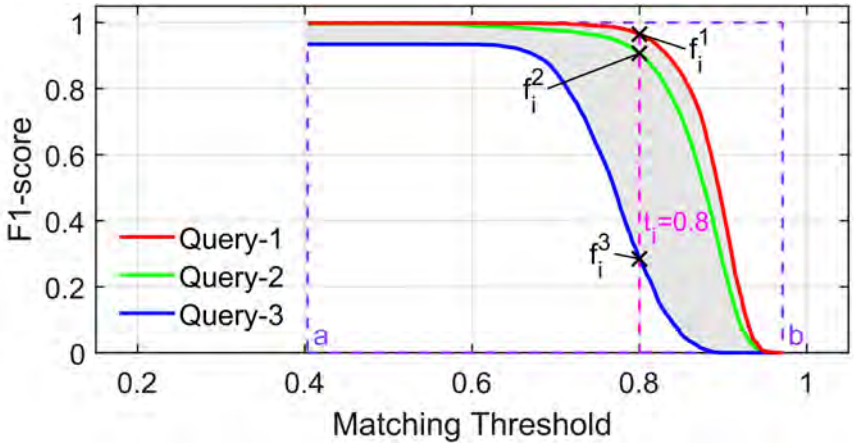


Figure 3.2: An example for how to compute the proposed *AwC-FT* (i.e., the area of gray area). The normalized *AwC-FT*, $\overline{AwC\text{-}FT}$ is the ratio of the original *AwC-FT* to the area of the purple dashed box.

reference sequence *Seq-j* and a query sequence *Seq-k*, where $k \in \{1, 2, ..., M\}$ is the index of the query sequence, a F-score Threshold curve (FT-curve) can be obtained by plotting F-score values against the matching thresholds. This is illustrated by the red, green, or blue curves in Fig. 3.2. In long-term PR, the reference sequence and the query sequence are usually different in terms of collection dates and environmental conditions, i.e., $j \neq k$. We denote $\{FT^k\}_j$ as the set of FT-curves that use different query sequences *Seq-k* with the same reference sequence *Seq-j*. *AwC-FT* can be computed based on the set $\{FT^k\}_j$:

$$\begin{aligned} AwC\text{-}FT &\approx \sum_{i=1}^{N-1} \frac{\Delta_i + \Delta_{i+1}}{2} \times (t_{i+1} - t_i)\,, \\ \Delta_i &= \mathrm{Max}\,(S_i) - \mathrm{Min}\,(S_i)\,,\ S_i = \left\{f_i^k\right\}_j\,, \end{aligned} \tag{3.1}$$

where N is the number of matching thresholds. t_i is the *i-th* matching threshold. We use matching similarity (i.e., probability) instead of descriptor distance to determine the value for t_i, which can ensure the ideal maximum value and minimum value for t_i are 1 and 0, respectively. $\{f_i^k\}_j$ is the set of F-score values from the set $\{FT^k\}_j$ when matching threshold equals to t_i. Max$(\cdot)$ and Min$(\cdot)$ are respectively the maximum and minimum values of the given set. Δ_i is the maximum difference of the F-score

values when the matching threshold equals t_i. An example for *AwC-FT* ($\beta = 1$) calculation is shown in Fig. 3.2. Three different sequences (i.e., the red, green, and blue curve) are used as query sequences to form RT-curves $\{FT^k\}_j$. $\{f_i^1, f_i^2, f_i^3\}$ are F_1 score values from the set $\{FT^k\}_j$ when $t_i = 0.8$. We have $\Delta_i = f_i^1 - f_i^3$. The value of *AwC-FT* equals to the area of the gray area shown in Fig. 3.2.

To simplify the calculation, we make the matching threshold t_i as a discrete uniform distribution $U\{a, b\}$. Let $s_l^k \in [0, 1]$ be the similarity between *l-th* frame of query sequence *Seq-k* and the retrieved frame from the given reference sequence, then $\{s_l^k\}_j$ is the set of matching similarities for all frames from query sequence *Seq-k* when using *Seq-j* as the reference. Values of a and b can be calculated as following:

$$a = \text{Min}\left(\text{Min}\left(\left\{s_l^1\right\}_j\right), \ldots, \text{Min}\left(\left\{s_l^M\right\}_j\right)\right), \tag{3.2}$$

$$b = \text{Max}\left(\text{Max}\left(\left\{s_l^1\right\}_j\right), \ldots, \text{Max}\left(\left\{s_l^M\right\}_j\right)\right), \tag{3.3}$$

Therefore, $t_{i+1} - t_i$ can be simplified as to $\frac{b-a}{N}$. Eq. 3.1 can be rewritten as:

$$AwC\text{-}FT \approx \frac{b-a}{N} \times \left(\frac{\Delta_1}{2} + \sum_{i=2}^{N-1} \Delta_i + \frac{\Delta_N}{2}\right), \tag{3.4}$$

The calculated *AwC-FT* is then normalized by dividing $((b - a) \times (1 - 0))$. The normalized *AwC-FT* is defined as $\overline{AwC\text{-}FT} \in [0, 1]$, representing the ratio of the original *AwC-FT* and the area of the purple dashed box, as shown in Fig. 3.2.

$$\overline{AwC\text{-}FT} \approx \frac{1}{N} \times \left(\frac{\Delta_1}{2} + \sum_{i=2}^{N-1} \Delta_i + \frac{\Delta_N}{2}\right). \tag{3.5}$$

The $\overline{AwC\text{-}FT}$ metric approximately represents the average performance variations of a PR method when conducting place recognition using different query sequences (i.e., query sequences collected on different date and weather conditions) and the same reference sequence, with the same matching thresholds. Theoretically, a larger value of $\overline{AwC\text{-}FT}$ indicates that the performance of the PR method is more sensitive to the matching thresholds under seasonal changes in long-term scenarios. Since $\overline{AwC\text{-}FT}$

only represents the performance variation, it is better to use the metric and the other metrics (e.g., Maximum F_1 score) at the same time for more comprehensive evaluation results. In this paper, we use the F_1 score for $\overline{AwC\text{-}FT}$. Unless specifically stated, otherwise all experimental results related to $\overline{AwC\text{-}FT}$ are calculated based on the F_1 score.

3.5 The Evaluations

3.5.1 Dataset and Experimental Setup

KITTI [94], Oxford Radar [95], and MulRan [96] are urban environment datasets that have been widely used in range sensing-based PR. However, none of these datasets contain both season and weather variations, since their collection dates span less than 3 months. Alternatively, we use Boreas Dataset [97], which includes more than 350 km of data collected by driving a repeated route over one year. Seasonal variations and adverse weather conditions, such as rain and snowstorms, can be found in the dataset. As for sensor configurations, the data-collection vehicle has a camera, a 360° radar, a 128-beam LiDAR, and GPS/IMU. Similar to [59], we select 6 sequences with stark weather variations for evaluation. Sample weather variations can be found in Fig. 3.3. We down-sample LiDAR frames to the scan frequency of the 360° radar (i.e., 4Hz). The down-sampled sequences are further filtered to keep the distance between two consecutive frames not less than 1m. Details of the evaluation sequences can be found in Tab. 3.1.

We use several SOTA open-sourced PR methods: Scan Context [71], LiDAR-Iris [77], MinkLoc3Dv2 [98], and OverlapTransformer [99]. Scan Context and LiDAR-Iris are handcrafted feature-based methods. They both extract descriptors from the projection results of point clouds, while LiDAR-Iris compares the similarity between two LiDAR frames in the frequency domain. MinkLoc3Dv2 and OverlapTransformer

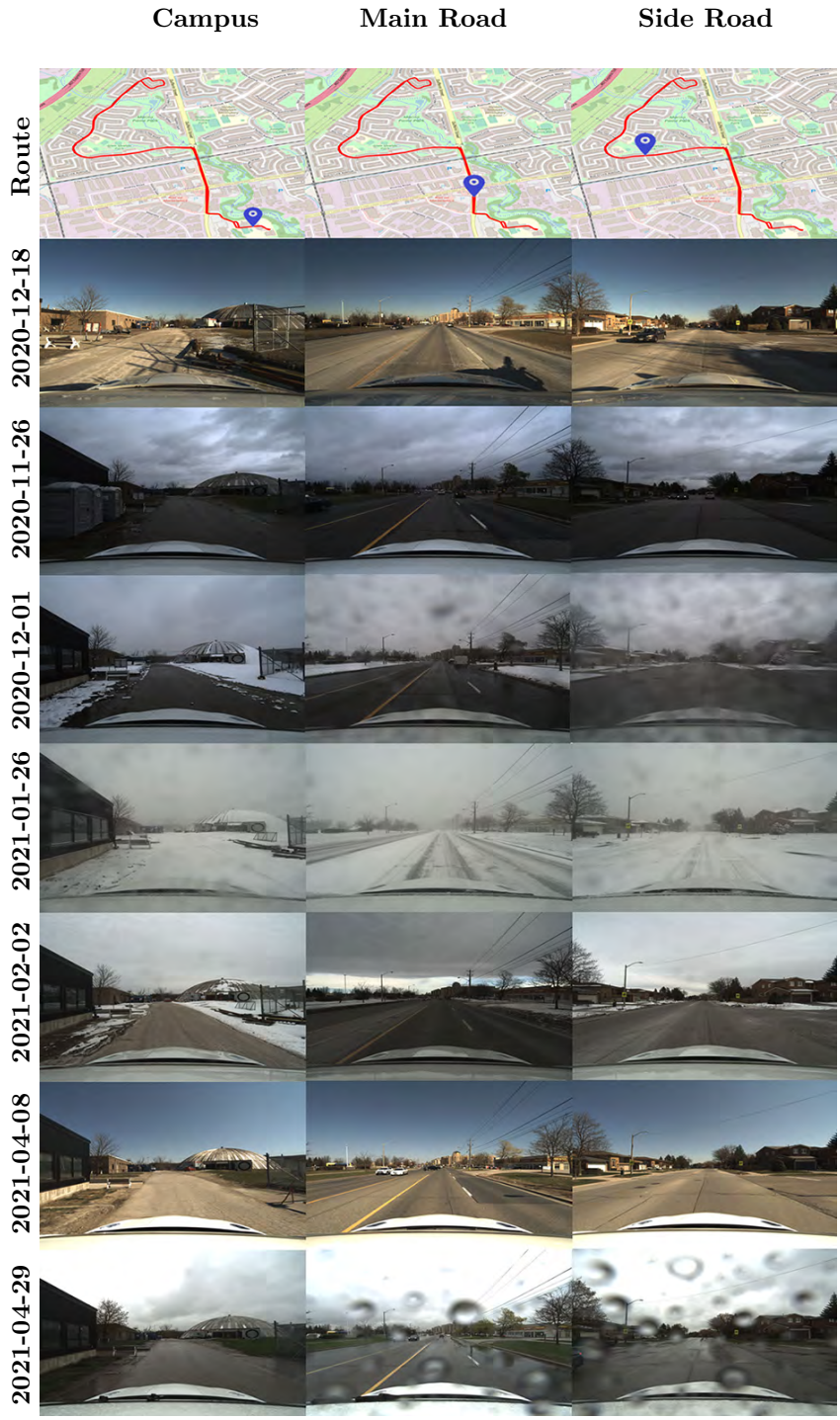


Figure 3.3: Examples of seasonal variations across sequences from Boreas Dataset. All the sequences are collected along the same route on different dates. Pictures in each column are captured in the same place. Sequence 2020-12-18 is used to fine tune the data-driven place recognition methods. All the other sequences are used for evaluation.

Table 3.1: Details of the evaluation sequences. Seq-ID refers to the collection date. Notation in a blanket is used to represent the corresponding sequence for simplicity. The frame number here is the number after down-sampling and filtering.

Seq-ID	Weather Condition	Frame Number
2020-11-26 (Seq-01)	overcast, snow	3305
2020-12-01 (Seq-02)	overcast, snow, snowing	3291
2021-01-26 (Seq-03)	overcast, snow, snowing (heavy)	4047
2021-02-02 (Seq-04)	overcast, snow (severe)	3324
2021-04-08 (Seq-05)	sun	3104
2021-04-29 (Seq-06)	overcast, rain	3181

are data-driven methods, while their loss functions are based on localization and overlap, respectively. We use the default parameters for Scan Context and LiDAR-Iris provided by the authors. Following the existing long-term PR works [89, 87], we fine tune MinkLoc3Dv2 and OverlapTransformer on another sequence 2020-12-18 which is different from all the other evaluation sequences.

During evaluation, we alternately use each sequence as the reference sequence, and the remaining five sequences as the query sequence. For example, if *Seq*-01 is the reference, *Seq*-02 to 06 are queries. Alternatively, if *Seq*-02 is the reference, *Seq*-01, 03, 04, 05, and 06 are queries. This allows testing of long-term PR performance under various seasonal conditions, like recognizing places on rainy days against a snowy day database, or vice versa. Specifically, we denote a sequence pair as $\langle k, j \rangle$, which contains a query sequence *Seq-k* and a reference sequence *Seq-j*. $\{\langle k, j \rangle\}_j$ is the set of sequence pairs that have the same reference sequence *Seq-j*. Here we have $j, k \in \{01, 02, 03, 04, 05, 06\}$ *and* $j \neq k$, representing the 6 evaluation sequences by the order of collection date. Following [87] and [89], we conduct all experiments using the top-1 retrieval results, i.e., only the retrieved frame with the highest similarity is used. In accordance with prior research [71] and [77], a matching detection is classified as

a true positive if the ground-truth pose distance between the query frame and the matched frame is less than 4m.

3.5.2 Evaluation with Widely-used Metrics

For a given PR method and $\langle k, j \rangle$, we evaluate its performance with several widely-used metrics: maximum F_1 score, EP, and Recall@1. The experimental results are displayed in Tab. 3.2. We also calculate the average values and standard deviations of the above metrics for each $\{\langle k, j \rangle\}_j$ to show the variations of PR performance.

LiDAR-Iris achieves the best performance in terms of all the metrics under all the seasonal conditions. The average values of the maximum F_1 score, EP, and Recall@1 are all higher than 99.5% with a small standard deviation (less than 0.5%) except the average of EP when using sequence *Seq*-03 as the reference (i.e., 97.83%). Scan Context also shows a competitive performance. There is only a minor decline observed across all the evaluation metrics. We can find prominent degradation for both OverlapTransformer and MinKLoc3Dv2.

According to Recall@1 values, there are respectively around 60% and 50% of the matching results that are correct for OverlapTransformer and MinkLock3Dv2, while more than 96% and 97% of the matching results are positive for Scan Context and LiDAR-Iris under all seasonal conditions. EP provides a good summary on both P_{R0} (i.e., the Precision at the minimum Recall value) and R_{P100} (i.e., the Recall value where the Precision drops from 100%) [65]. When EP is less than 0.5, P_{R0} is less than 1.0, meaning there exist false positives even using the highest matching threshold and the PR method can never provide a matching result at 100% Precision. We can see that the averages of EP for both MickLoc3Dv2 and OverlapTransformer in almost $\{\langle k, j \rangle\}_j$ are less than 0.5, while the average values of EP for both Scan Context and LiDAR-Iris are both larger than 0.94. This shows that Scan Context and LiDAR-Iris can reach a higher Recall without any false positives.

Table 3.2: PR performance for different $\{\langle k, j\rangle\}_j$. The best result for each evaluation metric is emphasized with different formats (i.e., bold face for F_1, wavy line for EP, and underline for R@1). Arrows with different directions indicate different value ranges. ↓ refers to [0, 20]. ↘ refers to [20, 40]. → refers to [40, 60]. ↗ refers to [60, 80]. ↑ refers to [80, 100].

Ref	Que	SC			Iris			OT			Mink		
		F_1	EP	R@1	F_1	EP	R@1	F_1	EP	R@1	F_1	EP	R@1
Seq-01	Seq-02	99.68	96.45	99.36	99.91	99.66	99.64	85.91	51.05	75.30	62.96	51.20	45.94
	Seq-03	93.46	82.42	87.72	100.00	100.00	100.00	27.53	50.02	15.96	1.23	0.00	0.62
	Seq-04	99.95	99.83	99.91	99.98	99.98	99.94	86.49	53.64	76.20	88.10	56.35	78.73
	Seq-05	99.98	99.81	99.97	99.98	99.98	99.97	86.30	51.80	75.90	86.49	56.68	76.19
	Seq-06	99.89	99.73	99.78	99.87	99.69	99.75	80.32	51.51	67.12	84.73	54.92	73.50
	ave	98.59↑	95.65↑	97.35↑	**99.95**↑	99.86↑	99.86↑	73.31↗	51.61→	62.10↗	64.70↗	43.83→	55.00→
	std	2.57	6.74	4.82	0.05	0.15	0.14	23.01	1.18	23.31	33.03	22.00	29.64
Seq-02	Seq-01	99.62	96.14	99.24	99.91	99.29	99.73	86.31	52.26	75.92	62.86	51.09	45.84
	Seq-03	97.91	90.97	95.90	99.99	99.94	99.75	31.30	0.00	18.56	1.37	0.00	0.69
	Seq-04	99.44	95.66	98.89	99.95	99.50	99.82	85.56	50.95	74.76	60.42	51.21	43.29
	Seq-05	99.29	97.92	99.72	99.95	98.76	99.90	83.59	50.40	71.81	68.61	50.48	52.22
	Seq-06	99.49	96.19	98.99	99.95	99.94	99.81	80.03	51.26	66.71	65.98	51.16	49.23
	ave	99.15↑	95.38↑	98.55↑	**99.95**↑	99.49↑	99.80↑	73.36↗	40.97→	61.55↗	51.85→	40.79→	38.26↘
	std	0.63	2.33	1.36	0.02	0.44	0.06	21.14	20.50	21.73	25.39	20.40	19.02
Seq-03	Seq-01	98.36	88.46	96.67	99.91	99.77	98.03	36.22	50.03	22.12	2.39	0.00	1.21
	Seq-02	98.82	89.75	97.11	99.95	99.95	97.54	41.05	0.00	25.83	2.99	0.00	1.52
	Seq-04	98.62	83.82	96.84	99.86	99.83	97.92	40.88	50.02	25.69	1.73	0.00	0.87
	Seq-05	98.04	89.50	96.04	99.77	99.74	97.58	34.54	50.02	20.88	2.10	0.00	1.06
	Seq-06	98.27	84.52	96.32	99.84	99.63	98.05	31.11	0.00	18.42	1.75	0.00	0.88
	ave	98.42↑	87.21↑	96.60↑	**99.87**↑	99.78↑	97.83↑	36.76↘	30.01↘	22.59↘	2.19↓	0.00↓	1.11↓
	std	0.27	2.53	0.38	0.06	0.11	0.22	3.81	24.51	2.85	0.47	0.00	0.24
Seq-04	Seq-01	99.91	99.55	99.82	100.00	100.00	99.85	85.84	51.23	75.19	88.17	54.17	78.85
	Seq-02	99.50	96.89	99.00	99.92	99.82	99.73	85.21	50.98	74.23	59.51	51.17	42.36
	Seq-03	93.35	82.59	87.52	99.98	99.90	99.83	28.23	50.01	16.43	0.84	0.00	0.42
	Seq-05	99.98	99.32	99.98	99.98	99.73	99.97	86.36	51.08	76.00	85.63	54.24	74.87
	Seq-06	99.94	99.87	99.87	99.97	99.84	99.94	80.10	50.30	66.80	83.03	52.00	70.98
	ave	98.53↑	95.64↑	97.24↑	**99.97**↑	99.86↑	99.86↑	73.15↗	50.72→	61.73↗	63.44↗	42.31→	53.50→
	std	2.60	6.61	4.87	0.03	0.09	0.09	22.57	0.48	22.89	32.93	21.19	29.48
Seq-05	Seq-01	99.94	99.94	99.88	99.98	99.88	99.97	85.98	51.47	75.40	86.39	56.35	76.04
	Seq-02	99.47	95.96	98.94	99.97	99.85	99.94	82.28	50.36	69.89	67.11	50.44	50.50
	Seq-03	91.43	79.75	84.21	99.88	99.70	99.75	25.49	50.01	14.60	0.64	0.00	0.32
	Seq-04	99.94	99.68	99.88	100.00	100.00	100.00	86.82	50.77	76.71	84.72	52.90	73.50
	Seq-06	99.94	99.34	99.87	99.98	99.98	99.98	81.59	50.68	68.91	84.95	53.13	73.84
	ave	98.14↑	94.93↑	96.56↑	**99.96**↑	99.88↑	99.93↑	72.43↗	50.66→	61.10↗	64.76↗	42.57→	54.84→
	std	3.36	7.73	6.18	0.04	0.11	0.09	23.56	0.48	23.45	32.84	21.37	28.81
Seq-06	Seq-01	99.82	99.06	99.61	99.92	99.48	99.73	79.95	50.86	66.60	84.54	54.83	73.22
	Seq-02	99.59	96.57	99.18	99.97	99.94	99.70	81.30	50.62	68.49	64.54	51.10	47.65
	Seq-03	92.30	76.38	85.69	99.90	99.76	99.70	23.47	0.00	13.29	1.08	0.00	0.54
	Seq-04	99.95	99.88	99.91	99.98	99.82	99.97	81.98	50.32	69.46	81.83	51.84	69.25
	Seq-05	99.92	99.47	99.84	99.94	99.39	99.87	83.13	50.73	71.13	85.10	56.29	74.07
	ave	98.32↑	94.27↑	96.85↑	**99.94**↑	99.68↑	99.79↑	69.97↗	40.51→	57.80→	63.42↗	42.81→	52.95→
	std	3.01	9.02	5.58	0.03	0.21	0.11	23.27	20.25	22.30	32.07	21.49	27.92

Intuitively, when the seasonal and weather conditions change between query and reference sequences, the performance of a PR method should vary. Interestingly, such performance variations are almost negligible for LiDAR-Iris. For instance, when using a snowstorm sequence *Seq*-03 as the reference, there was only a slight decrease in Recall@1 compared to using the other sequences as references. Regarding Scan Context, there is an average performance degradation of about 8% in EP when using *Seq*-03 as the reference. Meanwhile, unlike LiDAR-Iris, the values of average EP and Recall@1 for Scan Context both decrease when using the snowstorm *Seq*-03 as query sequence. Similar trends of performance variations can also be found in the results for OverlapTransformer and MinkLoc3Dv2. However, such degradation is more dramatic than that for Scan Context and LiDAR-Iris. For example, for OverlapTransformer, the maximum F_1 score and Recall@1 decrease by about 40% on average when using *Seq*-03 as a reference sequence. Such degradation is even more prominent for MinkLoc3Dv2, which is more than 50%.

Our results show that the SOTA handcrafted feature-based LiDAR methods can be robust to seasonal variations in long-term PR. Two SOTA data-driven methods (i.e., OverlapTransformer and MinkLoc3Dv2) demonstrate a heightened sensitivity to seasonal variations. We guess the main reason is that these two networks were initially designed and trained using other datasets that vary from Boreas Dataset in terms of both scene layouts and sensor configuration (i.e., resolution and installation position). Limited by the network generalizability, the overall performance degrades than those reported in [98] and [99]. In addition, insufficient training data collected under different weather conditions may result in a significant performance variation during evaluations.

Table 3.3: Results of $\overline{AwC\text{-}FT}$ for different $\{\langle k, j\rangle\}_j$. The best result for each $\{\langle k, j\rangle\}_j$ (i.e., each row) is highlighted in bold font. "Ref" is short for reference.

Ref	Input: LiDAR				Input: Radar	
	SC	Iris	OT	Mink	SC	Iris
Seq-01	25.54	23.06	56.79	85.78	**4.78**	5.84
Seq-02	13.72	17.77	52.78	66.16	**3.32**	5.32
Seq-03	8.49	3.29	9.17	**0.90**	4.65	4.49
Seq-04	25.94	21.33	55.53	67.88	**3.45**	3.86
Seq-05	25.98	20.90	58.32	84.60	**3.14**	4.42
Seq-06	25.31	19.48	55.31	82.89	**5.10**	7.66
ave	20.83	17.64	47.98	64.70	**4.07**	5.26

3.5.3 Influences by Matching Thresholds

In the last part, we evaluate the performance of several SOTA LiDAR PR methods by using three widely-used metrics. Such evaluation provides a general statement about how good the performance can be. In this section, we evaluate a long-term PR method from a new perspective by using the proposed metric. For every $\{\langle k, j\rangle\}_j$, we compute its $\overline{AwC\text{-}FT}$, as shown in Tab. 3.3. The results are also visualized in Fig. 3.4.

We summarize the main findings as follows: Firstly, using the same matching thresholds, the average F_1 score variations for handcrafted feature-based methods is smaller than those for data-driven methods. As shown in Tab. 3.3, the average $\overline{AwC\text{-}FT}$ values for LiDAR-Iris and Scan Context are much smaller than those for Overlap-Transformer and MinkLoc3Dv2. Such variations can be also observed in Fig. 3.4. Secondly, we interestingly find that $\overline{AwC\text{-}FT}$ decreases when the reference sequence *Seq-j* for $\{\langle k, j\rangle\}_j$ is significantly different from all the other query sequence *Seq-k*. Specifically, the snowstorm sequence *Seq*-03 differs from all the other sequences in

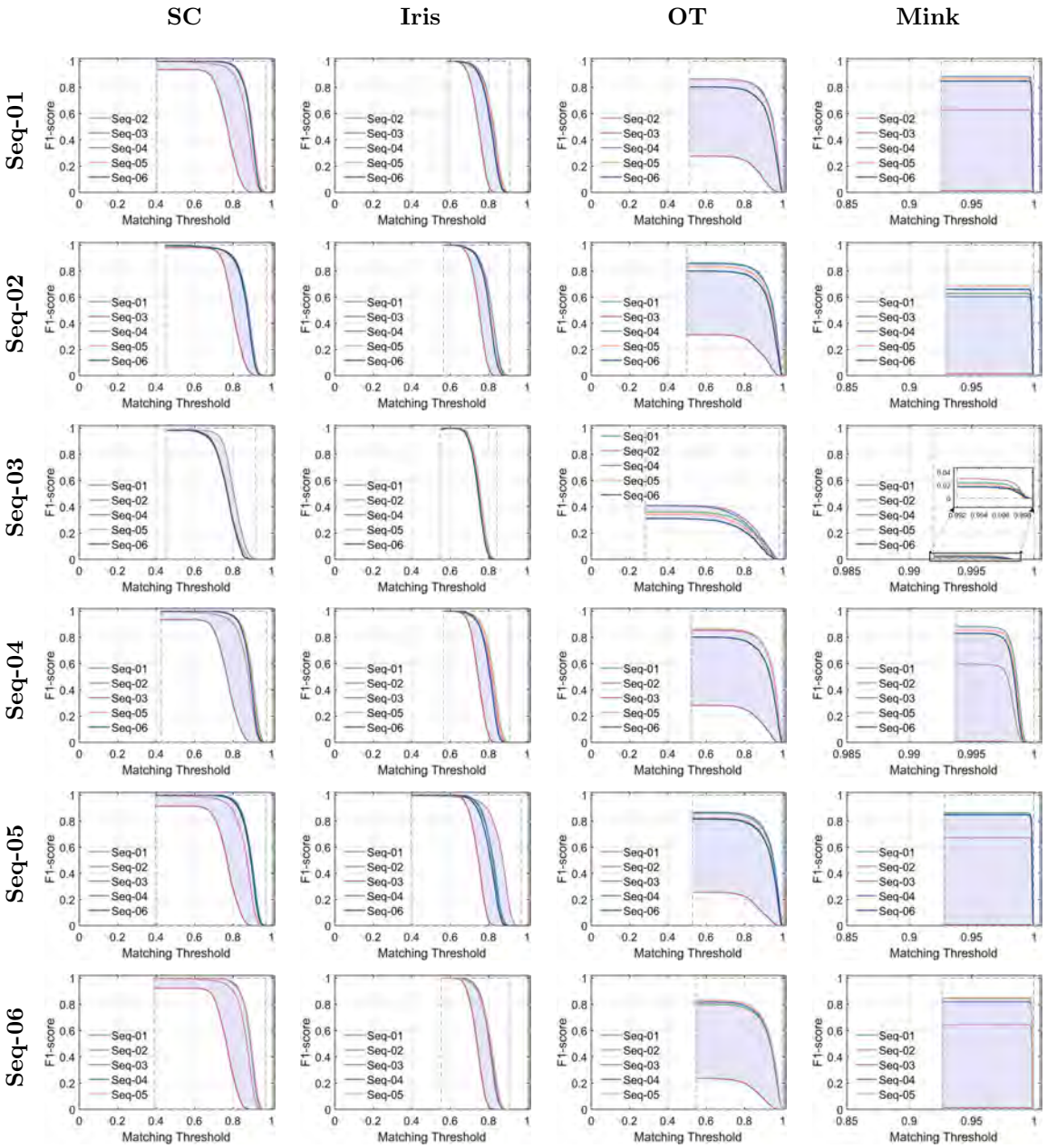


Figure 3.4: FT-curves $\{FT^k\}$ for different $\{\langle k, j\rangle\}_j$ using several SOAT LiDAR-based PR methods. Sequences ID on the left refers to the used reference sequence. Figures in each column are based on the same PR method.

terms of weather conditions. When using *Seq*-03 as the reference sequence, the variation in performance is much smaller than those results when using the other sequences as the reference, as shown in the third row in both Tab. 3.3 and Fig. 3.4.

Table 3.4: The frame number of each segmented region for different sequences.

Seq-ID	Frame Number			Total
	Region-01	Region-02	Region-03	
Seq-01	632	774	1899	3305
Seq-02	656	731	1904	3291
Seq-03	818	927	2302	4047
Seq-04	654	707	1963	3324
Seq-05	636	639	1829	3104
Seq-06	604	774	1803	3181

Considering practical applications, $\overline{AwC\text{-}FT}$ not only can evaluate the influence of matching thresholds on the performance of different PR methods in long-term localization but also can directly benefit the choice of reference sequences. For example, from Tab. 3.3, we can find that the performance variation is the smallest when using *Seq*-03 as a reference. However, the average values for maximum F_1 score, EP, and R@1 are smaller than those when using the other sequences as reference (see Tab. 3.2). In other words, using *Seq*-03 as the reference sequence leads to a poor yet stable result in long-term PR. In addition, the absolute performances for all the methods are very close when using *Seq*-01, 02, 04, 05, and 06 as references. It is reasonable to select any of them as the reference. Nevertheless, when using *Seq*-02 as the reference sequence, we find that the $\overline{AwC\text{-}FT}$ values for all the methods are the smallest. So, choosing *Seq*-02 as the reference achieves a robust and competitive long-term performance.

Overall, the experimental results show that the performance influenced by different matching thresholds vary across handcrafted feature-based and data-driven methods. The proposed metric can effectively evaluate such influence to show the robustness of the methods to the matching thresholds. Moreover, it can also benefit the choice of reference sequence.

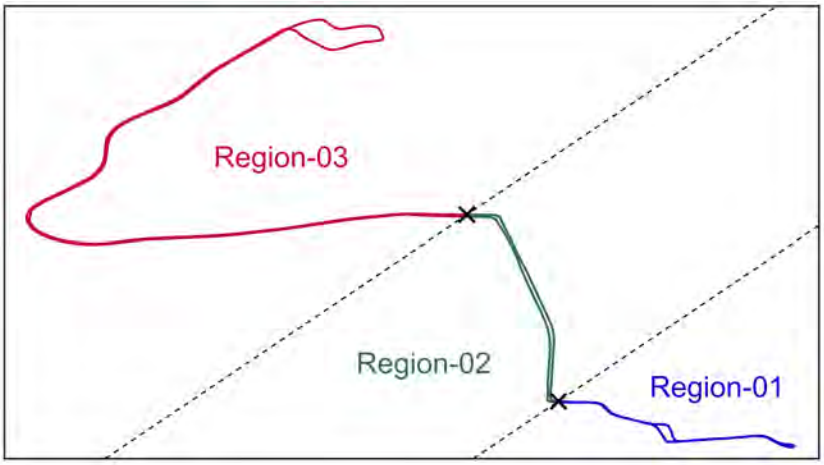


Figure 3.5: Example of the segmented regions along the route. ✕ refers to the locations used to segment each sequence into three different regions.

3.5.4 Matching Similarity Distributions

In large-scale environments, there may be significant variations in geometry across different regions, such as building styles and road layouts. These variations can directly influence the performance of PR methods. To assess influences of these variations on PR performance, we divide each query sequence into distinct regions based on ground-truth poses. Specifically, there are three different regions, denoted as Region-01 (campus region), Region-02 (main-road region), and Region-03 (side-road region), as shown in Fig. 3.5. Region-01 and 03 exhibit fewer dynamic objects, such as vehicles and pedestrians, whereas Region-02 is highly dynamic with many moving vehicles. Details about the frames of each region on different sequences can be found in Tab. 3.4.

For each sequence pair $\langle k, j \rangle$ from the set $\{\langle k, j \rangle\}_j$, we first compute the Recall@1 for each region, i.e., the ratio of positive matches within a specific region to the number of query frames within the same region. We then compute the average of Recall@1 for each region. Results are displayed in Tab. 3.5. Values in each row are computed using the same $\{\langle k, j \rangle\}_j$. To better present the differences in matching results in

Table 3.5: The average of Recall@1 (%) for different regions for different $\{\langle k, j\rangle\}_j$. Results in each row are computed using the same $\{\langle k, j\rangle\}_j$. The best result for each row is highlighted. Different formats are used to represent the best results from different regions (i.e., bold face for Region-01, wavy line for Region-02, and underline for Region-03).

Ref	Region-01				Region-02				Region-03			
	SC	Iris	OT	Mink	SC	Iris	OT	Mink	SC	Iris	OT	Mink
Seq-01	94.68↑	**99.84**↑	73.26↗	52.12→	93.90↑	99.52↑	34.55↘	49.83→	99.68↑	100.00↑	68.82↗	57.88→
Seq-02	98.28↑	**99.94**↑	75.34↗	41.10→	94.41↑	99.17↑	31.25↘	22.08↘	99.85↑	100.00↑	68.50↗	43.48→
Seq-03	85.66↑	**89.60**↑	40.26→	1.44↓	97.65↑	99.42↑	8.87↓	0.25↓	99.89↑	99.99↑	21.86↘	1.33↓
Seq-04	94.54↑	**99.67**↑	75.60↗	54.24→	93.50↑	99.67↑	34.56↘	47.88→	99.69↑	100.00↑	67.62↗	55.37→
Seq-05	93.71↑	**99.83**↑	76.44↗	54.66→	91.50↑	98.92↑	33.77↘	47.38→	99.59↑	100.00↑	66.73↗	57.79→
Seq-06	93.72↑	**99.60**↑	71.45↗	49.59→	93.64↑	99.47↑	33.94↘	50.08→	99.23↑	99.98↑	62.11↗	55.19→

different regions, we visualize some examples of the matching similarity distributions along the trajectory of query sequence as heat maps, as shown in Fig. 3.6.

In general, LiDAR-Iris demonstrates superior performance across all three regions, particularly in Region-03 where the average of Recall@1 approaches nearly 100% for all $\{\langle k, j\rangle\}_j$. As expected, for all the methods, the average Recall@1 of Region-02 is generally lower than those of the other two regions. This can be also supported by the matching similarity distributions in different regions as shown in Fig. 3.6. The heat map colors indicating the similarity of matching results in Region-02 are lighter than those in Region-01 and Region-03. This is particularly noticeable in the results for LiDAR-Iris (i.e., second row) and the radar-based Scan Context (i.e., fifth row). Compared to the handcrafted feature-based methods, the average Recall@1 for Region-02 significantly decreases when using the data-driven methods. This suggests, to some extent, that the influences of dynamic objects on Recall@1 is more pronounced for data-driven methods.

Based on our experimental results, the handcrafted feature-based methods are more robust to the variations in geometry across different regions in large-scale environ-

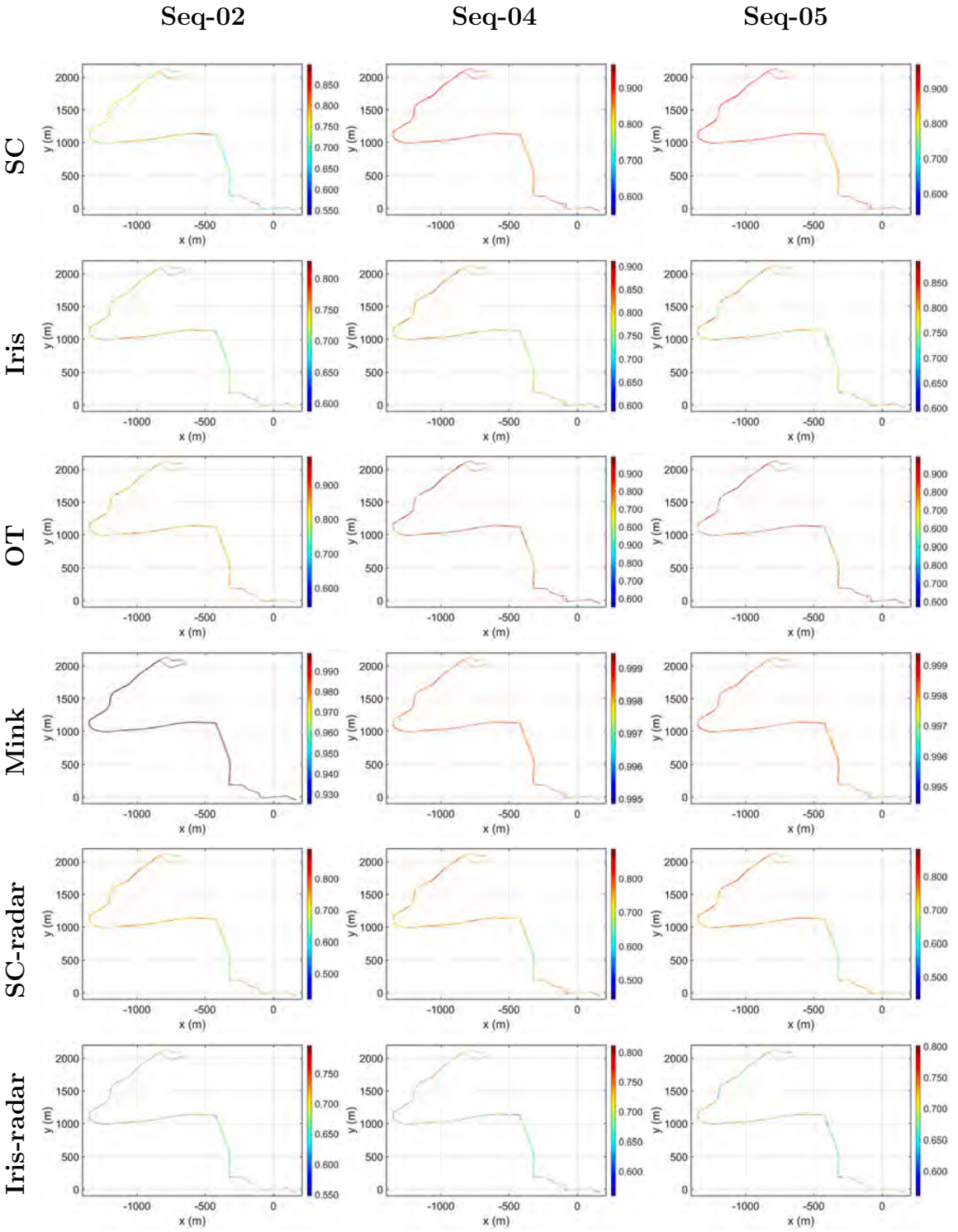


Figure 3.6: Examples of top-1 matching similarity distributions. All the results are based on the same reference sequence, Seq-01. The sequence ID on the top refers to the query sequence used in each column.

ments.

3.5.5 Comparisons with Radar-based Methods

Due to the robustness to diverse weather conditions, radar has recently gained a lot of attention, showing significant potential in long-term localization. However, radar suffers from multiple sources of artifacts and clutters, for example, speckle noise, receiver saturation, and multi-path reflections. So, we first use the filtering method [100] to reduce noise. The filtered points are then transferred from 2D radar images into 3D point clouds while the z-coordinate is set as 1 for all the filtered points. We then use the generated point clouds as inputs to run Scan Context and LiDAR-Iris. Results for maximum F_1 score, EP, and Recall@1 are displayed in Tab. 3.6.

Different from LiDAR-based methods, radar-based Scan Context outperforms the radar-based LiDAR-Iris. The radar-based Scan Context exhibits improvements in both maximum F_1 score and Recall@1 compared to the LiDAR-based Scan Context. The average values of maximum F_1 score and average values of Recall@1 for different $\{\langle k, j\rangle\}_j$ are all larger than 99.6% and 99.0%, respectively. The value of average EP increases in almost $\{\langle k, j\rangle\}_j$. Regarding radar-based LiDAR-Iris, the decrease of the maximum F_1 score and Recall@1 can be deemed insignificant. From the perspective of EP, there is about a 10% drop. We conjecture the reason could be the loss of height information in the radar-based point cloud, which is important for LiDAR-Iris to generate its global descriptor. Surprisingly, these two handcrafted feature-based LiDAR methods show good generalizability on radar sensors.

As expected, radar-based methods show small performance variations using the same matching thresholds in long-term scenarios. As shown in Tab. 3.3, the average $\overline{AwC\text{-}FT}$ values for radar-based Scan Context and radar-based LiDAR-Iris are 4.07% and 5.26%, respectively, which are greatly less than the other LiDAR-based methods. These results indicate that in radar-based methods, influences of matching thresholds

Table 3.6: PR performance for different $\{\langle k, j\rangle\}_j$ using radar-based methods. The best result for each evaluation metric is emphasized with different formats (i.e., bold face for F_1, wavy line for EP, and underline for R@1).

Ref	Que	SC-radar			Iris-radar		
		F_1	EP	R@1	F_1	EP	R@1
Seq-01	Seq-02	99.59	94.42	99.18	99.68	95.68	99.36
	Seq-03	99.80	99.23	98.00	99.80	98.35	97.97
	Seq-04	99.89	99.12	99.79	99.92	99.86	99.79
	Seq-05	99.91	92.84	99.82	99.95	99.30	99.91
	Seq-06	99.68	96.08	99.36	99.54	77.20	99.03
	ave	99.77↑	96.34↑	99.23↑	**99.78**↑	94.08↑	99.21↑
	std	0.12	2.53	0.66	0.15	8.56	0.69
Seq-02	Seq-01	99.76	93.69	99.51	99.63	80.38	99.27
	Seq-03	99.49	95.61	97.42	99.49	94.67	97.36
	Seq-04	99.65	94.86	99.30	99.68	95.81	99.33
	Seq-05	99.66	93.66	99.33	99.80	79.02	99.60
	Seq-06	99.73	97.23	99.45	99.71	98.67	99.42
	ave	**99.66**↑	95.01↑	99.00↑	**99.66**↑	89.71↑	99.00↑
	std	0.09	1.33	0.80	0.10	8.29	0.83
Seq-03	Seq-01	99.95	99.60	99.90	99.93	84.40	99.85
	Seq-02	99.79	96.93	99.58	99.79	74.17	99.58
	Seq-04	99.85	97.33	99.70	99.88	99.46	99.68
	Seq-05	99.80	94.53	99.60	99.84	98.80	99.68
	Seq-06	99.84	98.37	99.60	99.65	90.21	99.28
	ave	**99.85**↑	97.35↑	99.68↑	99.82↑	89.41↑	99.61↑
	std	0.06	1.69	0.12	0.09	9.46	0.19
Seq-04	Seq-01	99.92	82.24	99.85	99.95	83.12	99.91
	Seq-02	99.80	95.92	99.61	99.85	88.81	99.70
	Seq-03	99.82	97.02	97.89	99.79	99.52	97.86
	Seq-05	99.92	95.32	99.85	99.94	93.80	99.88
	Seq-06	99.86	91.58	99.73	99.83	95.05	99.67
	ave	**99.87**↑	92.42↑	99.39↑	**99.87**↑	92.06↑	99.40↑
	std	0.05	5.41	0.75	0.06	5.63	0.78
Seq-05	Seq-01	99.87	91.38	99.74	99.89	86.47	99.77
	Seq-02	99.56	94.41	99.13	99.73	81.12	99.45
	Seq-03	99.74	96.27	97.49	99.69	99.06	97.42
	Seq-04	99.90	99.02	99.81	99.89	99.68	99.77
	Seq-06	99.87	99.19	99.74	99.61	85.33	99.23
	ave	**99.79**↑	96.06↑	99.18↑	99.76↑	90.33↑	99.13↑
	std	0.13	2.94	0.88	0.11	7.59	0.88
Seq-06	Seq-01	99.76	94.26	99.53	99.70	77.85	99.40
	Seq-02	99.76	96.13	99.53	99.91	99.65	99.81
	Seq-03	99.81	98.61	97.89	99.68	97.61	97.80
	Seq-04	99.91	99.39	99.81	99.84	72.05	99.69
	Seq-05	99.92	92.94	99.84	99.75	77.03	99.50
	ave	**99.83**↑	96.26↑	99.32↑	99.78↑	84.84↑	99.24↑
	std	0.07	2.46	0.73	0.09	11.45	0.73

Table 3.7: The average of Recall@1 (%) for different regions for different $\{\langle k, j\rangle\}_j$ using different radar-based PR methods.

Reference	Region-01		Region-02		Region-03	
	SC-radar	Iris-radar	SC-radar	Iris-radar	SC-radar	Iris-radar
Seq-01	**99.77**↑	99.72↑	99.13↑	98.69↑	99.91↑	99.99↑
Seq-02	99.84↑	**99.91**↑	97.56↑	98.28↑	99.97↑	99.97↑
Seq-03	**89.50**↑	89.48↑	99.34↑	99.17↑	99.90↑	99.88↑
Seq-04	99.54↑	**99.57**↑	99.10↑	99.06↑	99.95↑	99.91↑
Seq-05	99.52↑	**99.81**↑	99.38↑	99.00↑	99.87↑	99.97↑
Seq-06	**99.54**↑	99.43↑	98.79↑	98.13↑	99.90↑	99.74↑

on long-term performance are typically smaller compared to those of the LiDAR-based method.

As for influences of geometry variation across different regions on PR performance, we find that the average Recall@1 values for different regions are very close. For almost $\{\langle k, j\rangle\}_j$, the values of average Recall@1 for different regions are all more than 99% (see Tab. 3.7). We guess the primary reason is that radar possesses specific penetration capabilities, enabling it to reliably observe through various obstacles, such as moving vehicles. However, the heat map colors indicating the similarity of matching results in Region-02 tend to be lighter than those in Region-01 and Region-03, as shown in the last two rows of Fig. 3.6. So, geometry difference, such as building layout and traffic conditions, still makes place recognition more difficult.

The above comparisons show the potential of radar-based methods in long-term scenarios, which is expected to achieve robust performance using general matching thresholds under diverse conditions.

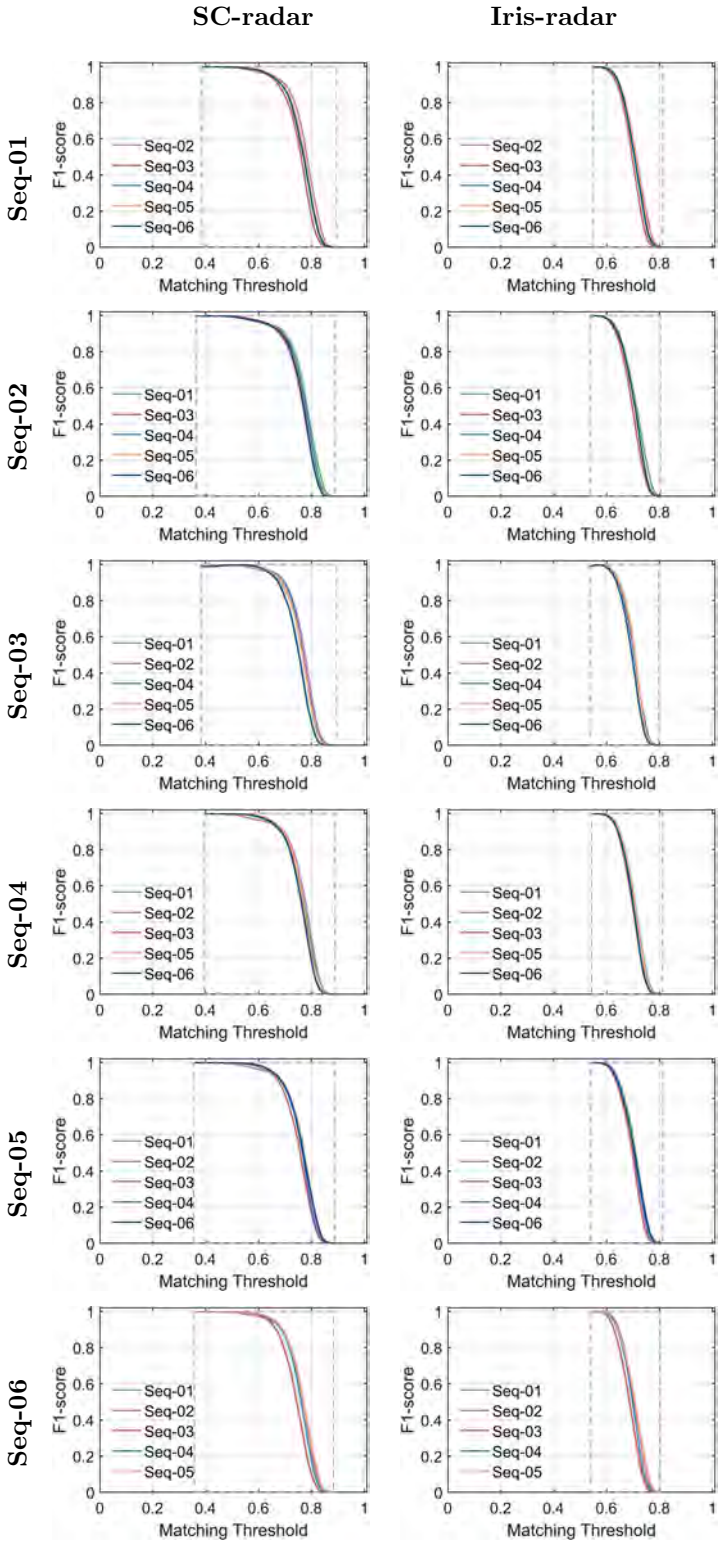


Figure 3.7: FT-curves $\{FT^k\}$ for different $\{\langle k, j\rangle\}_j$ using different radar-based PR methods. Sequence ID on the left refers to the used reference sequence in each row. Figures in each column are based on the same method.

3.6 Conclusions

In this work, we conduct a comprehensive evaluation of range sensing-based long-term place recognition in large-scale urban environments. We propose a novel metric to evaluate the influences of matching thresholds on long-term performance, which provides a new perspective of evaluation. Our experimental results provide the following important findings: i) current SOTA handcrafted feature-based LiDAR PR methods are more robust to season and weather variations in long-term and large-scale scenarios; ii) with a general matching threshold, current SOTA data-driven LiDAR-based PR methods tend to provide results with larger variations in long-term scenarios; iii) the variation in geometry information across different regions in large-scale environments, such as building layouts and traffic conditions, can lead to performance degradation. Such degradation is much smaller in handcrafted feature-based LiDAR methods; iv) radar-based PR methods show strong potential in long-term, large-scale scenarios, offering superior robustness in diverse weather and traffic conditions.

Chapter 4

Place Recognition based Localization using Lightweight Semantics

Global metric localization is one of the fundamental capabilities for autonomous vehicles. Most existing methods rely on global navigation satellite systems (GNSS). Some methods relieve the need of GNSS by using 3-D LiDARs. They first achieve place recognition with a pre-built geo-referenced point-cloud database for coarse global localization, and then achieve 3-DoF/6-DoF pose estimation for fine-grained metric localization. However, these methods require accessing point-cloud features and raw point clouds, making them inefficient and hard to be deployed in large-scale environments. To provide a solution to this issue, we propose a global metric localization method with triplet-based histogram descriptors. Specifically, we first convert the input LiDAR point clouds into a semantic graph and describe the vertices in the graph with the proposed descriptor for vertex matching and pose estimation. These vertex descriptors are then selected and aggregated into a global descriptor to decide whether two places correspond to the same place according to a similarity score.

Experimental results on the KITTI dataset demonstrate that our method generally outperforms the sate-of-the-art methods.

4.1 Introduction

Global metric localization refers to finding vehicle positions at the metric level relative to a given map, usually a geo-referenced map, such as Google Map. It is a fundamental capability for autonomous vehicles to navigate in real-world environments. Vision-based global localization has attracted great attentions in recent decades due to the affordability of visual cameras and great advancement of computer vision technologies. However, vision-based methods suffer from the intrinsic limitations of visual cameras, such as changes of illuminations, viewpoints, weathers, and seasons, etc.

Recently, many global localization methods use 3-D LiDARs due to their robustness to the changes of visual appearances and illuminations. The pipeline of the LiDAR-based methods can be generally divided into two steps: 1) Perform place recognition by matching the global descriptors extracted from both the current and off-line LiDAR point clouds; 2) Estimate relative metric poses by registering the current point cloud to the retrieved point cloud from the database. However, these methods require accessing point-cloud features and raw point clouds for place recognition and pose estimation, making them inefficient and hard to be deployed in large-scale environments. To address this issue, some researchers [71, 74] propose to design descriptors to achieve both place recognition and pose estimation. In this way, only point-cloud features are needed to be accessed, so the memory cost is reduced and the efficiency is increased.

However, many existing descriptor-based methods can only achieve 1-DoF/3-DoF localization and might be degraded when significant viewpoint changes happen between the on-line and off-line point clouds. So, in this work, we propose a novel method

called Triplet-Graph to address the above issues. Specifically, we convert a point cloud into a semantic object graph, and design an innovative semantic triplet-based histogram descriptor to embed semantic, topological, and geometric information for each vertex (i.e., an object) in the graph. We extract vertex descriptors from two different point clouds, then perform vertex matching and estimate the relative pose based on the matched vertices. The estimated pose and the descriptors of the vertices are further employed to build global descriptors by simple statistical calculations. The global descriptors are used to decide whether two places correspond to the same place according to their similarity score. The framework of our proposed method is shown in Fig. 4.1. Our contributions are summarized as follows:

1. We propose novel semantic triplet-based histogram descriptors for global metric localization using 3-D LiDAR point clouds.

2. We design a simple yet effective adding strategy to fast aggregate vertex descriptors into a global descriptor to reduce memory consumption.

3. We demonstrate that our method generally outperforms state-of-the-art methods in terms of both place recognition and pose estimation.

4.2 Related Work

The existing global metric localization methods using key-frame-based database can be generally divided into two categories: one-shot and filtering. Our method belongs to one-shot, which means that for the on-line data we only need the point cloud from the current moment. Different from one-shot, filtering methods require a sequence of on-line point clouds from different moments for filter algorithms, such as Bayesian filter. In this review, we only focus on one-shot methods, and divide them into non-6-DoF and 6-DoF methods.

4.2.1 Non-6-DoF Global Localization

In this work, non-6-DoF includes 1-DoF and 3-DoF. The 1-DoF global localization only estimates headings (i.e., yaw angles), while the 3-DoF localization estimates 2-D positions (i.e., x and y coordinates) and headings.

Existing methods usually convert point clouds from 3-D to 2-D, such as bird-eye-view (BEV) images and range images, to extract their descriptors. Scan Context [71] is a typical BEV-based method, in which raw point clouds are converted into BEV polar grids. The largest z coordinate among the points in each block of the polar grids is used to build a matrix-based global descriptor. The heading difference between the query frame and the matching frame is estimated by column shifting when their similarity reaches the maximum. Based on Scan Context, Wang et al. [74] combined intensity information with geometric information for descriptor extraction. Only 1-DoF pose estimation is provided in [71] and [74]. To achieve 3-DoF global localization, Wang et al. [77] proposed a global descriptor, LiDAR-Iris, which is extracted from the generated binary signature image of the LiDAR point cloud. The 3-DoF pose estimation is obtained from the Iris after Fourier transform. Xu et al. [75] transformed the Scan Context-based feature into the frequency domain to learn a rotation-invariant descriptor and pose. Kim et al.[72] proposed Scan Context++ by extending Scan Context with a Cartesian BEV-based descriptor to obtain the translation estimation. Instead of using geometric information, Li et al. [73] explored semantic information to extract Scan Context-based global descriptor. A two-step global semantic Iterative Closest Point method was proposed to find the 3-DoF pose, which was further used to improve matching performance. Instead of using polar grids, Ding et al. [101] performed Radon Transform to extract descriptors based on BEV images, which can estimate headings between point clouds without being affected by the translation variations. Lu et al. [102] also used Radon Transform for descriptor extraction, while the proposed learning-free descriptor can estimate both headings and translations.

Differently, some researchers used a spherical projection model to convert point clouds into range images instead of 2-D BEV images. Chen et al. [103] proposed OverlapNet, a deep neural network that can estimate overlap and relative yaw angle between two LiDAR point clouds. Lukas et al. [104] proposed a data-driven system for place recognition that estimates yaw discrepancies between LiDAR point-cloud scans at the same time.

4.2.2 6-DoF Global Localization

A common solution to achieve 6-DoF pose estimation relies on the correspondence of geometric elements (e.g., key points, lines, and planes). Shan et al. [105] projected 3-D point clouds to get intensity images, in which ORB feature descriptors are extracted and used for place recognition queries using BoW. The matched candidate is further validated by Perspective-n-Point (PnP) and Random Sample Consensus (RANSAC). Giammarino et al. [106] conducted experiments to show that straightforward adaptation of existing visual place recognition techniques on intensity information can achieve reliable loop closure detection. However, an expensive high-resolution LiDAR is usually needed to obtain dense intensity images for feature extraction.

Recently, some researchers directly extracted 3-D features from point clouds for local matching and global descriptors for place recognition. For example, Cattaneo et al. [107] proposed LCDNet, an end-to-end approach for both place recognition and pose estimation. A PV-RCNN-based network is used to extract local descriptors. The proposed pose regression network can register two point clouds with an arbitrary initial misalignment. Du et al. [108] integrated multi-level context information and channel-wise relations into local features using FlexConv and an SE block. The extracted local features are then aggregated as a global descriptor for place recognition. The 6-DoF pose estimation is also achieved by the local feature matching. Komorowski et al. [109] proposed EgoNN, a deep neural network based on MinkLoc3D to extract both

local and global descriptors. Different from DH3D, EgoNN can process more larger point clouds, while the number of points should be less than 10,000 in DH3D. Instead of performing place recognition based on the global descriptor generated from local features, Cui et al. [110] used 3-D features LinK3D [111] to build a BoW for loop closure detection. The full 6-DoF pose estimation was then obtained by using RANSAC and Singular Value Decomposition (SVD) on the point-to-point LinK3D matching results. Boss et al. [112] and Guo et al. [113] both used probabilistic voting strategy for place recognition. The 6-DoF poses are estimated based on 3-D keypoint and intensity-integrated keypoint, respectively.

The above methods all use point-level features, which are not memory efficient in large-scale environments. Instead, some researchers use semantic objects to build descriptors. GOSMatch [69] represents environments as semantic object graphs. Histogram-based descriptor was proposed for both global descriptor and vertex descriptor. Similarly, BoxGraph [114] builds a semantic object fully-connected graph from point clouds. The authors combined the object shape and centroid information into the vertex descriptor for both place recognition and pose estimation.

Like GOSMatch and BoxGraph, our method also uses a semantic graph to encode environment information. But differently, we explicitly use geometric, semantic, and topological information from stable static objects in the environment. GOSMatch [69] uses vehicle information, which is not stable and consistent when time changes. BoxGraph [114] does not explicitly use topological information for descriptor extraction. We believe that the topological information is beneficial for building a more descriptive representation for a semantic graph.

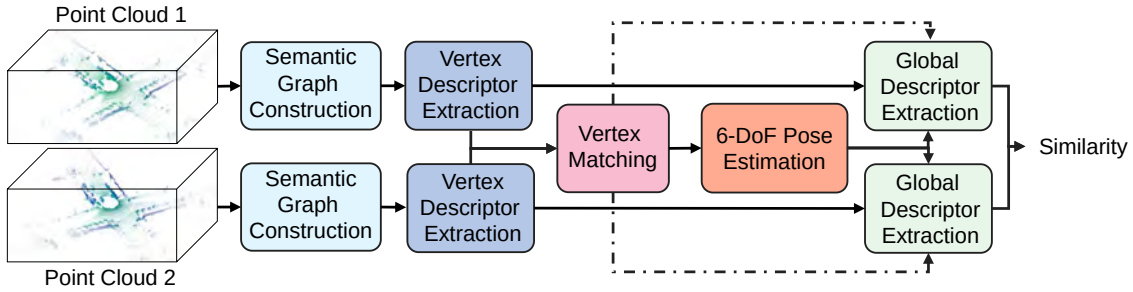


Figure 4.1: The framework of our proposed method. Point Cloud 1 and Point Cloud 2 respectively refer to an off-line point cloud from a pre-built database and an on-line point cloud captured at the current moment from a vehicle-mounted LiDAR.

4.3 Methodology

The framework of our proposed method is shown in Fig. 4.1. The method is generally divided into four parts: semantic-graph construction, vertex descriptor extraction and matching, 6-DoF pose estimation, as well as global descriptor extraction and similarity computing.

4.3.1 Semantic Graph Construction

Fig. 4.2(a) shows the semantic graph construction process. Given a point cloud, we first find its semantic segmentation result. We only consider static objects in this work. The class label is denoted as l, in which $l \in L = \{$sidewalk, building, fence, vegetation, trunk, pole, and traffic sign$\}$. Points with the same class label are then clustered into different object groups using Euclidean Cluster algorithm implemented by Point Cloud Library [115]. Each clustered group has its own ID, class label, and corresponding points. To improve the consistency of object clustering across different frames, we treat the clustered group as one individual object only when the number of points within the group is larger than a pre-defined threshold, which varies with the class label of the group. It is larger for sidewalk, building, and vegetation than the other classes.

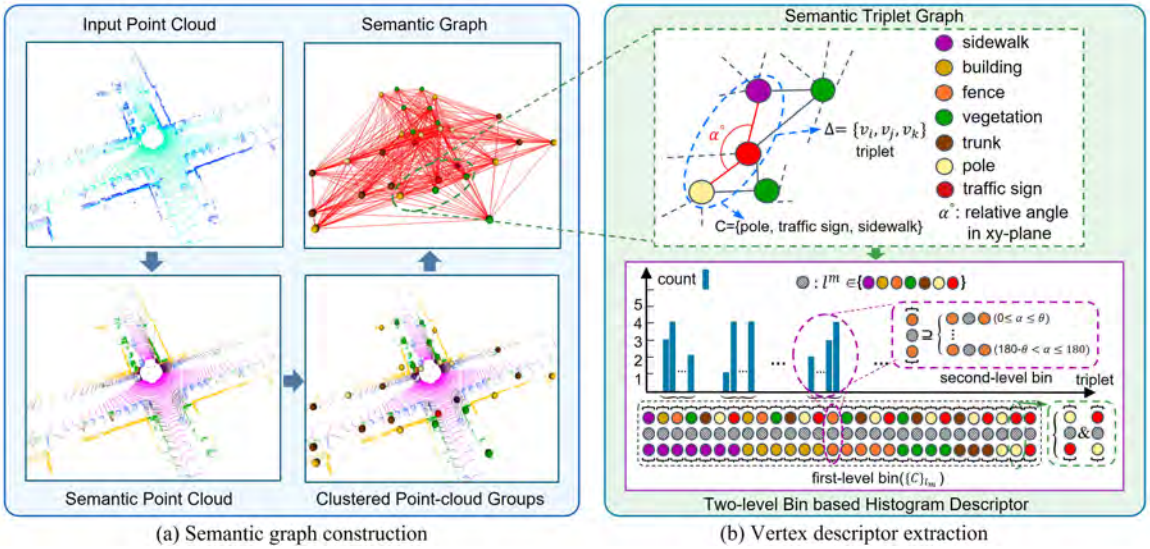
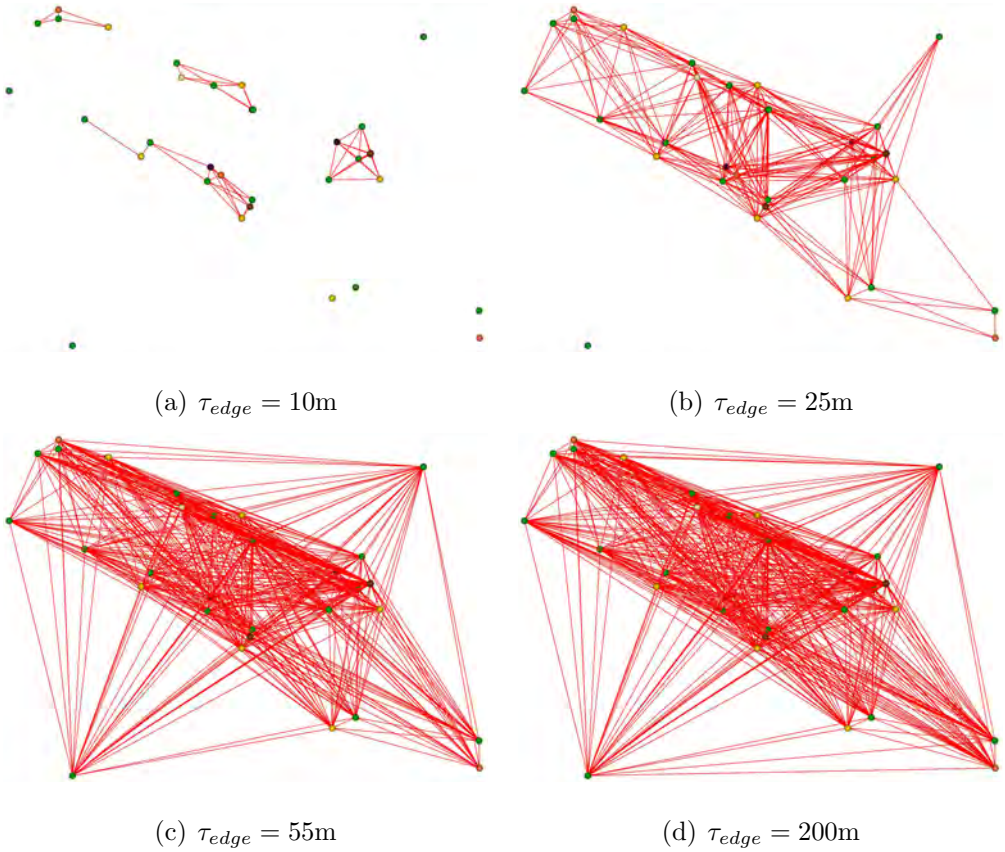


Figure 4.2: (a) Semantic graph construction and (b) vertex descriptor extraction. Semantic segmentation is first performed on the input LiDAR point cloud. A semantic graph is then constructed based on the clustered objects. The blue dotted ellipse shows a sample triplet. Finally, the triplet semantic histogram-based descriptor is employed to represent the vertices in the graph. l_m could be an arbitrary label from set L.

Similar to [69, 114], we represent the clustered point-cloud groups 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 object-i in the point cloud. Each vertex contains the geometric centroid position (i.e., xyz-coordinate) and class label-l of the object. $\boldsymbol{E} = \{e_{ij}\}$ 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 objects are connected only when their distance is less than a pre-defined threshold $\tau_{edge} = 55m$. As shown in Fig. 4.3, the semantic graph becomes a completely connected graph when τ_{edge} is large enough.



(a) $\tau_{edge} = 10$m (b) $\tau_{edge} = 25$m

(c) $\tau_{edge} = 55$m (d) $\tau_{edge} = 200$m

Figure 4.3: Semantic graphs with different values of τ_{edge}.

4.3.2 Triplet-based Vertex Descriptor Extraction and Matching

The geometric centroids of the clustered objects can be regarded as an abstract representation of the point cloud at the instance level. To find the relative pose between two scans based on two semantic graphs, vertices correspondences are needed. Inspired by the semantic histogram-based descriptor used in [116], we also use a semantic histogram descriptor to embed semantic and topological information for vertices in a graph. Differently, we explicitly embed geometric information into the histogram based on the relative angle among vertices. To calculate such a relative angle, at least three vertices are needed. However, using more vertices (e.g., >3) to calculate the relative angle at once can lead to higher computational complexity. So, we only use

three vertices and use the word *triplet* to name our descriptors.

We denote $\Delta = \{v^f, v^m, v^t\}$ as a triplet, in which f, m, and t are short for $first$, $middle$, and $third$, respectively. Δ_{ijk} refers to the triplet when $v^f = v_i$, $v^m = v_j$, $v^t = v_k$, in which $i \neq j \neq k$. v_i and v_k are both connected to v_j. The relative angle of Δ in the xy-plane is denoted as α. $C = \{l^f, l^m, l^t\}$ denotes the class combination for a triplet. An example of a Δ can be found in the blue dotted ellipse at the top right of Fig. 4.2(b). Note that Δ_{ijk} and Δ_{kji} are regarded as the same Δ because objects in point clouds are unordered. $\{\Delta\}_{v_j}$ is the set of all the triplets that satisfy $v^m = v_j$ in the graph.

Histogram based on Two-level Bins

To build a histogram, the first step is to "bin" the range of values, that is, to divide the entire range of values into a set of intervals. We innovatively design two-level bins to classify and count triplets.

We design the first-level bins based on class combinations of triplets. Specifically, given the class label set L, we first chose one class label for l^m of C, such as $l^m = \text{trunk}$. Then, we exhaustively chose class labels for l^f and l^t. All these class combinations form the first-level bins of the histogram for vertex whose class label is trunk, denoted as $\{C\}_{l^m=\text{trunk}}$. Since objects in point clouds are unordered, we regard {fence, trunk, pole} and {pole, trunk, fence} as the same C. The length of $\{C\}_{l^m=\text{trunk}}$ can be calculated by $N_1 = \sum_{w=1}^{|L|} \mathbf{C}_w^1$, in which $|L|$ is the cardinality of L and $\mathbf{C}$ refers to Combination in mathematics ($|L| = 7$ in this work, so $N_1 = 28$). Similarly, the first-level bins of the histogram for vertex with the other class labels can be obtained by the same way, such as $\{C\}_{l^m=\text{pole}}$, etc. It is worth noting that the first-level bins $\{C\}_{l^m}$ are different for vertices whose class labels are different, as shown in the bottom right of Fig. 4.2(b).

For each first-level bin $C \in \{C\}_{l_m}$, we further divide it into N_2 second-level bins.

Each second-level bin has the same combination as C, but with a different relative angle range. As shown in the purple dotted box in Fig. 4.2(b), we have $N_2 = 180°/\theta$, in which θ (we set θ as 5°) is the interval of the relative angle range. Note that θ should be divisible by 180°. When $\theta = 180°$, the second-level bins disappear, which means that only semantic combinations of triplets are embedded in the histograms, leading to performance degradation. The details can be found in .

Vertex Descriptor Extraction

Given a vertex v_j from a constructed graph, we first build the first-level bins (i.e., $\{C\}_{l^m}$) for its vertex histogram descriptor according to the class label of v_j. Then, we search the graph to get $\{\Delta\}_{v_j}$. For each Δ in $\{\Delta\}_{v_j}$, we find the corresponding first-level bin that has the same class combination C as Δ. By comparing α with the cover range of relative angles for the second-level bins, we can easily know which second-level bin the current triplet Δ belongs to. Then, the triplet number in the retrieved second-level bin is increased by one. Repeat the above steps for all the Δ in $\{\Delta\}_{v_j}$, we can obtain the final triplet-based histogram descriptor for v_j, see Fig. 4.2(b). The extracted vertex descriptor can be easily converted to a $N_1 \times N_2$ matrix, denoted as Des_{v_j}. Each element in the matrix equals the number of triplets in $\{\Delta\}_{v_j}$ with the same class combination C and range of α in the graph.

Vertex Matching

Vertex matching between two graphs $\boldsymbol{G_1}$ and $\boldsymbol{G_2}$ can be realized by comparing the similarity between the descriptors of vertices in the two graphs. Only vertices with the same class label are compared, for example, trunks in $\boldsymbol{G_1}$ will only be compared with trunks in $\boldsymbol{G_2}$. We use the cosine similarity to compare the similarity:

$$\mathrm{Sim}\left(v_j^1, v_t^2\right) = \frac{\sum Des_{v_j^1} \cdot Des_{v_t^2}}{\left\|Des_{v_j^1}\right\|_F \times \left\|Des_{v_t^2}\right\|_F}, \tag{4.1}$$

in which Sim is short for similarity, the dot $\cdot$ is the element-wise multiplication on two matrices, $\|\cdot\|_F$ is the Frobenious norm of a matrix, $\sum$ is the summation of all elements of a matrix, v_j^1 and v_t^2 are respectively arbitrary vertices in $\boldsymbol{G_1}$ and $\boldsymbol{G_2}$ with the same class label. For every v_j^1 in $\boldsymbol{G_1}$, we chose the vertex v_t^2 in $\boldsymbol{G_2}$ that has the highest similarity between v_j^1 as the matching result, denoted as $(v_j^1,\, v_t^2)$. We use $\{(v_j^1,\, v_t^2)\}$ to represent all vertex matching results between $\boldsymbol{G_1}$ and $\boldsymbol{G_2}$.

4.3.3 6-DoF Pose Estimation

There are mainly two steps in the 6-DoF pose estimation. Firstly, we use RANSAC [117] and SVD to estimate a coarse pose, $\widetilde{\mathrm{T}} \in SE(3)$, based on the vertex matching results $\{(v_j^1,\, v_t^2)\}$. Secondly, we optimize the pose estimation by minimizing the total projection error based on the inliers matches after the first step. For two given matched vertices, the projection error can be obtained as $\varepsilon = \left\|\bar{v}_j^1 - T \cdot \bar{v}_t^2\right\|$, in which $T \in SE(3)$ is the pose needed to be optimized, $\bar{v}_j^1$ and $\bar{v}_t^2$ are respectively the homogeneous coordinates for the vertices v_j^1 and v_t^2. The Ceres Solver [118] is used to solve this optimization problem. We denote the optimal 6-DoF pose obtained through the optimization process as T^*. To speed up the optimization process, we use $\widetilde{\mathrm{T}}$ as the initial guess for the optimization.

4.3.4 Global Descriptor Extraction and Similarity Computing

To achieve place recognition, the similarity between two point clouds is required. We aggregate the local (vertex) descriptors into a global descriptor to measure the similarity.

Global Descriptor Extraction

Let $\{Des_{v_j}\}$ denote the set of all vertex descriptors in a semantic graph. As mentioned in Section III-B, the first- and second-level bins of the histogram are exactly the same for the vertices whose class labels are the same. Therefore, Des_{v_j} and $Des_{v_{j+1}}$ from the same graph can be directly added together element-wisely when the class labels for v_j and v_{j+1} are the same. Instead of directly aggregating all the vertices in a graph into a global descriptor, we use remaining matched vertices. Specifically, we perform the projection operation (as mentioned in Part C) using T^* and the vertex matching results $\{(v_j^1, v_t^2)\}$. Only when the projection error is less than a pre-defined threshold τ_{proj}, the corresponding vertex match is kept. The remaining vertices matches are denoted as $\{(v_j^1, v_t^2)\}_{rem}$. For $\boldsymbol{G_1}$, we add up descriptors for vertices with the same class label-l from $\{(v_j^1, v_t^2)\}_{rem}$, denoted as $Des^{l,1}$. A $Des^{l,1}$ records all the triplets whose middle class is label-l from the remaining vertices in $\boldsymbol{G_1}$, which is the overall description for class-l in $\boldsymbol{G_1}$. Repeat the adding operation for every class label-l. Then, the global descriptor for $\boldsymbol{G_1}$ is the set of all $Des^{l,1}$, denoted as $\{Des^{l,1}\}$, in which $l \in L$. The global descriptor $\{Des^{l,2}\}$ for $\boldsymbol{G_2}$ can be obtained by the same way.

Similarity computing

We use cosine similarity to measure how close two global descriptors are. Specifically, given two global descriptors $\{Des^{l,1}\}$ and $\{Des^{l,2}\}$, we first calculate the similarity between $Des^{l,1}$ and $Des^{l,2}$ for each class-l. Then, a weighted model is adopted to combine the similarity scores for all the classes to get the final score between the global descriptors from the two graphs:

$$\mathrm{Sim}\left(\boldsymbol{G_1}, \boldsymbol{G_2}\right) = \sum_{l \in L} \frac{1}{|L|} \times \frac{\sum Des^{l,1} \cdot Des^{l,2}}{\left\|Des^{l,1}\right\|_F \times \left\|Des^{l,2}\right\|_F}. \tag{4.2}$$

4.4 Experiments

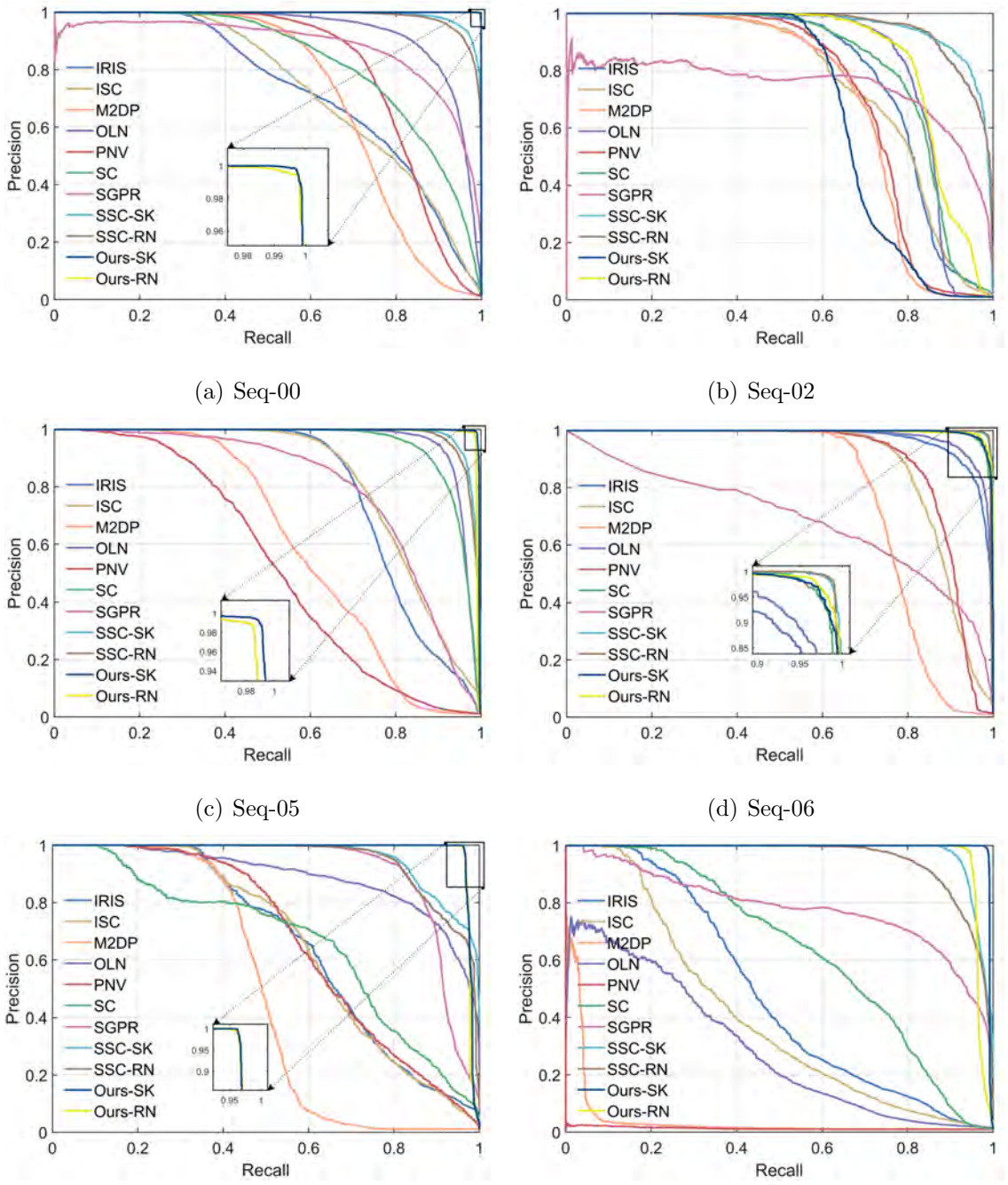
4.4.1 Dataset and Experimental Setup

We evaluate our method using the KITTI odometry dataset [119]. There are 11 sequences with the 6-DoF trajectory ground truth in the dataset. Loop closure only occurs in sequence 00, 02, 05, 06, 07, and 08. So, we only perform experiments on these sequences. Note that sequence-08 contains reverse loops (i.e., revisiting the same place from the opposite direction), while loop closure only happens with the same direction in the other sequences. To ensure a fair comparison, we follow the same setup as in [120] and [73]. They use a large number of point-cloud pairs for evaluation. A true-positive loop closure occurs when the relative distance is less than 3m between a point cloud pair. If the distance exceeds 20m, the corresponding pair is considered as a negative sample. We use the benchmark and evaluation pairs provided by [73]. For each sequence, there are N_p positive samples and $100 \cdot N_p$ negative samples that are selected randomly.

Following [73], we use the ground-truth semantic labels from the SemanticKITTI dataset [121], and the prediction labels from RangeNet++ [122] as the point-cloud semantic segmentation results, respectively denoted as SK and RN in Tab. 4.1 and Tab. 4.2. Due to the page limit, we present the parameter tuning results for τ_{edge}, θ, and τ_{proj} in terms of both place recognition and pose estimation in 4.4.6.

4.4.2 Place Recognition Performance

We compare our method with the open-sourced state-of-the-art methods, including Lidar Iris (IRIS) [77], M2DP [70], Overlapnet (OLN) [103], PointNetvlad (PNV) [78], SGPR[120], Scan Context (SC) [71], Intensity Scan Context (ISC) [74], and SSC [73]. In this work, we directly import the results from [73] for all the above methods.



(a) Seq-00

(b) Seq-02

(c) Seq-05

(d) Seq-06

(e) Seq-07

(f) Seq-08

Figure 4.4: Precision-Recall curves on the KITTI dataset. Seq is short for Sequence. RN and SK represent prediction labels from RangeNet++ and ground-truth semantic labels from SemanticKITTI dataset, respectively.

To ensure comprehensive evaluation, we not only show the Precision-Recall curves (see Fig. 4.4), but also provide the maximum F_1 score and the Extended Precision (EP) [65] as shown in Tab. 4.1. The F_1 score is calculated as $F_1 = 2\frac{PR}{P+R}$, in which P is precision and R is recall. For EP, we have $EP = \frac{1}{2}(P_{R0} + R_{P_{100}})$, in which P_{R0} is the precision at the minimum recall, and $R_{P_{100}}$ equals to the maximum recall at 100% precision.

As we can see in Fig. 4.4 and Tab. 4.1, our approach outperforms the other methods except on Seq-02 and Seq-06. SSC [73] outperforms our method slightly on Seq-00 and 06. We conjecture the reason could be the degradation of object clustering or inconsistency of objects between two LiDAR point clouds (i.e., some objects can not be both observed in two scans). When there are too few objects in the scene, the constructed semantic graph would be very sparse, leading to degradation in the descriptive capability of the vertex descriptor and global descriptor. Especially on Seq-02, we found that the average number of clustered objects is obviously less than that of the other sequences, which is shown as the yellow bars in Fig. 4.5. As for Seq-08, there exist many reverse loop closures (opposite viewpoints), our method can still achieve satisfactory performance of place recognition. This demonstrates the robustness of our method against viewpoint changes.

Interestingly, the place recognition performance between Ours-SK and Ours-RN is very close. This indicates that our method can be practically applicable using standard semantic segmentation networks. We guess the reason is that our method relies more on the consistency of semantic segmentation (i.e., similar segmentation results for different observations around the same location) rather than higher segmentation accuracy.

Table 4.1: Maximum F_1 score (%) and Extended Precision (EP) (%) on the KITTI dataset. Larger values indicate better performance. The best results are highlighted in bold font.

Methods	Seq-00		Seq-02		Seq-05		Seq-06		Seq-07		Seq-08	
	F_1	EP	F_1	EP	F_1	EP	F_1	EP	F_1	EP	F_1	EP
IRIS (ICRA 2020)[77]	66.8	62.6	76.2	66.6	76.8	74.7	91.3	79.1	62.9	65.1	47.8	56.2
M2DP (IROS 2016)[70]	70.8	61.6	71.7	60.3	60.2	61.1	78.7	68.1	56.0	58.6	7.3	50.0
OLN (Auton. Robot. 2021)[103]	86.9	55.5	82.7	63.9	92.4	79.6	93.0	74.4	81.8	58.6	37.4	50.0
PNV (CVPR 2018)[78]	77.9	64.1	72.7	69.1	54.1	53.6	85.2	76.7	63.1	59.1	3.7	50.0
SGPR (IROS 2020)[120]	82.0	50.0	75.1	50.0	75.1	53.1	65.5	50.0	86.8	72.1	75.0	52.0
SC (IROS 2018)[71]	75.0	60.9	78.2	63.2	89.5	79.7	96.8	92.4	66.2	55.4	60.7	56.9
ISC (ICRA 2020)[74]	65.7	62.7	70.5	61.3	77.1	72.7	84.2	81.6	63.6	63.8	40.8	54.3
SSC-RN (IROS 2021)[73]	93.9	82.6	89.0	74.5	94.1	90.0	**98.6**	**97.3**	87.0	77.3	88.1	73.2
Ours-RN	99.6	92.6	83.1	**76.4**	98.6	87.8	97.6	86.4	97.8	96.0	96.3	93.2
SSC-SK (IROS 2021)[73]	95.1	84.9	**89.1**	74.8	95.1	90.3	98.5	96.9	87.5	80.5	94.0	93.2
Ours-SK	**99.8**	**99.5**	71.6	74.1	**99.1**	**92.5**	97.1	92.6	**98.1**	**97.9**	**99.0**	**97.3**

4.4.3 Pose Estimation Performance

We compare our results with SSC [73] on the positive pairs. Note that SSC can only provide 3-DoF pose estimation (x, y, yaw), while ours provides 6-DoF pose estimation. Relative Translation Error (RTE) and Relative Rotation Error (RRE) [123] are employed to evaluate the translation and orientation accuracy, respectively. RTE is calculated as $RTE = \|t_{est} - t_{gt}\|_2$, in which $t_{est} \in T^*$ is the estimated translation, t_{gt} is the ground-truth for the estimated translation. RRE is calculated as $RRE = \cos^{-1}\left(\frac{\mathrm{Tr}\left(R_{gt}^{-1}R_{est}\right)-1}{2}\right)$ or $RRE = \cos^{-1}\left(\frac{\mathrm{Tr}\left(R_{est}^{T}R_{gt}\right)-1}{2}\right)$, in which $R_{est} \in T^*$ is the estimated rotation matrix, R_{gt} is the ground truth for the estimated rotation matrix, and $Tr(\cdot)$ represents the trace operation.

Tab. 4.2 shows the RTE average and RRE average with standard deviations. We can see that our method presents better translation estimations than SSC on almost all the sequences except Seq-02. Especially on Seq-00 and Seq-06, the average RTE of our method is less than 10cm. As for orientation estimation, SSC only outperforms ours

Table 4.2: Average RTE (m) and average RRE (degree) with standard deviations on the KITTI dataset. Smaller values indicate better performance. The best results are highlighted in bold font.

		Seq-00	Seq-02	Seq-05	Seq-06	Seq-07	Seq-08
RTE	SSC-RN	0.38±0.43	1.07±0.81	0.51±0.59	0.33±0.37	0.35±0.45	0.45±0.57
	Ours-RN	**0.07±0.05**	0.95±1.61	0.16±1.01	**0.07±0.05**	0.35±1.59	0.55±2.23
	SSC-SK	0.32±0.41	**0.91±0.81**	0.47±0.60	0.30±0.38	0.30±0.33	0.25±0.33
	Ours-SK	**0.07±0.13**	1.66±3.72	**0.11±0.56**	0.09±0.10	**0.22±0.10**	**0.20±1.09**
RRE	SSC-RN	0.58±2.24	**0.87±1.15**	0.66±4.53	0.64±0.81	1.43±10.43	1.35±1.37
	Ours-RN	0.37±0.28	4.52±20.69	1.05±9.64	**0.26±0.16**	2.13±14.46	2.34±9.87
	SSC-SK	0.80±5.75	1.22±6.03	0.62±0.67	0.76±0.79	**0.52±2.16**	1.55±1.37
	Ours-SK	**0.34±0.31**	16.74±44.56	**0.58±6.30**	0.27±0.27	1.75±13.08	**1.04±2.56**

on Seq-02 and Seq-07. On the other sequences, our method outperforms SSC with a large margin, which can generally achieve an average RRE at the sub-degree level. In general, our method can provide a competitive metric-level 6-DoF pose estimation against SSC (only 3-DoF available).

As we can see in Tab. 4.2, on Seq-02, our method cannot well estimate the poses. There are two main possible reasons for this: 1) There might be too many incorrect vertices matches, leading to inferior optimization results; 2) Too few vertices matches are obtained, which cannot provide enough constraints during the optimization process. To better assess the difference between clustered objects in different sequences, we calculate the average number of the clustered objects (i.e., the average number of objects in an input LiDAR point-cloud pair). In addition, object number difference (i.e., the absolute value of object number difference between the LiDAR point-cloud pair) and the object number difference among pairs whose RTE/RRE is larger than the upper quartile of RTE/RRE are also calculated.

The average values and standard deviations of the above items for every sequence are visualized in Fig. 4.5. Generally, small object number difference refers to better consistency of clustered objects in two frames, which should usually have a better

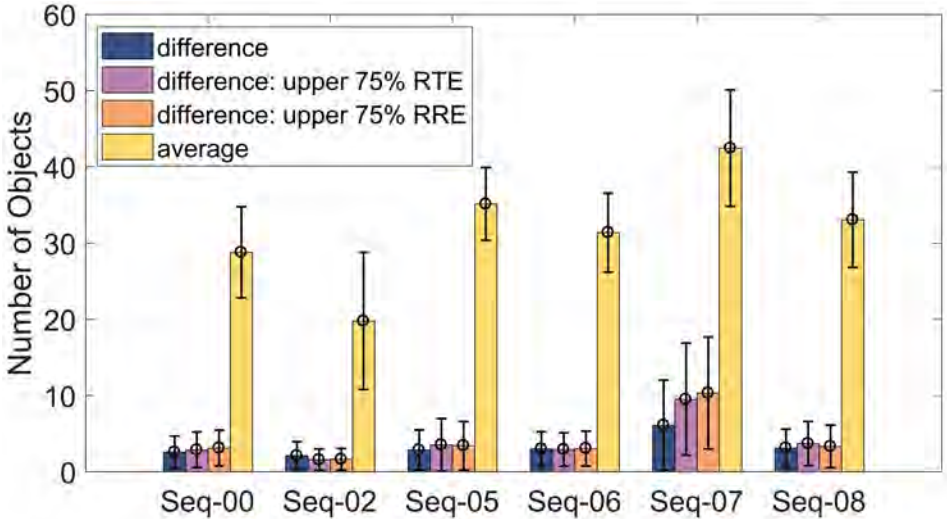
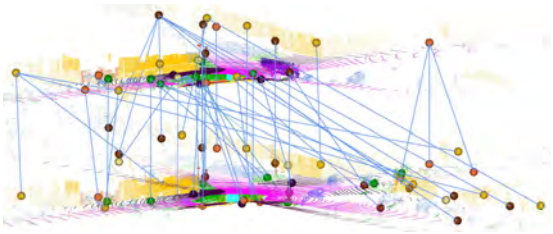
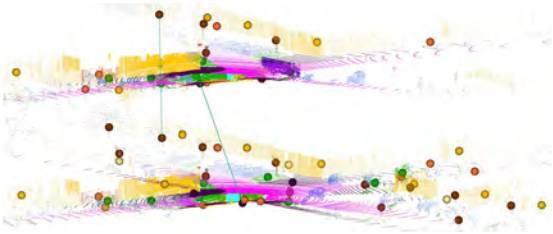


Figure 4.5: Distribution of the numbers of clustered objects for different KITTI sequences in the form of bar chart. Results are obtained based on the ground-truth semantic labels. Seq is short for Sequence.

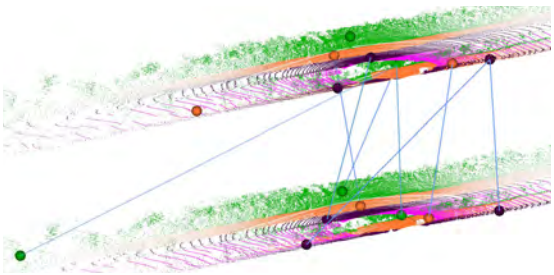
vertex matching result. Besides, a larger average object number contributes to a denser graph, which can also result in a better vertex matching result. Although the object number difference is small on Seq-02, we find that its average object number is the smallest, which is less than 20, while the other sequences have at least 30 objects for each scene on average. In addition, we find that all the three object number differences on Seq-07 are much larger than those on the other sequences. Such a huge difference leads to degradation in pose estimation even Seq-07 has the largest average object number. These results are all consistent with our conjecture. Some examples of vertices matches are visualized, including a failing example with occlusion caused by a truck (see Fig. 4.6(a) and (b)), a failing example with few clustered objects (see Fig. 4.6(c) and (d)), and a success example (see Fig. 4.6(e) and (f)). The degradation of vertex matching can lead to the failure of pose estimation, which would further affect place recognition. In conclusion, a larger average object number and less object number difference can significantly improve the pose estimation performance.



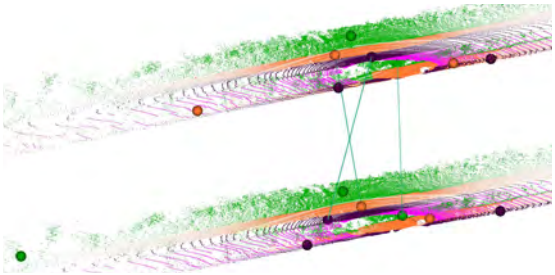
(a) Original matches in a scene with occlusion



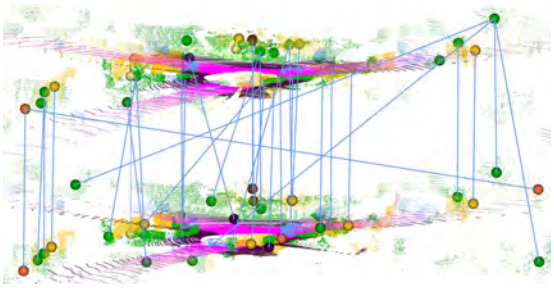
(b) Remaining matches in a scene with occlusion



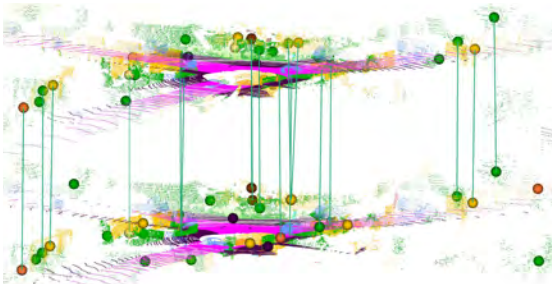
(c) Original matches in a scene with few objects



(d) Remaining matches in a scene with few objects



(e) Original matches in a simple scene



(f) Remaining matches in a simple scene

Figure 4.6: Examples for original vertices matches (the left column) and the remaining vertices matches (the right column). Original vertices matches represent the matching results from Section III-B. Remaining vertices matches refer to the selected vertices for global descriptor extraction in Section III-D. In each figure, the upper and lower point cloud represents the visited and the revisiting frame, respectively. The figure is best viewed in color.

Table 4.3: Maximum F_1 score (%) and Extended Precision (EP) (%) on the KITTI dataset with/without vertices matches selection operation for global descriptor extraction.

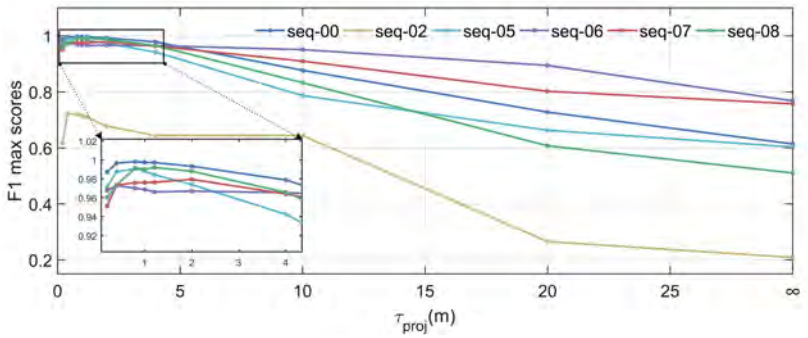
Sequence	F_1		EP	
	(w/ select)	(w/o select)	(w/ select)	(w/o select)
Seq-00	99.8	57.3	99.5	56.9
Seq-02	71.6	17.9	74.1	50.4
Seq-05	99.1	56.7	92.5	50.9
Seq-06	97.1	73.1	92.6	64.0
Seq-07	98.1	76.6	97.9	73.6
Seq-08	99.0	47.7	97.3	53.0
average	94.1	54.9	92.3	58.1

4.4.4 Ablation Study

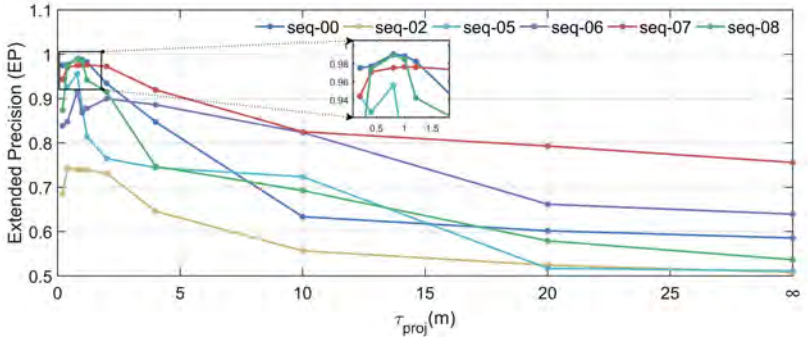
To demonstrate the contribution of the projection-based selection operation, we disable the selection operation and directly generate the global descriptor using all vertices. Note that our ablation study is conducted using ground-truth semantic labels to avoid the impact caused by the quality of semantic segmentation. Results can be found in Tab. 4.3. As we can see, the vertices matches selection operation can significantly improve the performance of place recognition in term of both maximum F_1 score and EP. The average maximum F_1 score and average EP are increased by 0.392 and 0.342, respectively.

4.4.5 Memory Consumption

Compared to using raw point clouds, only vertex descriptors need to be accessed in our method. On average, throughout all the sequences, there are 31 clustered objects in a graph. So, the average memory consumption for a graph is $(N_1 \times N_2 + 4) \times 4 \times 31$



(a) F_1 max scores



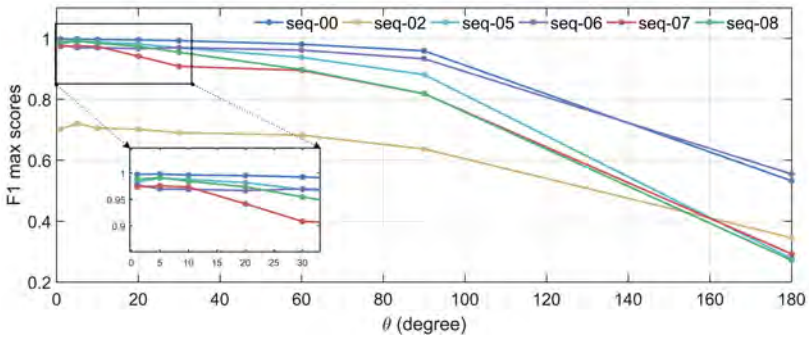
(b) Extended Precision (EP)

Figure 4.7: F_1 max score and Extended Precision corresponding to different τ_{proj}.

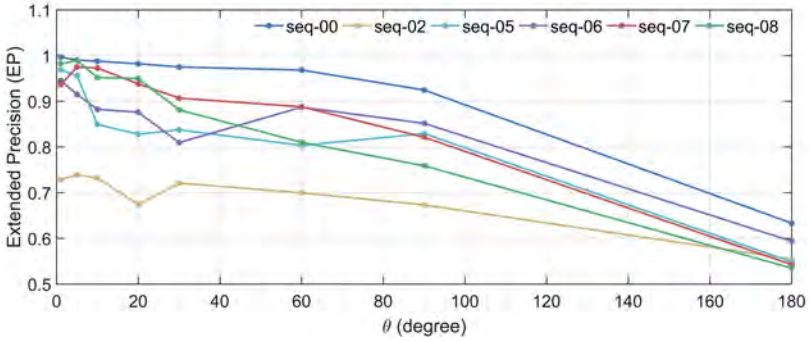
bytes. In our case, this equates to 125.5KB. The average number of 3-D points per point-cloud scan is 123,584, which approximately consumes 1483.0KB of memories. This demonstrates our high efficiency in terms of memory consumption.

4.4.6 Parameter Tuning

In this section, we investigate the effects of τ_{edge}, θ, and τ_{proj} on both place recognition and relative pose estimation. For each parameter, we change its value while keep the others static, and then we run our method for all the sequences. To reduce the interference from semantic segmentation result, all the tuning experiments are performed based on the ground-truth semantic labels.



(a) F_1 max scores

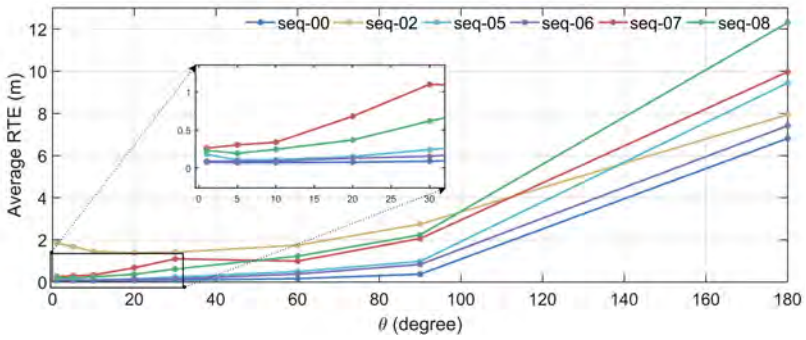


(b) Extended Precision (EP)

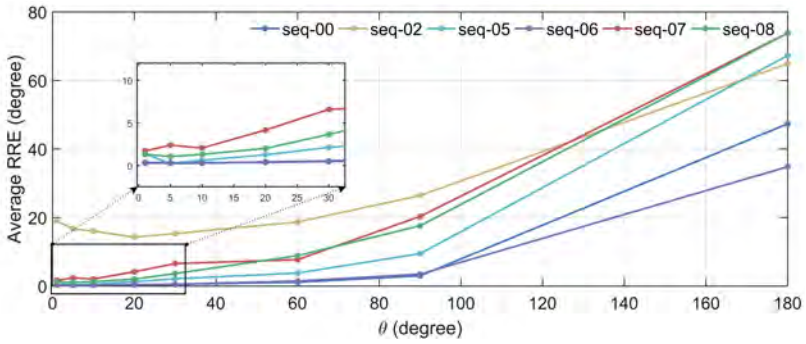
Figure 4.8: F_1 max score and Extended Precision corresponding to different θ.

Effect of τ_{proj}

τ_{proj} only affects place recognition because it is only used for global descriptor extraction. As shown in Fig. 4.7, both F_1 max scores and EP generally decrease when τ_{proj} becomes larger. This is because the vertex matching merely based on vertex descriptor is not perfect, there are still some incorrect matching result. The project operation can be considered a geometry verification, which can effectively filter the incorrect matches. However, when such verification is too strict, i.e. a small value of τ_{proj}, both F_1 max scores and EP decrease. The reason could be the error from pose estimation T^* and the geometry centroid coordination of the object, which will lead to the position inconsistency between the same object in two different frame after projection.



(a) Average RTE(m)

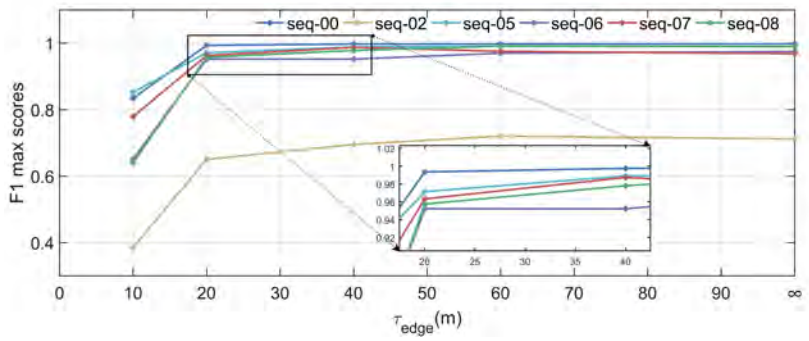


(b) Average RRE(degree)

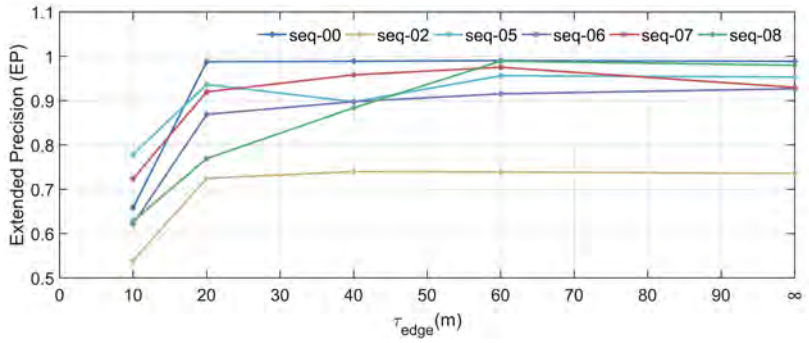
Figure 4.9: Average RTE and average RRE corresponding to different θ.

Effect of θ

In our method, θ directly affects the dimension of the vertex descriptor and the global descriptor. Fig. 4.8 and shows the F_1 max score and EP to different θ. Fig. 4.9 shows average RTE and average RRE corresponding to different θ. As θ increases, the performance of place recognition and pose estimation gradually decrease among most of the sequences. Obviously, when θ is larger, the dimension of the vertex and global descriptor become smaller. As a consequence, triplets with the same semantic combination while with different relative angle α contribute equally for the descriptor construction. The discriminating power for both vertex and global descriptors degrades, leading to more incorrect vertex matches. Therefore, the pose estimation and place recognition become worse. Meanwhile, we also found that when θ is too small (i.e. 1 degree), the general performance also degrades. The main reason



(a) F_1 max scores



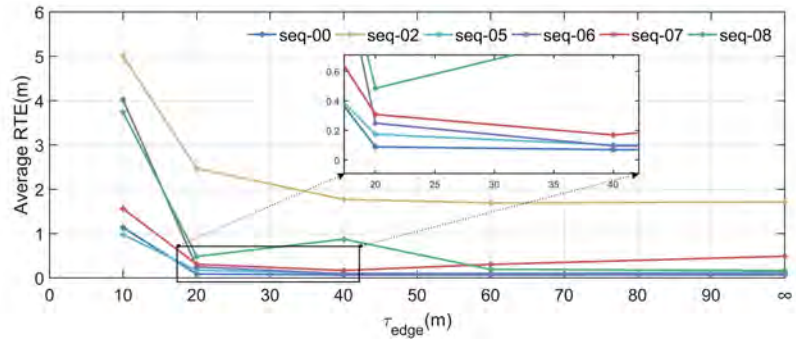
(b) Extended Precision (EP)

Figure 4.10: F_1 max score and Extended Precision corresponding to different τ_{edge}.

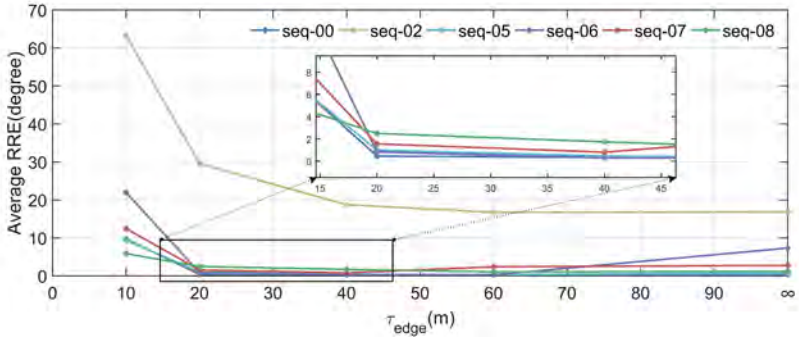
is that we can not obtain the consistent geometry centriod of the same object from two different LiDAR frame. View-point change and local obstruction can easily lead to such inconsistency.

Effect of τ_{dege}

Different values of τ_{dege} will result in different constructed graphs. As τ_{dege} increases, the number of edges connecting vertexes increases, which means the graph becomes denser. And the overall performance becomes better, as shown in Fig. 4.10 and 4.11. Theoretically, a small τ_{dege} leads to less triplet for the same vertex, which means the vertex descriptor focuses on embedding local information. In addition, the same object in frame-1 might not be observed in frame-2, which can lead to a huge difference between the local information in different frames. A denser graph can



(a) Average RTE(m)



(b) Average RRE(degree)

Figure 4.11: Average RTE and average RRE corresponding to different τ_{edge}.

embed both local and global information for each vertex, which can greatly relieve the inconsistency between vertex descriptors caused by missing some common objects. However, we also found that the overall performance will slightly decrease when τ_{dege} is big enough (i.e. a fully-connected graph), such as in seq-02, 07, and 08. We conjecture the reason could be that the 6-DoF pose between the query and matched frames is relatively larger, making a huger difference between the cluttered objects, especially those that are far away from the center. Empirically, a value of τ_{dege} that can connect most of the vertexes but not fully connect every two vertexes has a better overall performance.

4.5 Conclusion

In this work, we propose a novel method, TripletGraph, to achieve both place recognition and 6-DoF relative pose estimation using lightweight semantics. For every input LiDAR point cloud, a semantic graph will be first constructed to represent the scenario in a compact and high-level way. We design a triplet-based semantic histogram descriptor to embed both geometric information and topology information for the every vertex in the semantic graph. Such descriptors are then used to perform pose estimation and aggregated into a global descriptor for place recognition. We have shown quantitative and qualitative results for our proposed method, which also shows our high competitive against the-state-of-art algorithms.

Chapter 5

Registration based Localization using Lightweight Semantics

This chapter 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.

5.1 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 multipath 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 [5, 9].

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) [71, 74, 73, 77, 12]. 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 [124, 125, 126] 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 [124, 126] and condition-meeting [125] strategies have been proposed to build instance-to-instance correspondences. However, they may still be inefficient at times when addressing the registration 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

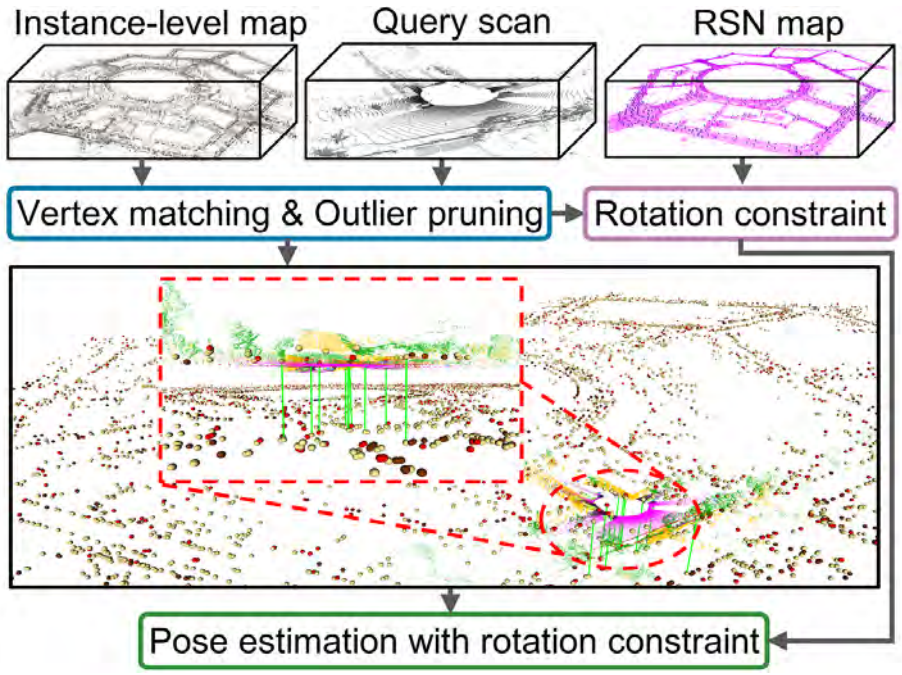


Figure 5.1: TripletLoc Framework. Semantic graphs are constructed to represent both the single query scan and the global reference map. Vertex descriptors are then extracted from the graphs to build instance-to-instance correspondences. Graph-theoretic outlier pruning and rotation constraint from the RSN map are integrated for the 6-DoF pose estimation.

a compact instance-level semantic graph, which has been used in the current methods [12, 125, 69, 114] 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, an RSN map is proposed to provide a prior rotation constraint for better pose estimation performance. Lastly, graph-theoretic pruning is used in the 6-DoF pose estimation to improve the robustness against outlier correspondences. The framework of TripletLoc is shown in Fig. 5.1. 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 [12] 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.

5.2 Related Work

LiDAR-based global localization can be generally categorized into two branches: one-shot-based and sequence-based methods [5]. 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 [127]. 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.

5.2.1 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 [5]. 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 [67]. Instead of using local features, global descriptors extracted directly from LiDAR points have become another popular solution for PR in recent years [5]. Scan Context [71] embeds geometric information from a 3-D point cloud as a 2-D matrix global descriptor using bird-eye-view (BEV) projection. Similarly, SSC[73], Intensity Scan Context [74], RING++[76], and LiDAR-Iris[77] all use BEV projection to extract their global descriptors. More recently, graph structure has been used in many methods to embed environment layouts [69, 114, 12, 128]. Most of them rely on the clustered semantic instances to build semantic graphs [69, 114, 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 [128]. 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 [78, 81, 82].

5.2.2 Registration-based Methods

Unlike PR-based methods, which use a key-frame-based retrieval method, 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 3-D point-level correspondences. Alternatively, several researchers extract instances from environments to build correspondences [124, 125, 126]. All-to-all strategy is used to build correspondences in [126], 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 [124]. 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 [125], 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 method to select inlier correspondences for pose estimation. These methods might build a large number of correspondences in large-scale environments, making it very slow or even computationally intractable for outlier pruning.

5.2.3 Differences from Previous Studies

Our previous work, Triplet-Graph [12], is a typical scan-to-scan based approach. Each scan includes its surrounding instances, which saves the same instance 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 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.

5.3 The Proposed Method

Our proposed one-shot global localization method 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. 5.1

5.3.1 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. Instead, we represent the scene at the instance level for better efficiency, as shown in Fig. 5.2. To enhance localization robustness, we also propose a novel RSN map (see Fig. 5.3) for prior rotation constraints in 6-DoF pose estimation.

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 [129] 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 [115]. 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 0.5m between each other into a new single

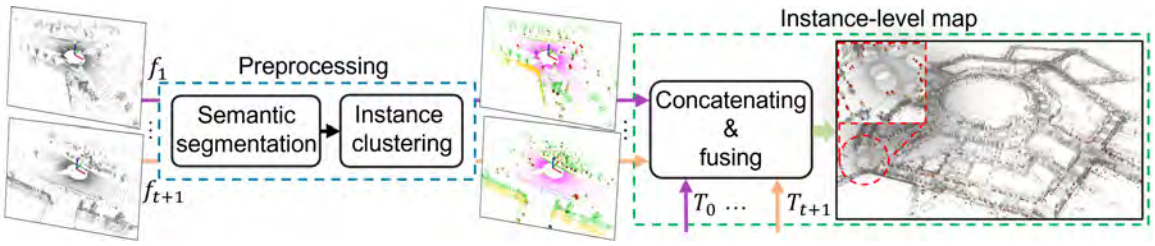


Figure 5.2: The construction pipeline of the instance-level map. Semantic segmentation is first performed on each LiDAR scan, followed by instance clustering. The clustered instances of all scans are then concatenated and fused as a large-scale reference map.

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.

RSN Map

Based on the semantic segmentation results from SPVNAS, we divide 3D road points into discrete road grids (10m×10m). Within each grid, if the number of 3D points exceeds a pre-defined threshold (1000 in 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.

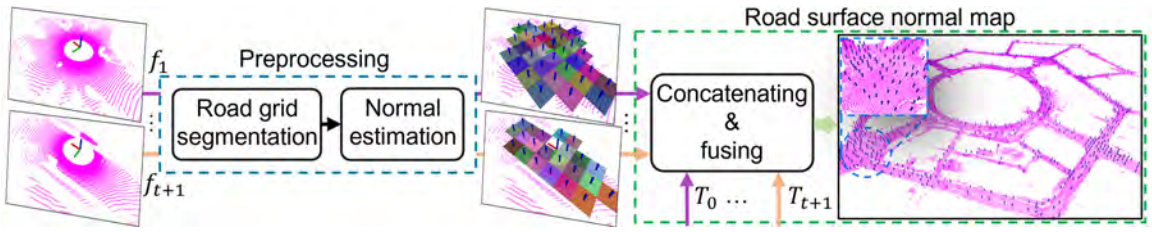


Figure 5.3: The construction pipeline of the RSN map. Road grid segmentation is first performed on road points, followed by road surface normal estimation. The road-grid-based surface normal of all scans are then concatenated and fused.

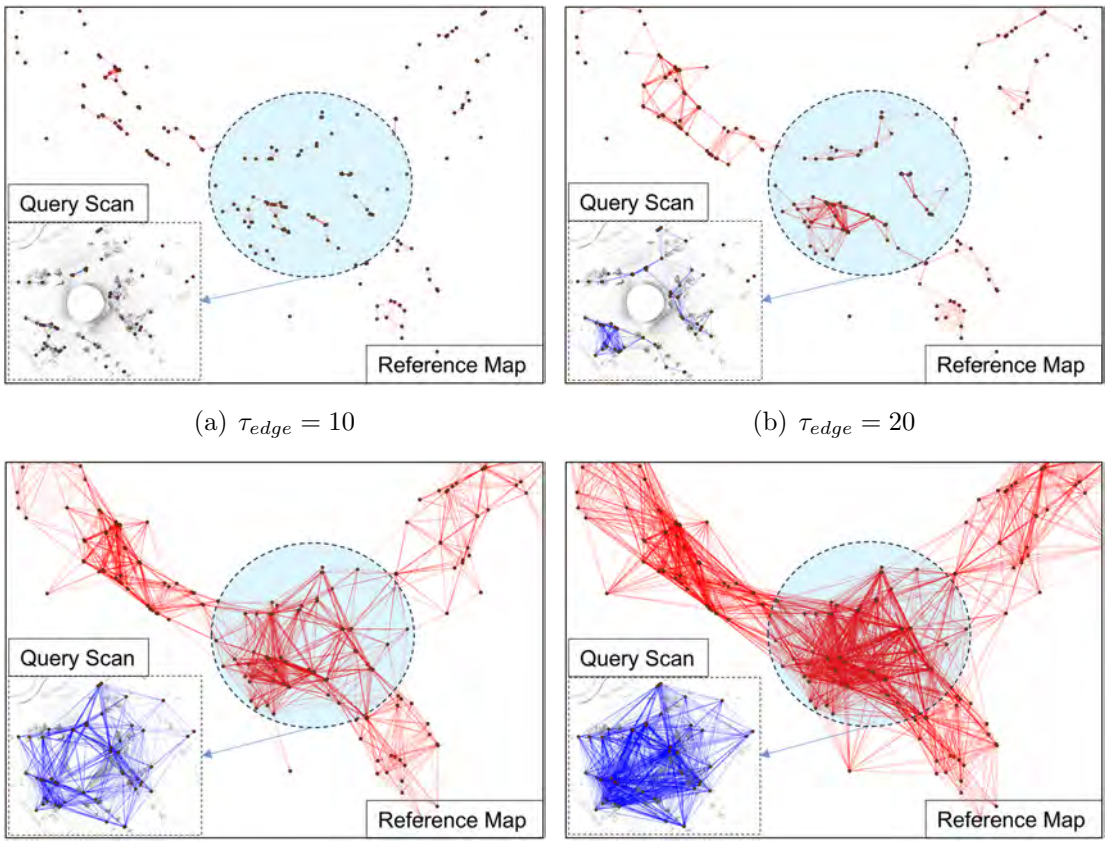
We also compute the relative angle between each fused normal and the new normal. These angles (differences) are used to determine the standard deviation, $\sigma_{\underrightarrow{n}} \in \mathbb{R}$, which reflects the uncertainty of the road surface normal estimation ($\underrightarrow{n}$) of the local area.

5.3.2 Vertex Descriptor Extraction and Matching

Instance correspondences are needed to register the query LiDAR scan to the instance-level map. We extend the vertex descriptor proposed in our previous method [12] to build more robust correspondences.

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 single 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,



(a) $\tau_{edge} = 10$ (b) $\tau_{edge} = 20$

(c) $\tau_{edge} = 40$ (d) $\tau_{edge} = 60$

Figure 5.4: 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.

as shown in Fig. 5.4. Therefore, we use a smaller threshold (i.e., 20m) to reduce such differences.

Vertex Descriptor Extraction

In our previous work, Triplet-Graph [12], 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

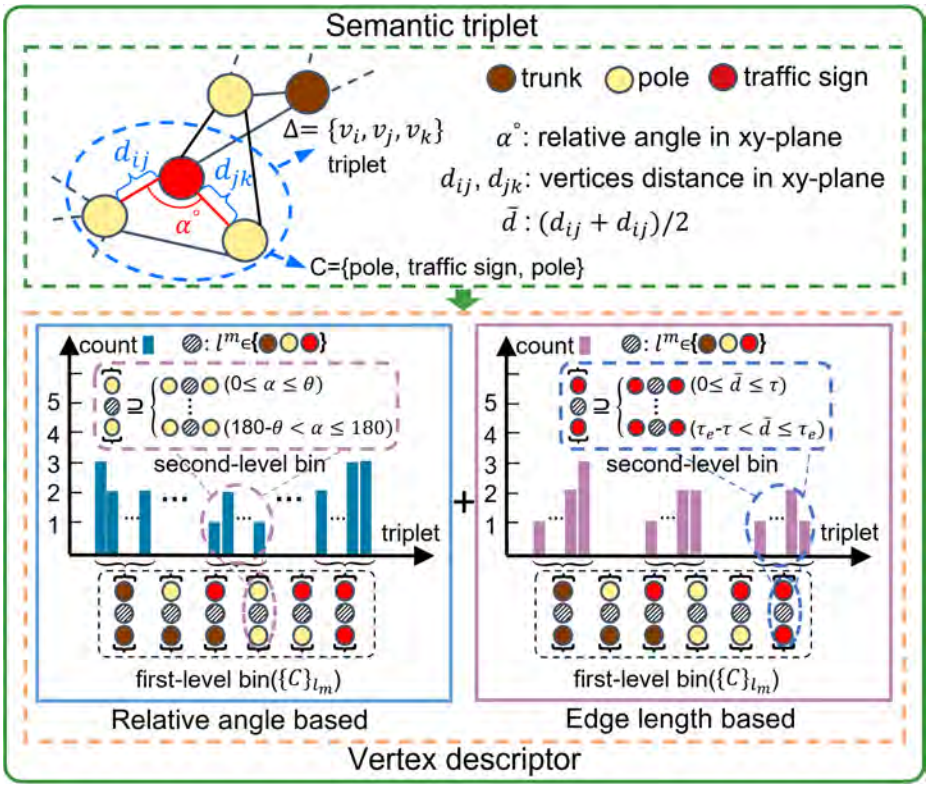


Figure 5.5: Vertex descriptor extraction. A semantic graph is first constructed to represent the input LiDAR point cloud. The blue dotted ellipse shows a sample triplet. The triplet semantic histogram-based descriptor is then employed to represent the vertices in the graph.

as a histogram descriptor, as shown in the blue box in Fig. 5.5. 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[12] 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. 5.5, 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 $\overline{d} = (d_{ij} + d_{jk})/2$. Since two vertices are connected only when their distance is less than τ_{edge}, we have $\overline{d} \leq \tau_{edge}$. Similar to $Des^{\alpha}_{v_j}$, a histogram based on two-level bins is used to embed the edge length information (i.e., $\overline{d}$) 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. 5.5. 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 [12] 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. 5.5, we have $N_3 = \tau_{edge}/\tau$, in which τ (we set τ as 0.5m) 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}.

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, similar to our previous work [12]. 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:

$$\mathrm{Sim}\left(v_j^{que}, v_t^{ref}\right) = \frac{\sum Des_{v_j^{que}} \cdot Des_{v_t^{ref}}}{\left\|Des_{v_j^{que}}\right\|_F \times \left\|Des_{v_t^{ref}}\right\|_F}, \tag{5.1}$$

in which Sim is short for similarity, the dot $\cdot$ is the element-wise multiplication on two matrices, $\|\cdot\|_F$ is the Frobenious 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 $\mathcal{A}_{raw}$. A K-D tree is created to accelerate the matching process. Specifically, given two normalized vectors $\boldsymbol{a}$ and $\boldsymbol{b}$, their cosine similarity $cos(\boldsymbol{a}, \boldsymbol{b})$ and euclidean distance $dis(\boldsymbol{a}, \boldsymbol{b})$ follows: $cos(\boldsymbol{a}, \boldsymbol{b}) = 1 - 0.5dis(\boldsymbol{a}, \boldsymbol{b})^2$. 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.

5.3.3 Pose Estimation

Given a raw set of instance-wise correspondences $\mathcal{A}_{raw}$ between $\boldsymbol{G_{que}}$ and $\boldsymbol{G_{ref}}$, we aim to achieve robust 6-DoF pose estimation for the relative pose of a single query LiDAR scan against the instance-level map.

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 will be 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 use a graph-theoretic pruning method to filter outliers in $\mathcal{A}_{raw}$.

Given arbitrary two associations a_j and $a_i \in \mathcal{A}_{raw}$. Assuming $a_j = (v_j^{que}, v_t^{ref})$ and $a_i = (v_i^{que}, v_k^{ref})$, they are considered geometrically consistent only when the distance

between the points is preserved, i.e., $\|\overline{v}_i^{que} - \overline{v}_j^{que}\| = \|\overline{v}_k^{ref} - \overline{v}_t^{ref}\|$. $\overline{v}_i^{que}$ and $\overline{v}_j^{que}$ are the xyz coordinates for vertex v_i^{que} and v_j^{que} from the query scan, respectively. $\overline{v}_k^{ref}$ and $\overline{v}_t^{ref}$ are the xyz coordinates for vertex v_k^{ref} and v_t^{ref} from the instance-level reference map, respectively. In practice, due to measurement noise, a threshold δ_{cons} is set to consider associations consistent when $dis(a_i, a_j) = |\|\overline{v}_i^{que} - \overline{v}_j^{que}\| - \|\overline{v}_k^{ref} - \overline{v}_t^{ref}\|| < \delta_{cons}$. Then the in-liner associations $\mathcal{A}$ can be obtained by finding the largest set of consistent associations, which can be defined formally as:

$$\begin{aligned} &\max_{\mathcal{A} \subset \mathcal{A}_{raw}} |\mathcal{A}|, \\ &\text{s.t. } dis(a_i, a_j) < \delta_{cons}, \ \forall a_i, a_j \in \mathcal{A} \end{aligned} \tag{5.2}$$

The above problem can be modeled as a graph whose vertices represent associations and edges represent consistent associations. The optimal solution $\mathcal{A}$ is equivalent to the maximum clique of the graph. Although finding the Maximum Clique (MCQ) of the graph is NP-hard, it can be solved relatively quickly for sparse graphs using the parallel maximum clique (PMC) algorithm [130]. Noted that different types of semantics instances are treated uniformly during the MCQ searching.

Point-to-Point Registration

After the outliers pruning, we estimate the unknown transformation between the single 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 vertex pair can be associated by:

$$\overline{v}_t^{ref} = \boldsymbol{R}\overline{v}_j^{que} + \boldsymbol{t} + \boldsymbol{\epsilon}_{jt}, \tag{5.3}$$

where $\boldsymbol{\epsilon}_{jt}$ models the measurement noise.

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 [131] to calculate the estimation of $\boldsymbol{T}$, $\hat{\boldsymbol{T}} = [\hat{\boldsymbol{R}}, \hat{\boldsymbol{t}}]$, as a Truncated Least Squares (TSL) registration problem:

$$\hat{\boldsymbol{R}}, \hat{\boldsymbol{t}} = \underset{\mathbf{R}\in SO(3), \mathbf{t}\in\mathbb{R}^3}{\arg\min} \sum_{j,t\in\mathcal{B}} \min\left(\left\|\overline{v}_t^{ref} - \boldsymbol{R}\overline{v}_j^{que} - \boldsymbol{t}\right\|_2, c_{jt}\right), \tag{5.4}$$

where c_{jt} is the truncation parameter[132].

Rotation Constraint using RSN Map

Eq. 5.4 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 Eq. 5.4. 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 [133]. 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:

$$\underrightarrow{\hat{n}} \cdot \underrightarrow{r_3} = 1 + \epsilon_{normal}. \tag{5.5}$$

where $\cdot$ denotes the dot product, and ϵ_{normal} 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., $\epsilon_{normal} = \sigma_{\underset{\rightarrow}{n}} + \delta$. Eq. 5.5 is then used as a prior constraint for $\hat{\boldsymbol{R}}$.

Solving using Graduated Non-Convexity

The non-convex optimization problem (Eq. 5.4) with the prior rotation constraint (Eq. 5.5) is solved using Graduated Non-Convexity (GNC) [131]. GNC is a widely used method for optimizing generic non-convex cost functions, $\rho(\cdot)$, and has applications in various fields, including vision [134] and machine learning [135]. The core idea of GNC involves introducing a surrogate cost function, $\rho_\mu(\cdot)$, controlled by a parameter μ. This function is designed so that (i) for a specific value of μ, $\rho_\mu(\cdot)$ is convex, and (ii) in the limit (typically as μ approaches 1 or infinity), it converges to the original non-convex function, $\rho(\cdot)$. GNC solves the non-convex problem by starting with its convex surrogate and gradually adjusting μ, progressively increasing the level of non-convexity until the original function is restored. The solution from each iteration serves as the initial guess for the next. Specifically, we utilize the TSL-based GNC implementation in GTSAM [136] as the solver, where the `PriorFactor` in GTSAM is employed as the prior rotation constraints described in Eq. 5.5.

5.4 Experiments

5.4.1 Dataset and Experimental Setup

Baselines

We compare the proposed TripletLoc with several state-of-the-art place recognition-based and registration-based methods. PR-based baselines include Scan Context (`SC`) [71], `STD` [128], `GOSMatch` [69], and `Triplet-Graph` [12]. We use `GLOSOM` [124] and `Outram` [125] as the registration based baselines, which both follow ont-shot based

Table 5.1: The details about the evaluated sequences from HeliPR dataset. Time span refers to the collection time span between the query sequence and the reference sequence.

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

scan-to-map mechanism. All the mentioned methods are implemented in C++ and evaluated on a PC with an Intel i7-12700F CPU and 64GB RAM.

Dataset and Setup

We evaluate TripletLoc and all the other baselines using the HeLiPR Dataset [137], 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

Table 5.2: Total clustered instance number in the reference map and average clustered instance number in a single query scan. "Ref" is short for reference. "Que" is short for query.

	Sequence	Instance Number					
		car	parking	trunk	pole	traffic-sign	total
Ref	DCC04	536	8	1534	5278	1142	8498
	KAIST04	1619	23	1228	2182	513	5565
	Roundabout01	1670	5	1274	7856	1714	12519
	Town01	1515	3	1040	4237	1210	8005
	Riverside04	415	10	1026	3104	541	5096
	Bridge01	1636	2	1561	10817	2009	16025
Que	DCC05	4.89	0.02	15.87	46.05	9.45	76.28
	DCC06	4.30	0.05	21.55	59.77	11.20	96.86
	KAIST05	17.12	0.25	10.55	16.30	4.52	48.73
	KAIST06	12.62	0.20	18.11	24.58	4.64	60.15
	Roundabout02	8.60	0.03	5.22	34.60	7.41	55.86
	Roundabout03	6.89	0.03	6.33	40.05	8.33	61.63
	Town02	5.58	0.01	4.67	22.17	6.12	38.54
	Town03	3.18	0.01	4.75	23.37	6.27	37.58
	Riverside05	4.93	0.01	9.17	37.85	6.00	57.96
	Riverside06	4.41	0.04	14.40	48.44	6.84	74.13
	Bridge02	8.62	0.02	2.64	20.93	3.92	36.13
	Bridge03	2.57	0.00	2.81	22.65	4.27	32.30

setup for PR methods in [137], we sample query scans at 10m intervals and reference scans at 5m intervals. It should be noted that `STD` originally uses accumulated sub-map of 10 consecutive scans [128]. So we here 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 use global descriptor without the selection operation to first get candidates, and then use the global descriptor with the selection operation to obtain the final loop closure result from candidates [12]. For registration-based methods, since `GLOSOM` and `Outram` didn't

open-source code for their instance-level map generation, we use the same instance clustering method and the generated map for all registration-based methods. In addition, we follow the setup for the KITTI dataset in [124], only parking and traffic signs are used in `GLOSOM`. We use default parameters for `Outram` [125] for scenario KAIST and Town. For the other scenarios, wefine-tune the parameters to limit association number to avoid memory exhaustion when searching MCQ. The details about the evaluated sequences are displayed in Tab. 5.1.

Evaluation Metrics

We use Relative Translation Error (RTE) and Relative Rotation Error (RRE) [12, 125] to evaluate the translation and rotation accuracy for global localization, respectively. RTE is calculated as $RTE = \left\|\hat{\boldsymbol{t}} - \boldsymbol{t}\right\|_2$. RRE is calculated as $RRE = \cos^{-1}\left(\frac{\mathrm{Tr}\left(\boldsymbol{R}^{-1}\hat{\boldsymbol{R}}\right)-1}{2}\right)$. 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 $RRE < 10°$, termed a true positive [137]. Similarly, we consider a localization result with $RTE < 7.5$m and $RRE < 10°$ 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.

Table 5.3: Success rate and average run time. "∗" refers to place recognition-based method. "†" refers to one-shot registration based method. "`no RSN`" is refers to omitting rotation constraint from RSN map. "`TripletLoc(`$Des^{\alpha}_{v_j}$`)`" refers only using $Des^{\alpha}_{v_j}$ for vertex matching.

Metrics	Method	Query Sequence										
		DCC05	DCC06	KAIST05	KAIST06	Roundabout02	Roundabout03	Town02	Ton03	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
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

5.4.2 Global Localization Performance

Success Rate

As shown in Tab. 5.3, 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 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 appearance and geometric structure. 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 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. As shown in Tab. 5.2, the average number of clustered instances per scan in Town02/03 (32/34) and Bridge02/03 (27/29) is much lower than in other scenarios (e.g., 93 in DCC06 and 69 in Riverside06). When few clustered instances are present, TripletLoc's performance declines. While instance numbers also affect `Triplet-Graph`[12], 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 performes reasonably well. To show the influence of RTE and RRE thresholds on success rate. We calculate success rate of TripletLoc under different RTE and RRE thresholds, as shown in Fig. 5.6 and Fig. 5.7. The results indicate that TripletLoc maintains a reasonable success rate even under stricter threshold settings. In fact, when using more stringent thresholds,

such as (5m, 5°) or (2.5m, 2.5°), the success rate only shows a small decline.

Localization Accuracy

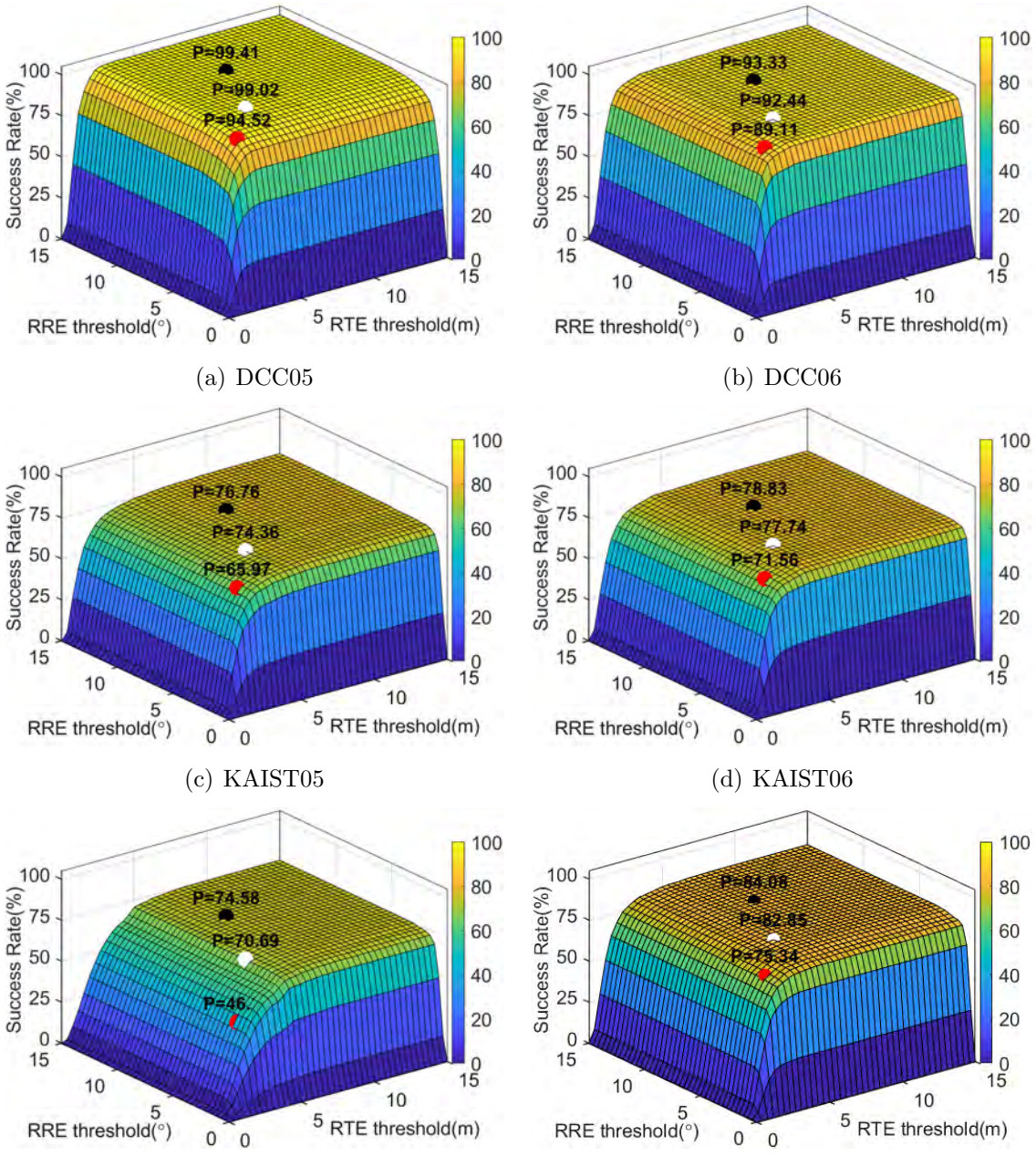
Only successfully localized scans (i.e., $RTE < 7.5$m and $RRE < 10°$) are used to evaluate localization accuracy. As shown in Tab. 5.4, registration-based methods outperform PR-based methods. `GLOSOM` shows the best RTE performance, with values under 1m in all query sequences except Riverside05 and 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 sequences. Indeed, TripletLoc achieves a higher overall success rate. So, more scans are considered for calculating 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 $RRE < 2°$ in most cases.

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 Tab. 5.3, PR-based methods generally have better efficiency. Especially, only around 2ms 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 can slow down as the number of correspondences and outliers increases. `GLOSOM` uses an all-to-all 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

Table 5.4: RTE and RRE. "-" refers to not available.

Metrics	Method	Query Sequence											
		DCC05	DCC06	KAIST05	KAIST06	Roundabout02	Roundabout03	Town02	Ton03	Riverside05	Riverside06	Bridge02	Bridge03
RTE (m)	* SC	-	-	-	-	-	-	-	-	-	-	-	-
	* GDSMatch	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^{α}_{ij})	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**
	* GDSMatch	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^{α}_{ij})	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

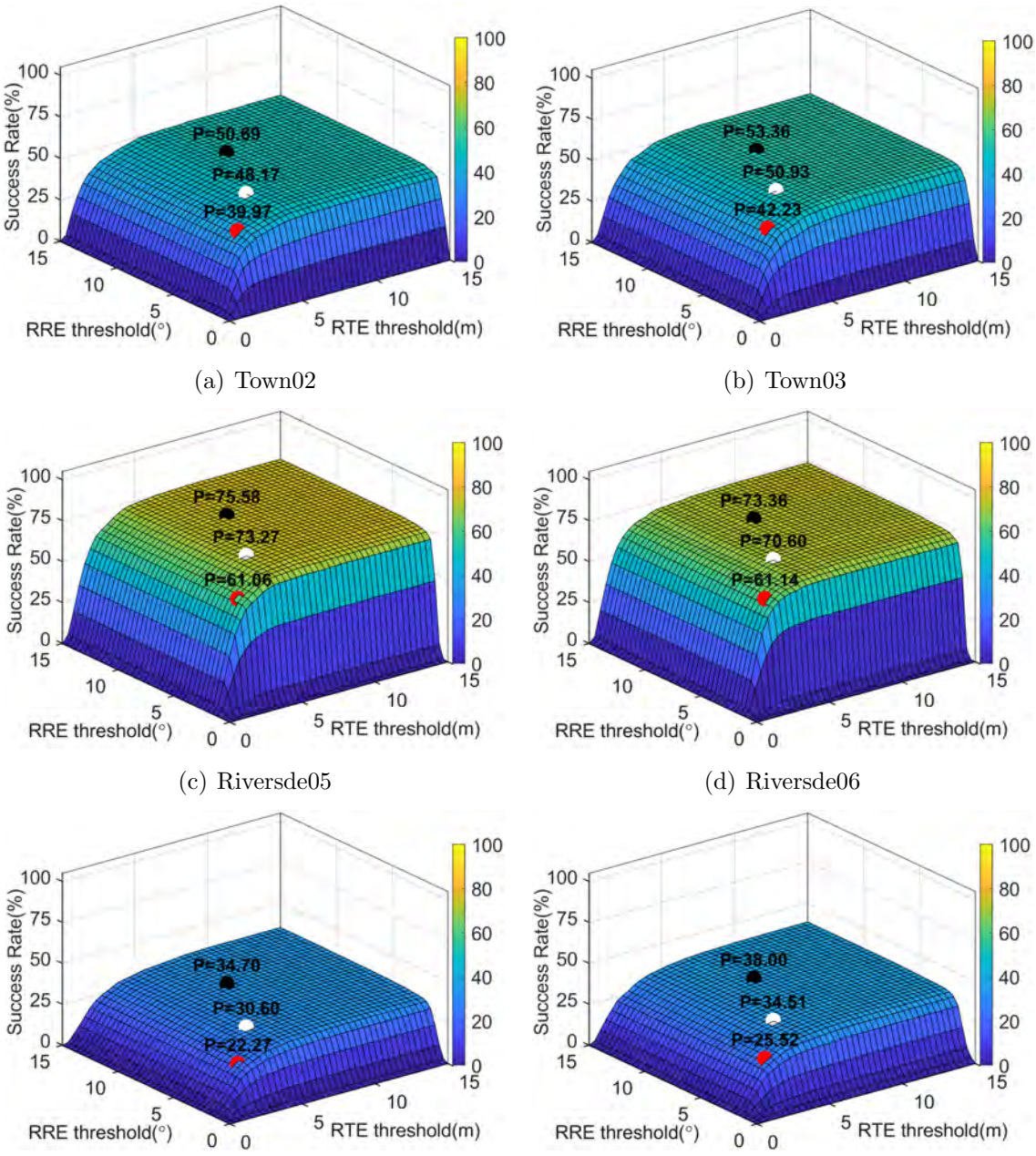


(a) DCC05 (b) DCC06

(c) KAIST05 (d) KAIST06

(e) Roundabout02 (f) Roundabout03

Figure 5.6: Examples of success rate under different thresholds of RTE and RRE. Black, white, and red dots represent success rates for thresholds of (7.5m, 10°), (5m, 5°), and (2.5m, 2.5°) for RTE and RRE, respectively.

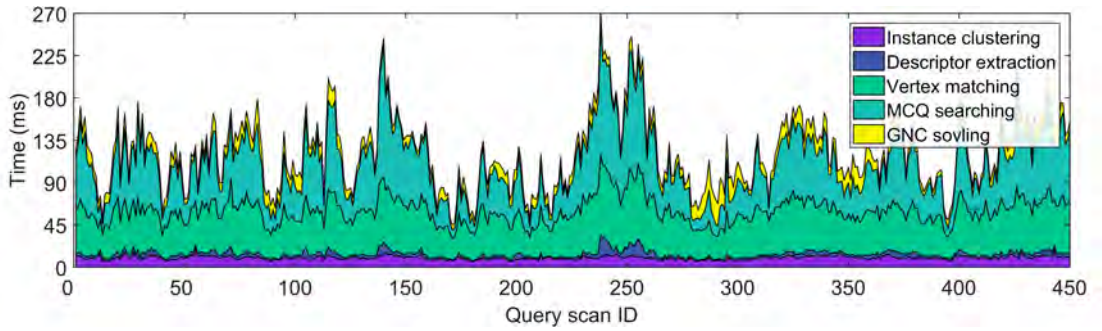


(a) Town02 (b) Town03

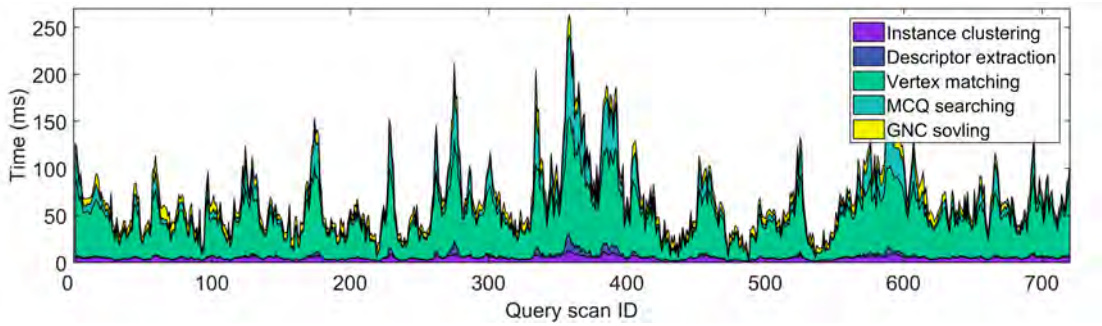
(c) Riversde05 (d) Riversde06

(e) Bridge02 (f) Bridge03

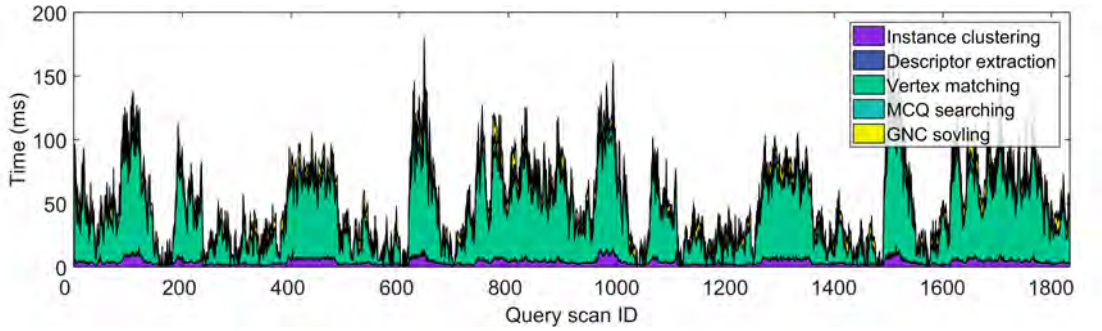
Figure 5.7: Examples of success rate under different thresholds of RTE and RRE. Black, white, and red dots represent success rates for thresholds of (7.5m, 10°), (5m, 5°), and (2.5m, 2.5°) for RTE and RRE, respectively.



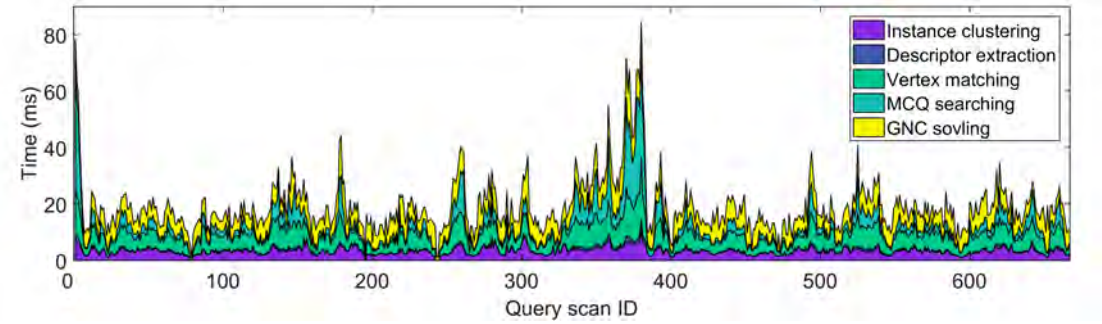
(a) DCC06



(b) Roundabout02



(c) Bridge03



(d) KAIST05

Figure 5.8: Time cost breakdown for sequence DCC06 (the slowest), Roundabout02 and Bridge03 (both intermediate), and KAIST05 (the fastest).

the number of correspondences is too large in large-scale environments. The average runtime cost for `GLOSOM` and `Outram` is higher than that for other methods. `GLOSOM` operates in real-time only when the number of instances in the reference map is relatively small (e.g., KAIST04 and Riverside04, see Tab. 5.2). Differently, TripletLoc uses simple yet efficient top-k matches to limit number of correspondences, allowing it to run much faster in real-time. Fig. 5.8 presents the breakdown of computation time per frame for TripletLoc across some query sequences, including DCC06 (the slowest), Roundabout02 and Bridge03 (both intermediate), and KAIST05 (the fastest). Overall, vertex matching and MCQ searching (including consistency graph construction) dominate, as shown in Tab. 5.5. The runtime for vertex matching depends primarily on the number of instances in the query scan, the number of instances in the reference map, and the value of k for top-k matches. As these values increase, the runtime cost for vertex matching grows accordingly. The time for MCQ searching depends on the size of $\mathcal{A}_{raw}$—the more associations that need filtering, the longer it takes to construct the consistency graph and perform the search. Descriptor extraction is fast, averaging 1.62ms per scan across all query sequences. Instance clustering and GNC solving are also efficient, averaging 5.34ms and 8.13ms, respectively.

5.4.3 Ablation Study

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 Tab. 5.3 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

Table 5.5: The average runtime (ms) for different parts of TripletLoc.

Query sequence	Average runtime cost (ms)				
	Instance clustering	Descriptor extraction	Vertex matching	MCQ searching	GNC solving
DCC05	7.95	2.04	35.63	25.70	11.32
DCC06	9.94	3.73	46.30	50.72	11.21
KAIST05	**3.26**	0.42	**5.64**	3.58	**6.10**
KAIST06	5.14	0.89	8.90	11.62	6.57
Roundabout02	4.69	1.34	44.62	10.85	7.90
Roundabout03	5.40	1.81	51.42	14.85	8.89
Town02	4.09	0.51	13.17	5.30	7.06
Town03	4.16	0.53	13.68	5.64	6.83
Riverside05	4.98	2.67	14.58	17.10	7.73
Riverside06	6.94	4.72	18.91	33.13	8.43
Bridge02	3.66	**0.34**	36.02	**2.71**	7.72
Bridge03	3.88	0.40	38.52	3.50	7.82
average	5.34	1.62	27.28	15.39	8.13

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.

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 Tab. 5.3. 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

Table 5.6: Memory consumption for prior reference map

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

constraint. These results confirm that the RSN-based rotation constraint can enhance global localization.

5.4.4 The Memory Consumption for Reference Map

To show the memory efficiency of TripletLoc, we compute 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 (28.75KB). As shown in Tab. 5.6, the memory cost for GLOSOM is the lowest. TripletLoc also shows competitive efficiency in memory.

5.4.5 Parameter Tuning

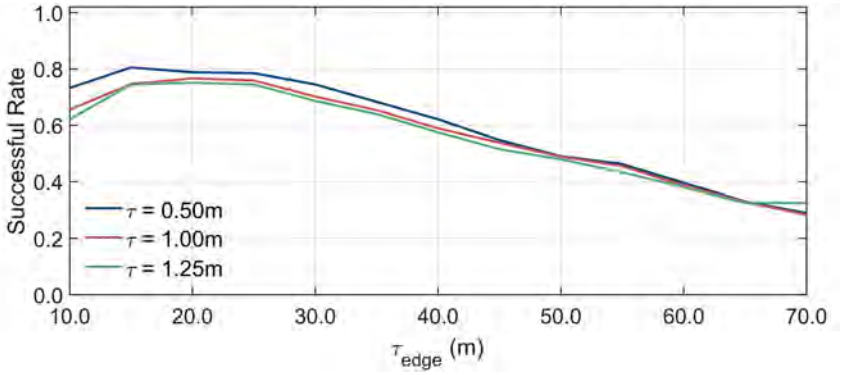
To better understand the influences of k, τ_{edge}, τ, on the global localization performance, we also provide results for parameter tuning. KAIST06 is selected as the query sequence with the instance-level map and RSN map generated from sequence KAIST04. For k, we keep τ_{edge} and τ unchanged (i.e., $\tau_{edge} = 20$m and $\tau = 0.5°$) and change the value of k from 1 to 100. As shown in Tab. 5.7, as k becomes larger, the success rate gradually increases, and RTE and RRE gradually become smaller. This is because more correspondences are used for pose estimation when k is larger, which might introduce more inlier correspondences into $|\mathcal{A}_{raw}|$. However, more runtime is

Table 5.7: Influences of k on globalization localization performance. $|\mathcal{A}_{raw}|$ is the average size of $\mathcal{A}_{raw}$.

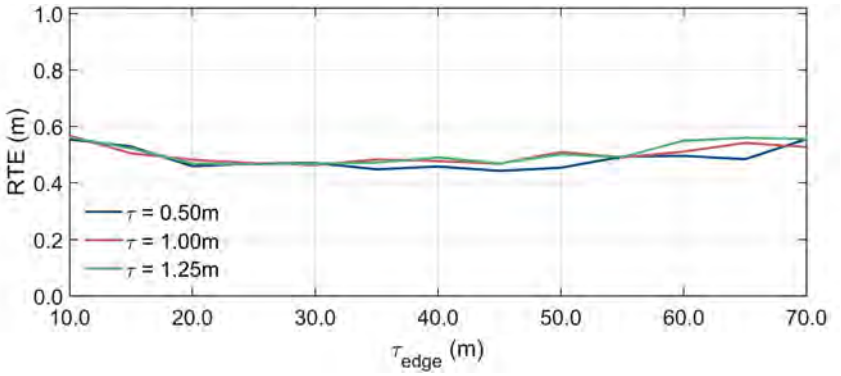
Top-k	P (%)	RTE (m)	RRE (°)	$\lvert\mathcal{A}_{raw}\rvert$	t_{ave}(ms)
1	70.17	0.55±0.38	1.67±1.42	**44**	**19.24**
5	76.04	0.50±0.33	1.49±1.20	220	22.31
10	77.74	0.49±0.34	1.38±1.15	440	23.30
15	77.43	0.47±0.25	1.36±1.07	660	26.01
20	78.98	0.46±0.28	1.36±1.07	880	30.04
25	78.83	0.46±0.28	1.31±1.07	1100	33.12
30	79.91	0.46±0.27	1.31±1.08	1320	41.04
35	80.83	0.46±0.27	1.32±1.10	1540	48.87
40	80.99	0.46±0.27	1.30±1.13	1760	57.60
45	80.68	0.46±0.27	1.29±1.09	1980	67.78
50	81.45	0.45±0.23	1.26±1.00	2200	77.44
100	**85.32**	**0.44±0.22**	**1.21±1.00**	4400	240.61

required to locate the vehicle when using a larger $\mathcal{A}_{raw}$.

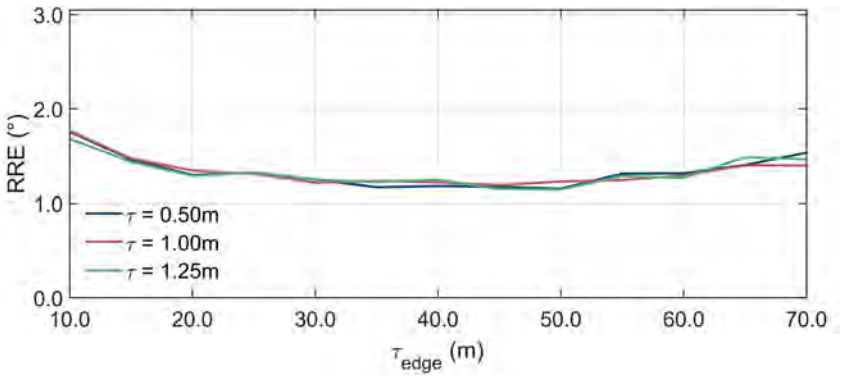
To investigate the influences of τ_{edge} and τ on TripletLoc, we set $k = 25$ and change values of τ_{edge} and τ, as shown in Fig. 5.9. Intuitively, when τ becomes smaller, the resolution of $Des^d_{v_j}$ is smaller, which can improve the descriptive capability of the vertex descriptor. As a result, more inlier correspondences might be involved in $\mathcal{A}_{raw}$, providing more correct constraints during pose optimization. Differently, when τ_{edge} becomes smaller, the semantic graph will become more sparse, leading to a decline in the number of triplets extracted from the graph. So Des_{v_j} will also become more sparse (i.e., most elements will be 0), leading to the degradation of vertex matching performance. However, when τ_{edge} is larger, the localization performance will also degrade. This is because only partial instances in the reference map can be observed by the single query scan. When τ_{edge} is too large, edges that connect instances distributed along the margin of the query scan, are more different from edges that



(a) Success rate with different τ_{edge} and τ



(b) RTE with different τ_{edge} and τ



(c) RRE with different τ_{edge} and τ

Figure 5.9: Global localization performance with different τ_{edge} and τ for query sequence Roundabout03. All experimental results are obtained when $\alpha = 5°$ and $k = 25$.

connect the same instances in the reference map, as shown in Fig. 5.4. Empirically, an appropriate τ_{edge} and small τ can generally result in a better performance.

5.5 Conclusion

In this work, 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. A novel RSN map is proposed to provide a prior rotation constraint for pose estimation. Alone with this rotation constraint, graph-theoretic outlier pruning is used to select inlier correspondences for the robust 6-DoF pose estimation. Extensive experiments in diverse and large-scale urban environments demonstrate that TripletLoc is highly competitive to state-of-the-art methods. Quantitative and qualitative results show the robustness, accuracy, memory and real-time efficiency of TripletLoc.

Chapter 6

Multi-Robot Localization using Lightweight Semantics: A Preliminary Study

6.1 Introduction

Multi-robot systems offer several advantages over single-robot systems across a variety of applications. For instance, they can significantly enhance efficiency in surveying environments at disaster sites, such as those affected by earthquakes or industrial accidents, where time is of the essence. In addition, multi-robot systems can accelerate the completion of higher-level tasks, such as goods transportation in warehouses and cleaning operations [138, 139]. In autonomous driving, the interaction between different vehicles is crucial to ensure that fleets can collaborate effectively for safe and efficient transportation. To achieve such high-level autonomy, a robot must first determine its position relative to the environment and other robots. This positional awareness enables the robot to perform downstream tasks such as path planning and navigation. Traditionally, exchanging GNSS coordinates has provided accurate inter-

robot localization. However, GNSS performance can be compromised due to signal occlusions and multipath effects in urban canyons or underground environments.

Recent research has explored the use of on-board measurements to localize robots relative to a prior database or map, demonstrating potential for global localization without relying on GNSS [125, 128, 12, 124]. However, such prior databases or maps are not always available for certain multi-robot tasks, such as environmental surveying or exploration. As an alternative, multi-robot Simultaneous Localization and Mapping (SLAM) has been employed to eliminate the need for GNSS or prior maps/databases for inter-robot localization in earlier works [7, 140, 141, 142, 143, 144]. Typically, sensor measurements from different robots are sent to a server or lead robot, which are then used to update maps and localization. However, as the number of robots increases, the volume of LiDAR or image data that needs to be transmitted also increases, which might be inefficient when communication bandwidth is limited. To address this issue, recent works have employed lightweight map representations (e.g., segments, Gaussian Mixture Models (GMM), semantics, objects) for multi-robot SLAM [138, 145, 146, 147]. These methods have shown promise in reducing the required communication bandwidth in multi-robot systems. Some of these approaches rely on submap-to-submap based place recognition methods for inter-robot localization, which separates the visited environment as multiple submaps. This approach might not ensure sufficient overlap between the query keyframes and the reference keyframes across different robots sometimes.

In this study, we consider the visited environment for each robot in a team as a continuous instance-level map rather than multiple submaps. The proposed multi-robot system builds on our previous work, `Tripletloc`(see 5), which provides pairwise robot localization results between robots. Since the instance-level map accumulates data from consecutive keyframes, the overlap between this map and query keyframes from other robots is typically much larger. While `Tripletloc` was initially designed for single-robot localization, it only provide localization result for the single query

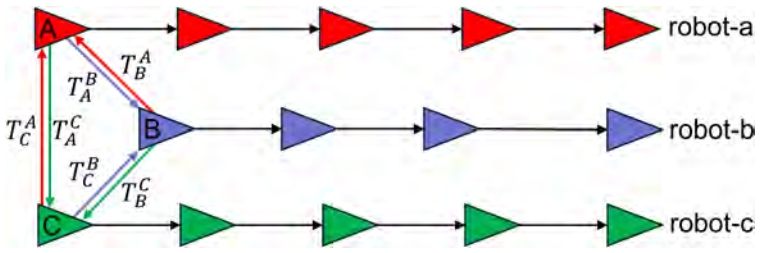


Figure 6.1: The example of relative poses between local reference frames of different robots for a team with three robots.

scan. To enhance the robustness of inter-robot localization, we use multiple one-shot localization results between each pair of robots, further optimizing them through pose averaging. Multi-robot alignment is then achieved by searching for the shortest transformation chains among all currently available transformation chains.

6.2 Problem Formulation

In a multi-robot system, inter-robot localization aims to determine each robot's location respective to a shared reference frame. This shared reference frame is usually chosen as the local reference frame of one robot from the robot team. Assuming we have three robots, robot-a, robot-b and robot-c, with local reference frames A, B, and C. Then the inter-robot localization aims to obtain relative poses between these local reference frames, i.e., $\boldsymbol{T}_B^A$(or $\boldsymbol{T}_A^B$), $\boldsymbol{T}_C^A$(or $\boldsymbol{T}_A^C$), and $\boldsymbol{T}_C^B$(or $\boldsymbol{T}_B^C$), to align them to a shared reference frame, as shown in Fig. 6.1. Where $\boldsymbol{T}_B^A \in SE(3)$ is the ground truth of the relative pose between A and B, using A as the reference frame. Theoretically, having two of the above poses (e.g., $\boldsymbol{T}_B^A$ and $\boldsymbol{T}_C^A$, or $\boldsymbol{T}_B^A$ and $\boldsymbol{T}_C^B$, or $\boldsymbol{T}_C^B$ and $\boldsymbol{T}_C^A$, where A is chosen as the shared reference frame) can completely align the three local reference frames into the shared reference frame.

6.2.1 One-shot Localization between Pairwise Robots

First, let us see how to obtain an estimation for the relative pose between the local reference frames of robot a and b. Given two instances map M^{a_i} and M^{b_j}, two new instance-level keyframes f^{a_i} and f^{b_j}, two trajectories $Traj^{a_i}$ and $Traj^{b_j}$ from robot a and b, respectively. M^{a_i} is the accumulated instance map using keyframes $\{f^{a_1}, f^{a_2}, ..., f^{a_i}\}$ from robot a (see 5.3.1). $Traj^{a_i}$ is the accumulated odometry trajectory using keyframes $\{f^{a_1}, f^{a_2}, ..., f^{a_i}\}$ from robot a. For simplification, trajectory drift from odometry is not considered. Indeed, current LiDAR-inertial Odometry and LiDAR-inertial-Visual Odometry can provide trajectory estimation with very low drift even after travelling long distance [148, 149].

One-shot localization is then achieved using `Tripletloc`, where f^{a_i} and M^{b_j} are used as the query scan and the reference map, respectively. The global localization result is denoted as $\hat{\boldsymbol{T}}^B_{a_i}$, where a_i represents the query scan, and B refers to the reference frame. Although `Tripletloc` can provide global localization result with competitive success rate (see 5.4.2), it still might provide incorrect results sometimes. Therefore, we need to determine whether a localization result is valid. We validate the results using two metrics: the ratio of the Maximum Clique (MCQ) size to the total number of associations, noted as λ_{MCQ}, and the projection error ϵ. Typically, a localization result is more likely valid when there are more inlier associations (i.e., larger Maximum Clique). Considering the instance number might be very different across different query keyframes, it is not easy to set a specific threshold based on the size of MCQ. We therefore normalized the size of MCQ by dividing the total association number. A valid localization result should satisfy the following condition:

$$\lambda_{MCQ} = \frac{|\mathcal{A}|}{|\mathcal{A}_{raw}|} \geq \dot{\lambda}, \tag{6.1}$$

where $\mathcal{A}$ and $\mathcal{A}_{raw}$ are MCQ and raw instance-to-instance associations (see Eq. 5.2), respectively. $\dot{\lambda}$ is a pre-defined threshold. To further reduce the risk of incorrect

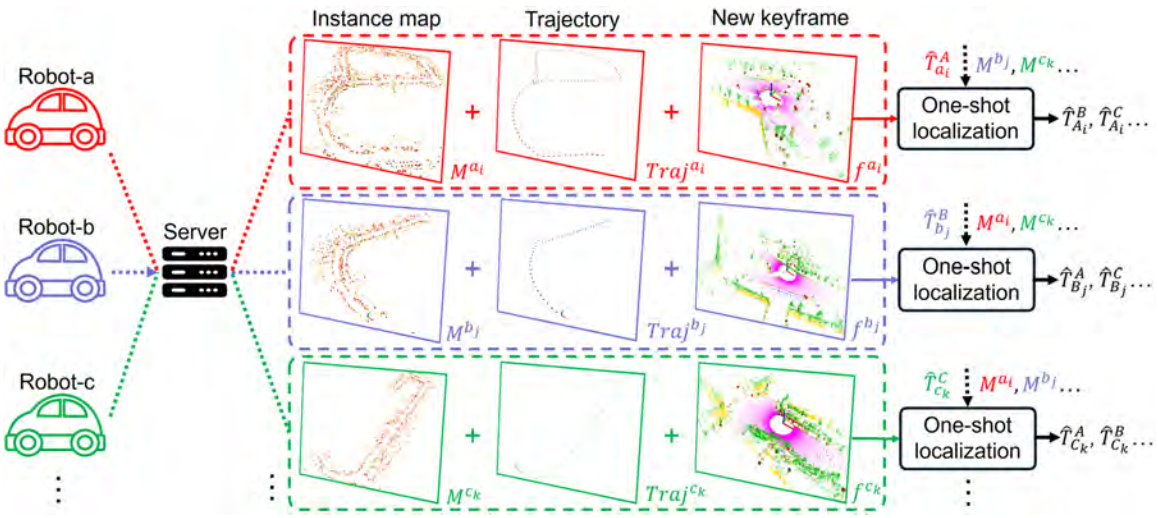


Figure 6.2: Demonstration of one-shot localization between pairwise robots in a robot team. New keyframes and their odometry poses are uploaded to the server from different robots, which are then used to estimate the relative pose between different local reference frames. Noted that the colored point cloud in the new keyframe is included for better presentation purposes and is not required to be uploaded to the server in the system.

localization, we also account for the projection error ϵ between matched instances.

$$\epsilon = \sum_{j,t \in \mathcal{B}_a^b} \left\| \overline{v}_t^b - \hat{\boldsymbol{R}} \overline{v}_j^a - \hat{\boldsymbol{t}} \right\|_2, \tag{6.2}$$

Where $\mathcal{B}_a^b$ (more details can be found in 5.3.3) represents vertex pairs determined by the above MCQ, $\mathcal{A}$ (see Eq. 6.1). $\hat{\boldsymbol{R}}$ and $\hat{\boldsymbol{t}}$ are the rotation and translation components of $\hat{\boldsymbol{T}}_{a_i}^B$, respectively. In the absence of measurement noise, when $\hat{\boldsymbol{T}}_{a_i}^B$ yields a perfect localization result, the projection error ϵ equals to 0. Therefore, a valid localization result should also satisfy the following condition:

$$\epsilon \leq \dot{\epsilon}, \tag{6.3}$$

where $\dot{\epsilon}$ is a pre-defined threshold for the projection error. Therefore, a localization result is considered valid when both Eq. 6.1 and Eq. 6.3 are met.

Based on the trajectory of robot-a, $Traj^{a_i}$, we can easily obtain the pose of keyframe f^{a_i} respective to its own local reference frame A, namely $\hat{\boldsymbol{T}}^A_{a_i}$. By integrating $\hat{\boldsymbol{T}}^B_{a_i}$ and $\hat{\boldsymbol{T}}^A_{a_i}$, we can have the alignment estimation between local reference frame A and B:

$$\hat{\boldsymbol{T}}^B_{A_i} \triangleq \hat{\boldsymbol{T}}^B_{a_i}(\hat{\boldsymbol{T}}^A_{a_i})^{-1}, \tag{6.4}$$

where A_i indicates that this alignment estimation is based on the localization result from `TripletLoc` using query keyframe f^{a_i}. Similarly, we can also get $\hat{\boldsymbol{T}}^A_{B_j}$ as follow:

$$\hat{\boldsymbol{T}}^A_{B_j} \triangleq \hat{\boldsymbol{T}}^A_{b_j}(\hat{\boldsymbol{T}}^B_{b_j})^{-1}, \tag{6.5}$$

For alignment estimations between other local reference frames, i.e., $\hat{\boldsymbol{T}}^C_{A_i}$, $\hat{\boldsymbol{T}}^A_{C_k}$, $\hat{\boldsymbol{T}}^B_{C_k}$, and $\hat{\boldsymbol{T}}^C_{B_j}$, can be also obtained by the same way. Demonstration of one-shot localization between pairwise robots in a robot team can be found in Fig. 6.2.

6.2.2 Optimization Using Pose Averaging

From Eq. 6.4, we see that each new keyframe $f^{a_{i-1}}$ or f^{a_i} can provide an alignment estimation $\hat{\boldsymbol{T}}^B_{A_{i-1}}$ or $\hat{\boldsymbol{T}}^B_{A_i}$ between local reference frame A and B. When two alignment estimations $\hat{\boldsymbol{T}}^B_{A_{i-1}}$ and $\hat{\boldsymbol{T}}^B_{A_i}$ are both valid inlier, they are expected to be in mutual agreement [150, 146]. Therefore, integrating multiple alignment estimations to get a final alignment estimation is usually more reliable than only using single estimation. Similar to [146], we also optimize the result using multi alignment estimations. Differently, we explicitly use estimations for both $\hat{\boldsymbol{T}}^B_A$ and $\hat{\boldsymbol{T}}^A_B$ for the reliable alignment estimation instead of only use one of them. Assume we have n and m valid alignment estimations for $\hat{\boldsymbol{T}}^B_A$ and $\hat{\boldsymbol{T}}^A_B$, noted as $\{\hat{\boldsymbol{T}}^B_{A_i}\}_n$ and $\{\hat{\boldsymbol{T}}^A_{B_j}\}_m$, respectively, as shown in Fig. 6.3. Let $\boldsymbol{S}^B_A$ *or* $\boldsymbol{S}^A_B = \{\hat{\boldsymbol{T}}^B_{A_i}\}_n \cup \{\hat{\boldsymbol{T}}^A_{B_j}\}_m$ be the set of current valid alignments between reference frame A and B. For arbitrary $\hat{\boldsymbol{T}}^A_{B_j}$ and $(\hat{\boldsymbol{T}}^B_{A_i})^{-1}$, we can have the following relation:

$$\hat{\boldsymbol{T}}^A_{B_j} \approx (\hat{\boldsymbol{T}}^B_{A_i})^{-1}, \tag{6.6}$$

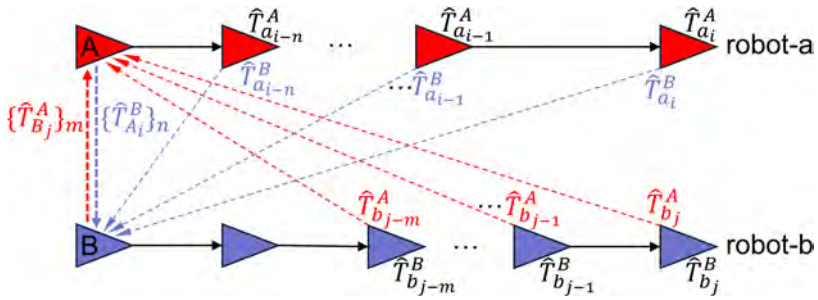


Figure 6.3: Multiple alignment estimations between frame A and B. These estimations are then used to obtain a reliable alignment estimation using pose averaging.

Then the reliable alignment result is achieved by solving the following *pose averaging* problem:

$$\hat{\boldsymbol{T}}_B^A \in \arg\min_{\boldsymbol{T}\in SE(3)} \sum_{\hat{\boldsymbol{T}}_{B_j}^A \in \{\hat{\boldsymbol{T}}_{B_j}^A\}_m} \rho\left[\left\|\boldsymbol{T} \oplus \hat{\boldsymbol{T}}_{B_j}^A\right\|\right] + \sum_{\hat{\boldsymbol{T}}_{A_i}^B \in \{\hat{\boldsymbol{T}}_{A_i}^B\}_n} \rho\left[\left\|\boldsymbol{T} \oplus \left(\hat{\boldsymbol{T}}_{A_i}^B\right)^{-1}\right\|\right], \textit{or}$$

$$\hat{\boldsymbol{T}}_A^B \in \arg\min_{\boldsymbol{T}\in SE(3)} \sum_{\hat{\boldsymbol{T}}_{A_i}^B \in \{\hat{\boldsymbol{T}}_{A_i}^B\}_n} \rho\left[\left\|\boldsymbol{T} \oplus \hat{\boldsymbol{T}}_{A_i}^B\right\|\right] + \sum_{\hat{\boldsymbol{T}}_{B_j}^A \in \{\hat{\boldsymbol{T}}_{B_j}^A\}_m} \rho\left[\left\|\boldsymbol{T} \oplus \left(\hat{\boldsymbol{T}}_{B_j}^A\right)^{-1}\right\|\right] \quad (6.7)$$

where $\left\|\boldsymbol{T} \oplus \hat{\boldsymbol{T}}_{B_j}^A\right\|$ is the residual measurement, i.e., the geodestic distance between two poses, $\rho[\cdot]$ is the Truncated Least Squares (TSL) robust cost function [131]. Similarly, the reliable alignment results for $\hat{\boldsymbol{T}}_A^C$, $\hat{\boldsymbol{T}}_C^A$, $\hat{\boldsymbol{T}}_B^C$, and $\hat{\boldsymbol{T}}_C^B$ can be obtained by the same way. The Eq. 6.7 can be solved by using Graduated Non-Convexity (GNC) [131]. Specifically, we use the GNC implementation in GTSAM [136] with a fixed diagonal covariance matrix for all the residual measures.

6.2.3 Multi Robots Alignment Using Shortest Transformation Chain

Based on Eq. 6.7, we can obtain the optimized pairwise alignment estimations between different robots within a multi-robot system. Once these estimations are determined, it becomes necessary to establish a shared reference frame to bring all robots into a

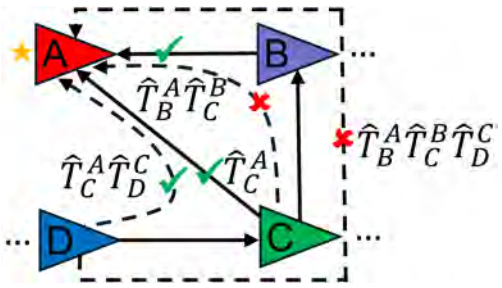


Figure 6.4: Example of aligning different robots to a shared reference frame A using the shortest transformation chains.

shared reference frame for global alignment. In principle, this shared reference frame can be chosen arbitrarily from any of the local reference frames of the robots in the team. As an example, and without loss of generality, we select the local reference frame A as the shared reference for multi robots alignment. For a team with three robots (a, b, and c), the goal of multi robots alignment is to align both robot b and c in reference frame A. Suppose we have already obtained the optimized relative pose between robot a and b, corresponding to the transformation between frames A and B. In this case, only reference frame C needs to be aligned. There are two possible transformation chains that can be used to achieve this alignment. The first chain is to directly use the transformation $\hat{T}_C^A$. The second chain is an indirect chain, using the composition of transformations $\hat{T}_B^A\hat{T}_C^B$, which sequentially aligns robot c through the intermediate frame B. When both $\hat{T}_C^A$ and $\hat{T}_B^A\hat{T}_C^B$ are available, we prefer to use $\hat{T}_C^A$ because it minimizes the accumulated localization error. The indirect chain $\hat{T}_B^A\hat{T}_C^B$ suffers from compounded errors introduced by the sequential application of two separate transformations, $\hat{T}_B^A$ and $\hat{T}_C^B$, as illustrated in Fig. 6.4. When only $\hat{T}_B^A\hat{T}_C^B$ is available, we align robot c using $\hat{T}_B^A\hat{T}_C^B$. This hierarchical approach is scalable and can be generalized to teams with more than three robots. For any number of robots, once a shared reference frame is selected, each robot is aligned by following the shortest available transformation chain to the reference frame, minimizing the error propagation due to multiple transformations.

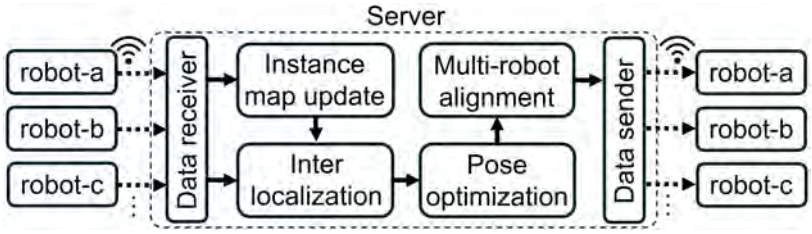


Figure 6.5: The overview of the proposed centralized multi-robot localization system.

6.3 System Overview

We design our system using a centralized architecture, where all the robots in the team upload their data to a central server. This server is responsible for performing inter-localization between the robots and subsequently distributing the results back to the team. As outlined in Section 6.2.1, to compute an alignment estimation (e.g., $\hat{\boldsymbol{T}}_{A_i}^{B}$), three key elements are required: a reference instance map M^{b_j}, semantic instances from keyframe f^{a_i}, and the pose of keyframe f^{a_i} with respect to the reference frame A (i.e., $\hat{\boldsymbol{T}}_{a_i}^{A}$). Since the reference instance map M^{b_j} can be incrementally constructed from consecutive keyframes, robots do not need to upload the entire map with every update. Instead, they only need to transmit the semantic instances detected in the current keyframe f^{a_i} and the corresponding pose $\hat{\boldsymbol{T}}_{a_i}^{A}$. This approach significantly reduces the communication load. Once the server receives a new keyframe from any robot, it proceeds to update that robot's instance map. The server then performs pairwise localization (see 6.2.1) by treating the newly received keyframe as the query frame and using the instance maps from the other robots as reference maps. As more keyframes are uploaded by different robots, the server accumulates multiple pairwise alignment estimations between different pairs of robots. These pairwise estimations are then used for the pose-averaging optimization process (see 6.2.2). After obtaining the optimized pairwise alignment estimations, the system proceeds to align all robots to a shared reference frame. We use the shortest transformation chains between robots to minimize error propagation (see 6.2.3). The overview of the system architecture,

including the process flow from data uploading to multi-robot alignment, can be found in Fig. 6.5.

6.4 Experiment

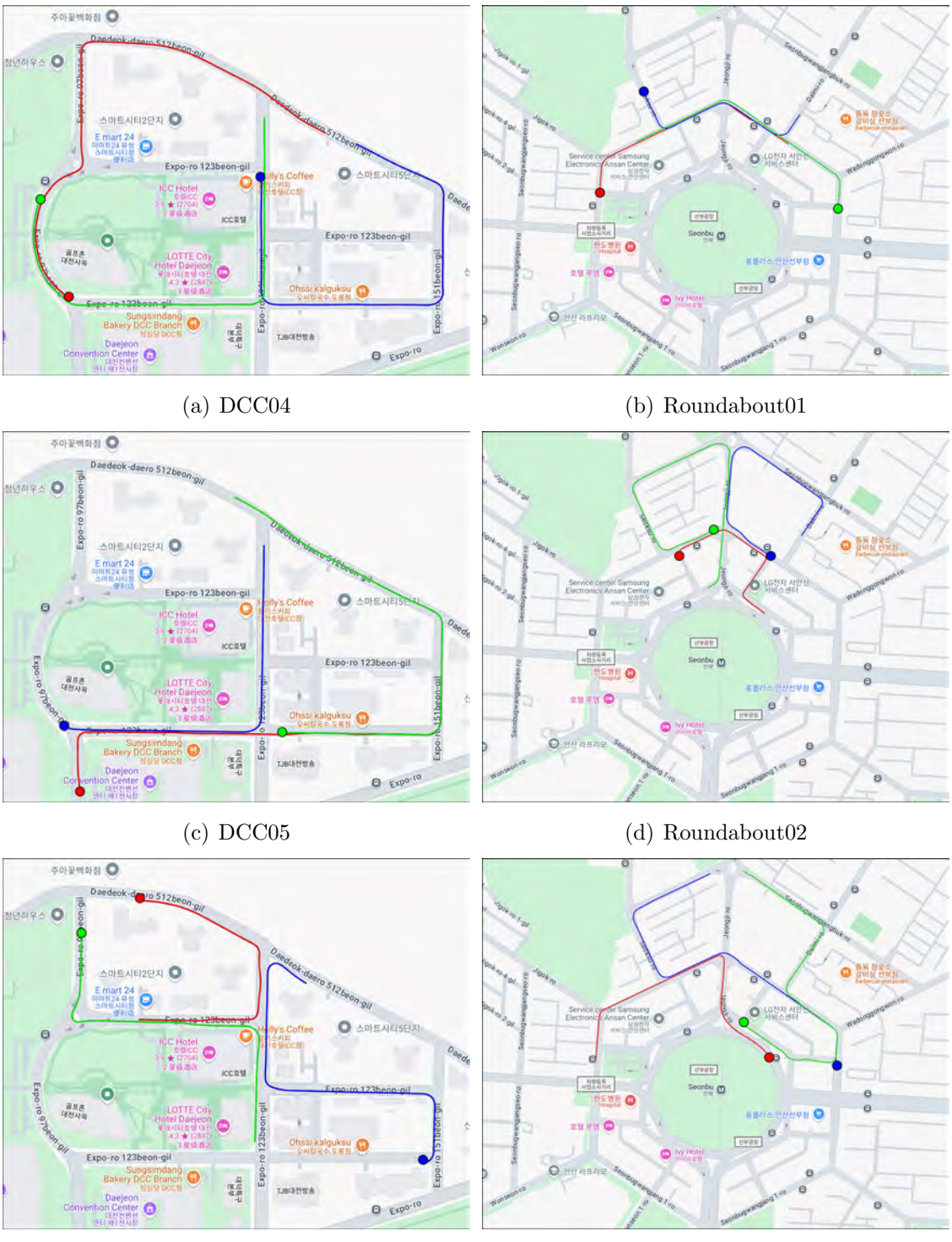
6.4.1 Dataset and Setup

We evaluate the proposed system using scenario DCC and Roundabout in the HeLiPR Dataset [137]. For each sequence, we select three distinct segments with approximately equal travel distances to simulate a three-robot collaborative scenario. Each segment is required to have overlaps with at least one of the other segments to facilitate inter-localization. Additionally, to ensure that different robots do not share identical observations (i.e., the same frames), the overlapping frames across segments must be captured at different times. This simulates the real-world scenario where overlapping areas are visited by different robots at separate intervals. Distribution of the selected segments can be seen in Fig. 6.6.

6.4.2 One-shot Localization Analysis

Evaluation of Valid Localization Determination

The determination of valid localization results relies on two key thresholds, $\dot{\lambda}$ and $\dot{\epsilon}$. We evaluate how these thresholds influence the performance of localization determination using the F_1-score, a common metric in classification tasks (see 3.3.2 for more details on the F_1-score). For each localization result (i.e., $\hat{\boldsymbol{T}}^A_{b_j}$, $\hat{\boldsymbol{T}}^A_{c_k}$, $\hat{\boldsymbol{T}}^B_{a_i}$, $\hat{\boldsymbol{T}}^B_{c_k}$, $\hat{\boldsymbol{T}}^C_{a_i}$, and $\hat{\boldsymbol{T}}^C_{b_j}$), we calculate the Relative Translation Error (RTE) and Relative Rotation Error (RRE). A valid localization result is termed as True Positive (TP) when satisfying RTE $\leq$ 7.5m and RRE $\leq 10^\circ$; otherwise, it is classified as a False



(a) DCC04 (b) Roundabout01

(c) DCC05 (d) Roundabout02

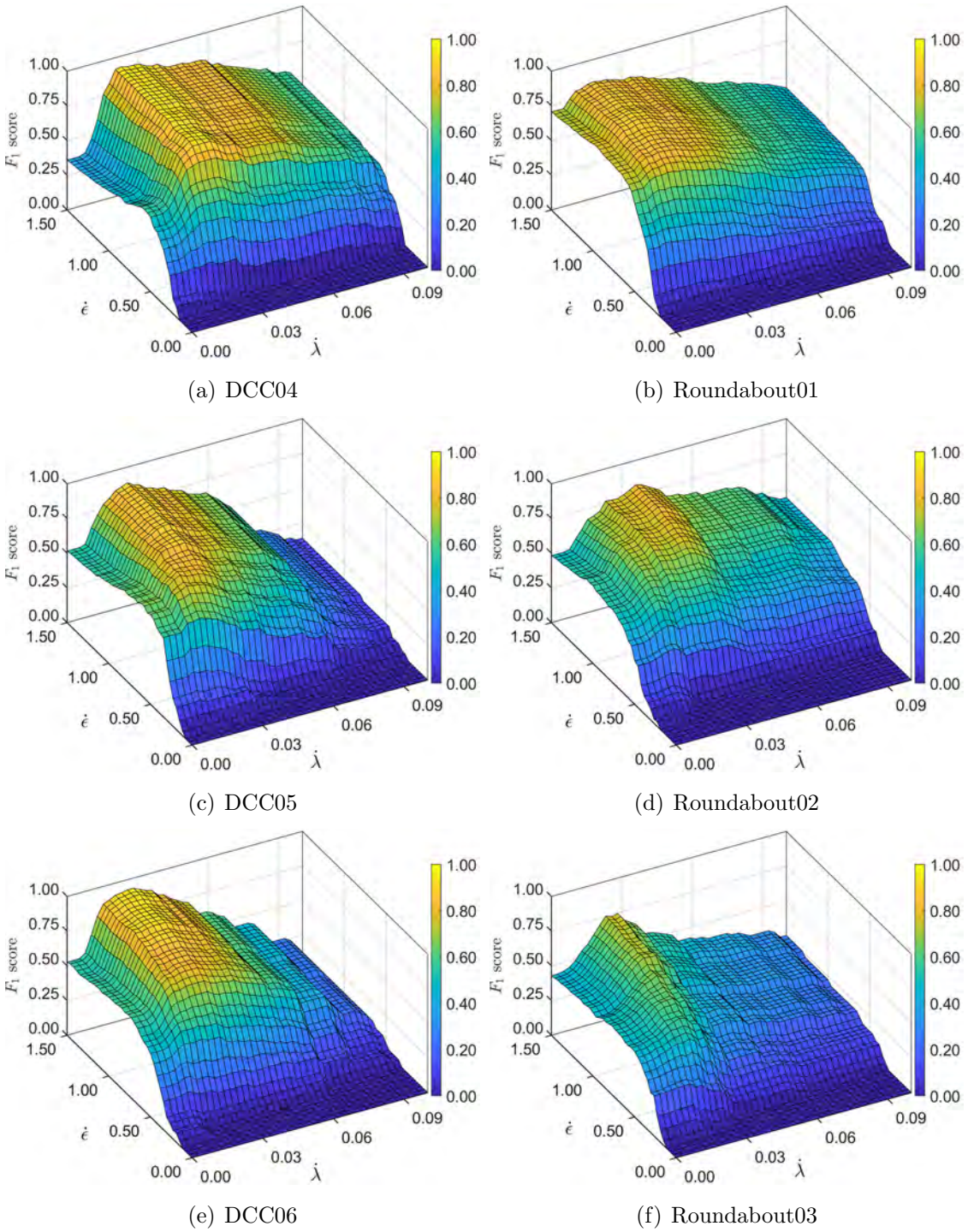
(e) DCC06 (f) Roundabout03

Figure 6.6: Selected segments for multi-robot localization evaluation in the DCC and Roundabout scenarios. Different colors represent the travel routes of each robot, with larger dots indicating the starting points of the routes.

Positive (FP). Similarly, an invalid localization result that meets these conditions is classified as a False Negative (FN), and if it does not, it is a True Negative (TN). We then vary the values of $(\dot{\lambda}, \dot{\epsilon})$ and calculate the corresponding F_1-score. As shown in Fig. 6.7, the determination performance is more sensitive to $\dot{\lambda}$ than to $\dot{\epsilon}$. When $\dot{\lambda}$ is kept unchanged, the F_1-score remains relatively stable for $\dot{\epsilon} \in [0.8, 1.0]$, but begins to decrease when $\dot{\epsilon} < 0.8$, due to stricter criteria for valid localization results. For $\dot{\lambda}$, there is no clear range where the F_1-score remains stable. Instead, the score increases initially but decreases as $\dot{\lambda}$ continues to grow. Generally, a higher $\dot{\lambda}$ and lower $\dot{\epsilon}$ can filter out more localization results and retain only higher-quality ones, but this may result in too few valid results for the subsequent pose-averaging optimization. On the other hand, a lower $\dot{\lambda}$ and higher $\dot{\epsilon}$ ensure a sufficient number of valid results but may introduce more outliers, complicating the optimization process. Empirically, we set $\dot{\epsilon}$=0.8 and $\dot{\lambda}$=0.03 in our system, achieving F_1-scores of 0.83 and 0.67 in the DCC and Roundabout scenarios, respectively.

Localization Accuracy Evaluation

For each sequence, we calculate the average, standard deviation, maximum, and minimum for all the true positives of one-shot localization results. As shown in Tab. 6.1, a sub-meter localization accuracy is achieved by the one-shot localization in average. The overall rotation accuracy for one-shot localization is less than 2°. We also compute the RTE and RRE for each alignment estimation between local reference frames, i.e., $\hat{\boldsymbol{T}}^A_{B_j}$, $\hat{\boldsymbol{T}}^A_{C_k}$, $\hat{\boldsymbol{T}}^B_{A_i}$, $\hat{\boldsymbol{T}}^B_{C_k}$, $\hat{\boldsymbol{T}}^C_{A_i}$, and $\hat{\boldsymbol{T}}^C_{B_j}$. In addition, the euclidean distance $dis_{odo} = \sqrt{x^2 + y^2 + z^2}$ is calculated for each corresponding odometry pose, where x, y, and z are the translations for $\hat{\boldsymbol{T}}^A_{a_i}$, $\hat{\boldsymbol{T}}^B_{b_j}$, and $\hat{\boldsymbol{T}}^C_{c_k}$. As shown in Tab. 6.2, the RTE for the alignment estimations between local reference frames is much larger than the RTE for the localization results from `TripletLoc` (see Tab. 6.1), although the RRE remains similar. This increase in RTE is primarily due to the compounded localization error from `TripletLoc` and the odometry poses, as described in Eq. 6.4



(a) DCC04

(b) Roundabout01

(c) DCC05

(d) Roundabout02

(e) DCC06

(f) Roundabout03

Figure 6.7: The F_1 score for the valid one-shot localization result determination across different values of $(\dot{\lambda}, \dot{\epsilon})$.

Table 6.1: The average, standard deviation, maximum, and minimum for the true positive one-shot localization results.

Sequence	RTE (m)			RRE (°)		
	ave±std	Max	Min	ave	Max	Min
DCC04	0.8155±0.2222	1.33	0.44	0.8119±0.3213	1.63	0.33
DCC05	**0.2152±0.1558**	0.86	0.06	**0.7541±0.4306**	2.00	0.17
DCC06	0.6135±0.2025	1.06	0.30	0.8544±0.4349	2.43	0.21
Roundabout01	0.6161±0.2524	1.13	0.17	0.8924±0.6438	3.35	0.13
Roundabout02	0.9096±0.4728	2.69	0.36	2.4192±1.3342	5.54	0.28
Roundabout03	0.3934±0.2487	1.06	0.07	1.1830±0.7172	2.82	0.33

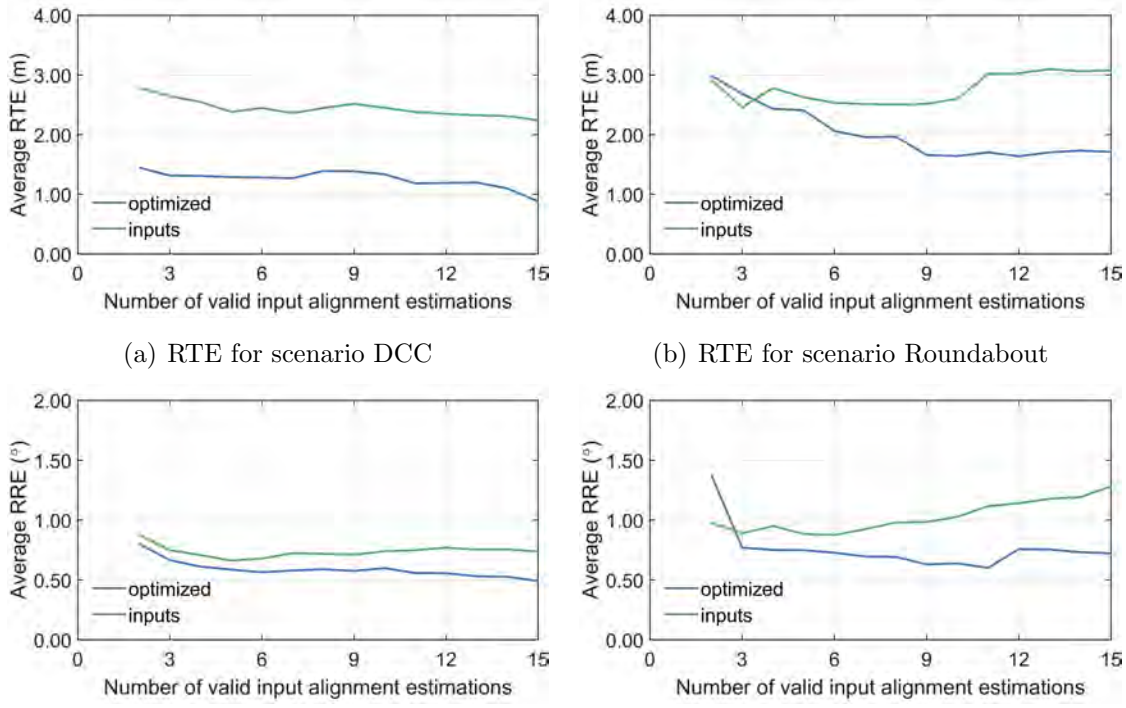
Table 6.2: The average, standard deviation, maximum, and minimum values for the true positive alignment estimations between local reference frames, along with the corresponding euclidean distances.

Sequence	RTE (m)			RRE (°)			dis_{odo}		
	ave±std	Max	Min	ave±std	Max	Min	ave	Max	Min
DCC04	2.29±1.73	6.82	0.60	0.81±0.32	1.63	0.33	216.49	351.90	62.29
DCC05	2.58±2.18	9.06	0.08	**0.75±0.43**	2.00	0.17	296.30	550.42	62.02
DCC06	3.23±2.63	11.70	0.23	0.85±0.43	2.43	0.21	279.93	370.37	176.46
Roundabout01	3.47±2.97	17.32	0.28	0.89± 0.64	3.34	0.13	332.35	487.81	178.37
Roundabout02	**2.10±1.41**	4.90	0.53	2.42±1.33	5.54	0.28	**93.29**	202.13	26.00
Roundabout03	3.29±2.40	9.12	0.55	1.18±0.72	2.82	0.33	199.81	272.27	51.41

and Eq. 6.5. In general, as a localization result from `TripletLoc` is obtained farther from the origin of its local reference frame (i.e., larger dis_{odo}), the error of alignment estimation between two local reference frames increases.

6.4.3 Multi Robot Alignment Analysis

As described in 6.2.2, we employ pose averaging optimization using multiple alignment estimations to obtain a final, reliable alignment result. The effectiveness of this



(a) RTE for scenario DCC

(b) RTE for scenario Roundabout

(c) RRE for scenario DCC

(d) RRE for scenario Roundabout

Figure 6.8: The average RTE(m) and RRE(°) for optimized alignment and the corresponding valid input alignment estimations in scenario DCC and Roundabout.

.

optimization is assessed by comparing the RTE and RRE of the optimized alignment with the RTE and RRE of the valid input alignment estimations for the optimization. For example, based on Eq. 6.7, we first compute the RTE and RRE for the optimized alignment between frames A and B, i.e., $RTE(\hat{\boldsymbol{T}}_A^B, \boldsymbol{T}_A^B)$ and $RRE(\hat{\boldsymbol{T}}_A^B, \boldsymbol{T}_A^B)$. We then calculate the average RTE and RRE for the corresponding valid input alignment estimations (i.e., $\boldsymbol{S}_A^B = \{\hat{\boldsymbol{T}}_{A_i}^B\}_n \cup \{\hat{\boldsymbol{T}}_{B_j}^A\}_m$) that are used for the optimization as following:

$$RTE(\boldsymbol{S}_A^B) = \frac{1}{n+m}\Big(\sum_{\hat{\boldsymbol{T}}_{A_i}^B \in \{\hat{\boldsymbol{T}}_{A_i}^B\}_n} RTE(\hat{\boldsymbol{T}}_{A_i}^B, \boldsymbol{T}_A^B) + \sum_{\hat{\boldsymbol{T}}_{B_j}^A \in \{\hat{\boldsymbol{T}}_{B_j}^A\}_m} RTE(\hat{\boldsymbol{T}}_{B_j}^A, \boldsymbol{T}_B^A)\Big) \quad (6.8)$$

$$RRE(\boldsymbol{S}_A^B) = \frac{1}{n+m}\Big(\sum_{\hat{\boldsymbol{T}}_{A_i}^B \in \{\hat{\boldsymbol{T}}_{A_i}^B\}_n} RRE(\hat{\boldsymbol{T}}_{A_i}^B, \boldsymbol{T}_A^B) + \sum_{\hat{\boldsymbol{T}}_{B_j}^A \in \{\hat{\boldsymbol{T}}_{B_j}^A\}_m} RRE(\hat{\boldsymbol{T}}_{B_j}^A, \boldsymbol{T}_B^A)\Big) \quad (6.9)$$

Results for other reference frame pairs (i.e.,A-C, and B-C) can be obtained by the same manner. Theoretically, the optimization becomes more reliable when there are more inputs (i.e., larger $n+m$). To investigate this, we vary $n+m$ and and calculate the RTE and RRE for both the optimized alignment and the corresponding valid input alignment estimations for all reference frame pairs (i.e., A-B, A-C, and B-C). We then compute the averages of these RTE and RRE results across different sequences within the same scenario (i.e., DCC and Roundabout). As shown in Fig. 6.8, the RTE and RRE of the optimized alignment generally decrease as more valid alignment estimations are used for the optimization. Moreover, the average RTE and RRE for the optimized alignment results are consistently smaller than those for the valid input alignments before optimization, except when $n+m=2$ and $n+m=3$ in the Roundabout scenario. Overall, the pose averaging optimization provides more reliable final alignment estimations compared to the original valid inputs, reaching RTE<1.5m and RRE<1.0° in scenario DCC, and RTE<2.0m with RRE<1.0° in scenario Roundabout. Using these reliable alignment results, robot a and b are then aligned to the shared reference frame A via shortest chain searching. Although the optimization tends to be more reliable with larger value of $n+m$, it also requires more valid alignment estimations, which in turn demands larger overlap between the areas visited by different robots. This may result in a longer time to successfully align multiple robots or even failure if there is insufficient overlap. Examples of the concatenated point could in the shared reference frame A can be found in Fig. 6.9 and Fig. 6.10, where $n+m$ is set as 6 empirically. The registration quality varies across different overlap areas. Generally, the registration quality improves when more valid localization results from `TripletLoc` (i.e., $\hat{\boldsymbol{T}}^A_{b_j}$, $\hat{\boldsymbol{T}}^A_{c_k}$, $\hat{\boldsymbol{T}}^B_{a_i}$, $\hat{\boldsymbol{T}}^B_{c_k}$, $\hat{\boldsymbol{T}}^C_{a_i}$, and $\hat{\boldsymbol{T}}^C_{b_j}$) fall within the overlap area, such as overlap-1 and 3 in Fig. 6.9, as well as overlap-2 and 3 in Fig. 6.10. Conversely, registration quality declines in areas with fewer valid localization results, such as overlap-2 in Fig.6.9 and overlap-1 in Fig.6.10.

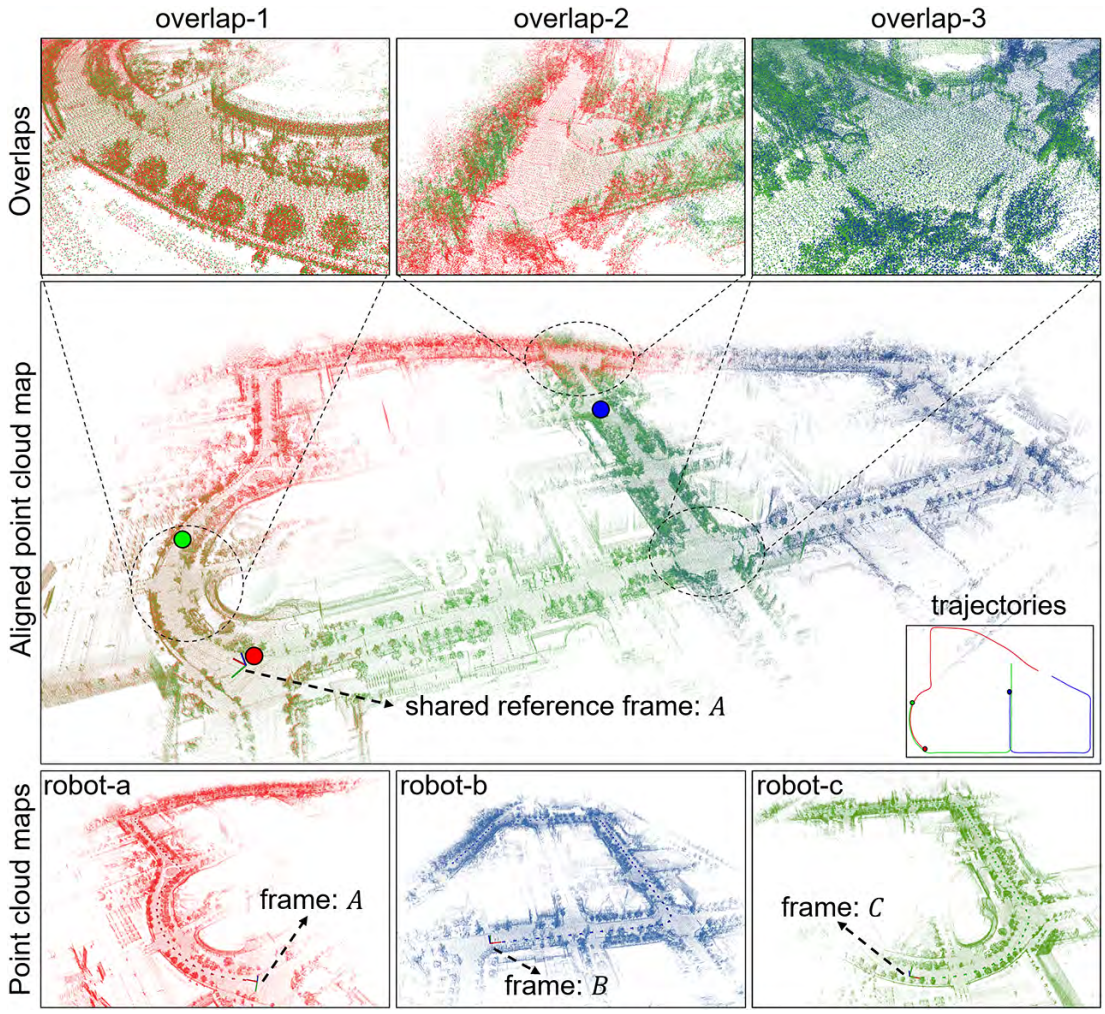


Figure 6.9: The concatenated point could map in the shared reference frame A for sequence DCC04, where $n + m$ is set as 6.

6.4.4 Communication Efficiency Analysis

To demonstrate the communication efficiency of our system, we calculate the average number of instances and 3D points per keyframe. The odometry pose is excluded from this calculation since it must be transmitted regardless of whether semantic instances or raw point clouds are used for inter-localization. As shown in Tab. 6.3, each keyframe contains an average of 66 instances. For each instance, the xyz coordinates and the corresponding semantic label are transmitted to the server, requiring

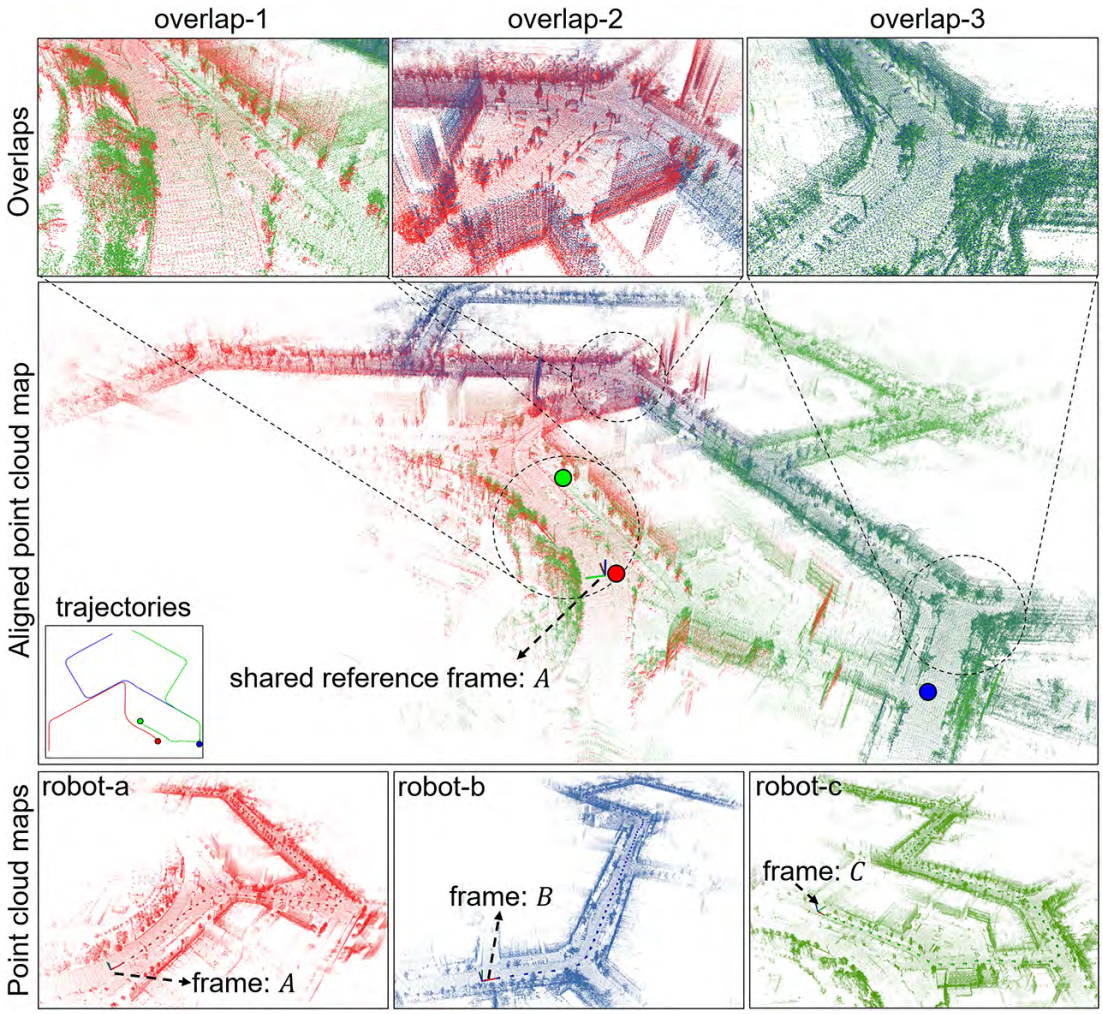


Figure 6.10: The concatenated point could map in the shared reference frame for sequence Roundabout03, where $n + m$ is set as 6.

approximately 1.85 KB of bandwidth per keyframe. In contrast, transmitting the raw point cloud, which contains an average of 117,397 points per keyframe, demands 2.82 MB of bandwidth. This means that transmitting the raw point cloud requires around 1,500 times more bandwidth than transmitting the semantic instances per keyframe. Thus, using lightweight semantic instances can significantly reduces the communication burden in multi-robot localization

Table 6.3: The average number of instances and raw point cloud points per keyframe, along with the corresponding memory consumption for each keyframe.

Sequence	Semantic instance		Raw pointcloud	
	ave instance num	memory (KB)	ave point num	memory (MB)
DCC04	70	1.96	114302	2.74
DCC05	74	2.07	118017	2.83
DCC06	88	2.46	117542	2.82
Roundabout01	55	1.54	119630	2.87
Roundabout02	55	1.54	117909	2.83
Roundabout03	55	1.54	116980	2.81
ave	66	1.85	117397	2.82

6.5 Conclusion

In this work, we develop a centralized multi-robot localization system, building upon our previous one-shot registration based global localization method, `TripletLoc`. We formulate the multi-robot localization as an optimization task and incorporating one-shot registration-based localization into the optimization process. Specifically, lightweight semantic instances from multiple robots are transmitted to a central server, which constructs instance maps for inter-robot localization. To reduce the risk of involving incorrect localization results, we propose a dual-metric validation strategy to confirm the validity of pairwise localization results from `TripletLoc`. These validated pairwise results are then optimized through pose averaging to obtain reliable alignment estimations between the local reference frames of different robots. A shortest transformation chain searching method is used to align all robots into a shared reference frame. Our preliminary results demonstrate the feasibility and effectiveness of the proposed system. It also shows promising potential for bandwidth-limited conditions, as the transmission of lightweight semantic instances significantly reduces data transfer between robots and the server.

Chapter 7

Conclusions, Limitations, and Future Work

This thesis aims to develop real-time and memory-efficient solutions for GNSS-free global localization in urban environments, with a focus on autonomous driving. We explore the use of OpenSreetMap and lightweight semantics to reduce the high memory demands of the 3D point clouds or 2D images based prior reference map/databases in large-scale scenarios. We mainly focus on reliable data association, outlier filtering, and robust pose estimation to achieve robust localization using these compact and efficient representations. The main contributions and research findings of this thesis are summarized below:

We propose SkyLoc, a novel global localization method that compares images from a sky-looking fish-eye camera with OpenStreetMap, utilizing a KLD-sampling-based particle filter to achieve real-time metric localization in dynamic traffic environments. The modality gap between fish-eye images and OpenStreetMap is bridged by leveraging junction types and building outline information present in both data sources. We present both quantitative and qualitative results, demonstrating good localization performance in challenging scenarios, along with high runtime efficiency, even on

embedded platforms.

In response to the sensitivity of vision-based localization to frequent condition changes in long-term scenarios, we shift to using ranging sensors for localization. To better understand the impact of seasonal and weather variations on localization performance, we conduct a comprehensive evaluation of range-sensing-based place recognition in large-scale urban environments. In addition, we propose a novel metric to assess the influence of matching thresholds on place recognition performance in long-term localization, offering valuable guidance for parameter setting to ensure more consistent results over time. Our results and findings provide fresh insights to the community and potential directions for future study.

To address the significant memory demands of storing additional point clouds or feature points for pose estimation in traditional place recognition-based localization methods, we propose a novel scan-to-scan method, Triplet-Graph, to achieve both place recognition and 6-DoF relative pose estimation with lightweight semantics. We convert LiDAR point clouds into semantic graphs and describe the vertices in the graphs with the proposed triplet-based histogram descriptor for vertex matching and pose estimation. These vertex descriptors are then selected and aggregated into a global descriptor to decide whether two places correspond to the same place according to a similarity score. Experimental results on the KITTI dataset demonstrate the competitive performance of our method compared to state-of-the-art approaches.

To further reduce the memory usage for the reference maps/databases, we propose TripletLoc, for fast and robust global registration of a single LiDAR scan to a large-scale reference map. The scan-to-map localization manner ensures that each semantic instance is stored only once in the reference map, unlike the scan-to-scan framework, which redundantly stores some of instances across multiple scans. We enhance the semantic histogram descriptor from Triplet-Graph to achieve more robust and effective instance-to-instance correspondences. In addition, we propose a novel Road Surface Normal (RSN) map to provide a prior rotational constraint, improving pose estima-

tion. To enhance robustness against outlier associations, we apply graph-theoretic outlier pruning and Graduated Non-Convexity. Extensive experiments in diverse and large-scale urban environments demonstrate that our TripletLoc is highly competitive to state-of-the-art methods, showing good robustness, accuracy, runtime efficiency, and memory efficiency.

Building on TripletLoc, we develop a centralized multi-robot localization system by formulating the problem as an optimization task and incorporating one-shot registration-based localization into the optimization process. A dual-metric validation strategy is proposed to confirm the validity of pairwise localization results from `TripletLoc`. The validated pairwise results are subsequently refined using pose averaging to achieve reliable alignment between local reference frames of different robots. A shortest transformation chain searching method is used to align all robots within a shared reference frame. Preliminary results confirm the system's feasibility and effectiveness, while also showcasing its strong potential in bandwidth-constrained environments.

Although our study demonstrates competitive performance and strong potential, several limitations remain. First, the absolute localization accuracy of SkyLoc is relatively low due to image distortion and the resolution of OpenStreetMap. This makes SkyLoc unsuitable for scenarios that require sub-meter localization accuracy. Second, Triplet-Graph, TripletLoc, and our multi-robot system currently rely solely on the geometric centroids of semantic instances, discarding other valuable geometric information such as shape, size, and orientation. Inconsistencies in semantic segmentation and instance clustering quality across different scans can result in shifts in the centroids of identical semantic instances, making association and pose estimation more challenging. Moreover, an instance-level map that relies solely on geometric centroids may not be sufficient for other tasks, such as obstacle avoidance. Third, our multi-robot localization system has only undergone preliminary evaluation. A more comprehensive study, including extensive testing in diverse environments and detailed comparisons with state-of-the-art multi-robot solutions, is needed to fully

assess its performance and identify areas for improvement.

In the future, we plan to explore new compact representations that capture more geometric details to improve the robustness of data association, enhance localization accuracy, and support additional tasks such as navigation and path planning. In addition, we intend to incorporate uncertainties in instance-to-instance associations, including those related to semantic segmentation and instance clustering, to achieve more robust outlier filtering and pose optimization. Furthermore, we aim to integrate camera and LiDAR data to develop a more flexible reference map capable of supporting different types of onboard sensors.

References

[1] Daniel J Fagnant and Kara Kockelman. Preparing a nation for autonomous vehicles: opportunities, barriers and policy recommendations. *Transportation Research Part A: Policy and Practice*, 77:167–181, 2015.

[2] Gourav Bathla, Kishor Bhadane, Rahul Kumar Singh, Rajneesh Kumar, Rajanikanth Aluvalu, Rajalakshmi Krishnamurthi, Adarsh Kumar, RN Thakur, and Shakila Basheer. Autonomous vehicles and intelligent automation: Applications, challenges, and opportunities. *Mobile Information Systems*, 2022(1):7632892, 2022.

[3] Asif Faisal, Md Kamruzzaman, Tan Yigitcanlar, and Graham Currie. Understanding autonomous vehicles. *Journal of transport and land use*, 12(1):45–72, 2019.

[4] Li-Ta Hsu. Analysis and modeling gps nlos effect in highly urbanized area. *GPS solutions*, 22(1):7, 2018.

[5] Huan Yin, Xuecheng Xu, Sha Lu, Xieyuanli Chen, Rong Xiong, Shaojie Shen, Cyrill Stachniss, and Yue Wang. A survey on global LiDAR localization: Challenges, advances and open problems. *Int. J. Comput. Vis.*, pages 1–33, 2024.

[6] Peng Yin, Shiqi Zhao, Ivan Cisneros, Abulikemu Abuduweili, Guoquan Huang, Micheal Milford, Changliu Liu, Howie Choset, and Sebastian Scherer. General

place recognition survey: Towards the real-world autonomy age. *arXiv preprint arXiv:2209.04497*, 2022.

[7] Stephanie Lowry, Niko Sünderhauf, Paul Newman, John J Leonard, David Cox, Peter Corke, and Michael J Milford. Visual place recognition: A survey. *IEEE Trans. Robot.*, 32(1):1–19, Nov. 2015.

[8] Sourav Garg, Tobias Fischer, and Michael Milford. Where is your place, visual place recognition? In *IJCAI*, volume 8, pages 4416–4425, 2021.

[9] Weixin Ma, Huan Yin, Lei Yao, Yuxiang Sun, and Zhongqing Su. Evaluation of range sensing-based place recognition for long-term urban localization. *IEEE Trans. Intell. Veh.*, pages 1–12, 2024.

[10] Yongqiang Lu, Hongjie Ma, Edward Smart, and Hui Yu. Real-time performance-focused localization techniques for autonomous vehicle: A review. *IEEE Trans. Intell. Transp. Syst.*, 23(7):6082–6100, May. 2022.

[11] Peide Cai, Hengli Wang, Yuxiang Sun, and Ming Liu. DQ-GAT: Towards safe and efficient autonomous driving with deep Q-learning and graph attention networks. *IEEE Trans. Intell. Transp. Syst.*, 23(11):21102–21112, Nov. 2022.

[12] Weixin Ma, Shoudong Huang, and Yuxiang Sun. Triplet-Graph: Global metric localization based on semantic triplet graph for autonomous vehicles. *IEEE Robot. Automat. Lett*, 9(4):3155–3162, Apr. 2024.

[13] Dongchan Min, Minchan Kim, Jinsil Lee, Mihaela Simona Circiu, Michael Meurer, and Jiyun Lee. DNN-based approach to mitigate multipath errors of differential GNSS reference stations. *IEEE Trans. Intell. Transp. Syst.*, 23(12):25047–25053, Sep. 2022.

[14] David Betaille, Francois Peyret, Miguel Ortiz, Stephan Miquel, and Leila Fontenay. A new modeling based on urban trenches to improve GNSS positioning quality of service in cities. *IEEE Intell. Transp. Syst. Mag.*, 5(3):59–70, 2013.

[15] Rakesh Kumar and Mark G Petovello. A novel GNSS positioning technique for improved accuracy in urban canyon scenarios using 3D city model. In *Proc. 27th Int. Tech. Meeting Satell. Division Inst. Navigat. (ION GNSS),*, pages 2139–2148, Sep. 2014.

[16] Konstantinos A Tsintotas, Loukas Bampis, and Antonios Gasteratos. The revisiting problem in simultaneous localization and mapping: A survey on visual loop closure detection. *IEEE Trans. Intell. Transp. Syst.*, 23(11):19929–19953, May. 2022.

[17] Azam Rafique Memon, Zhe Liu, and Hesheng Wang. Invariant loop closure detection using step-wise learning with controlling embeddings of landmarks. *IEEE Trans. Intell. Transp. Syst.*, 23(11):20148–20159, May. 2022.

[18] M. Usman Maqbool Bhutta, Yuxiang Sun, Darwin Lau, and Ming Liu. Why-So-Deep: Towards boosting previously trained models for visual place recognition. *IEEE Robot. Automat. Lett*, 7(2):1824–1831, Apr. 2022.

[19] Tobias Törnros, Helen Dorn, Stefan Hahmann, and Alexander Zipf. Uncertainties of completeness measures in OpenStreetMap–a case study for buildings in a medium-sized german city. *ISPRS Ann. Photogramm. Remote Sens. Spat. Inf. Sci.*, 2:353–357, 2015.

[20] Matthias Hentschel and Bernardo Wagner. Autonomous robot navigation based on on OpenStreetMap geodata. In *Proc. IEEE trans. Intell. Transp. Syst.*, pages 1645–1650. IEEE, 2010.

[21] Weisong Wen, Yiyang Zhou, Guohao Zhang, Saman Fahandezh-Saadi, Xiwei Bai, Wei Zhan, Masayoshi Tomizuka, and Li-Ta Hsu. UrbanLoco: A full sensor suite dataset for mapping and localization in urban scenes. In *Proc. IEEE Int. Conf. Robot. Automat.*, pages 2310–2316, May 2020.

[22] Oliver Pink. Visual map matching and localization using a global feature map. In *Proc. IEEE Comput. Soc. Conf. Comput. Vision Pattern Recognit. Workshops*, pages 1–7, Jun. 2008.

[23] Keith Yu Kit Leung, Christopher M Clark, and Jan P Huissoon. Localization in urban environments by matching ground level video images with an aerial image. In *Proc. IEEE Int. Conf. Robot. Automat.*, pages 551–556, May 2008.

[24] Pratik Agarwal, Wolfram Burgard, and Luciano Spinello. Metric localization using Google Street View. In *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, pages 3111–3118, Sept. 2015.

[25] Xipeng Wang, Steve Vozar, and Edwin Olson. Flag: Feature-based localization between air and ground. In *Proc. IEEE Int. Conf. Robot. Automat.*, pages 3178–3184, May 2017.

[26] Noha Radwan, Gian Diego Tipaldi, Luciano Spinello, and Wolfram Burgard. Do you see the bakery? Leveraging geo-referenced texts for global localization in public maps. In *Proc. IEEE Int. Conf. Robot. Automat.*, pages 4837–4842, May 2016.

[27] Dong-Ki Kim and Matthew R Walter. Satellite image-based localization via learned embeddings. In *Proc. IEEE Int. Conf. Robot. Automat.*, pages 2073–2080, May 2017.

[28] Sixing Hu, Mengdan Feng, Rang MH Nguyen, and Gim Hee Lee. CVM-Net: Cross-view matching network for image-based ground-to-aerial geo-localization. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, pages 7258–7267, 2018.

[29] Sixing Hu and Gim Hee Lee. Image-based geo-localization using satellite imagery. *Int. J. Comput. Vis*, 128(5):1205–1219, 2020.

[30] Sijie Zhu, Mubarak Shah, and Chen Chen. Transgeo: Transformer is all you need for cross-view image geo-localization. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, pages 1162–1171, 2022.

[31] Rainer Kümmerle, Bastian Steder, Christian Dornhege, Alexander Kleiner, Giorgio Grisetti, and Wolfram Burgard. Large scale graph-based slam using aerial images as prior information. *Auton. Robots*, 30(1):25–39, Jan. 2011.

[32] Tim Yuqing Tang, Daniele De Martini, Dan Barnes, and Paul Newman. RSL-Net: Localising in satellite images from a radar on the ground. *IEEE Robot. Automat. Lett*, 5(2):1087–1094, Jan. 2020.

[33] Tim Y Tang, Daniele De Martini, Shangzhe Wu, and Paul Newman. Self-supervised learning for using overhead imagery as maps in outdoor range sensor localization. *Int. J. Robot. Res.*, 40(12-14):1488–1509, Dec. 2021.

[34] Marwan Hussein, Matthew Renner, Masaaki Watanabe, and Karl Iagnemma. Matching of ground-based LiDAR and aerial image data for mobile robot localization in densely forested environments. In *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, pages 1432–1437, Nov. 2013.

[35] Ian D Miller, Anthony Cowley, Ravi Konkimalla, Shreyas S Shivakumar, Ty Nguyen, Trey Smith, Camillo Jose Taylor, and Vijay Kumar. Any way you look at it: Semantic crossview localization and mapping with LiDAR. *IEEE Robot. Automat. Lett*, 6(2):2397–2404, Feb. 2021.

[36] Georgios Floros, Benito Van Der Zander, and Bastian Leibe. OpenStreetSLAM: Global vehicle localization using OpenStreetMaps. In *Proc. IEEE Int. Conf. Robot. Automat.*, pages 1054–1059, May 2013.

[37] Marcus A Brubaker, Andreas Geiger, and Raquel Urtasun. Map-based probabilistic visual self-localization. *IEEE Trans. Pattern Anal. Mach. Intell*, 38(4):652–665, Jul. 2015.

[38] Pilailuck Panphattarasap and Andrew Calway. Automated map reading: Image based localisation in 2-D maps using binary semantic descriptors. In *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, pages 6341–6348, Oct. 2018.

[39] Noe Samano, Mengjie Zhou, and Andrew Calway. You are here: Geolocation by embedding maps and images. In *Proc. Eur. Conf. Comput. Vis.*, pages 502–518. Springer, 2020.

[40] Mengjie Zhou, Xieyuanli Chen, Noe Samano, Cyrill Stachniss, and Andrew Calway. Efficient localisation using images and OpenStreetMaps. In *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, pages 5507–5513, Sept. 2021.

[41] Paul-Edouard Sarlin, Daniel DeTone, Tsun-Yi Yang, Armen Avetisyan, Julian Straub, Tomasz Malisiewicz, Samuel Rota Bulo, Richard Newcombe, Peter Kontschieder, and Vasileios Balntas. Orienternet: Visual localization in 2d public maps with neural matching. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, pages 21632–21642, 2023.

[42] Philipp Ruchti, Bastian Steder, Michael Ruhnke, and Wolfram Burgard. Localization on OpenStreetMap data using a 3D laser scanner. In *Proc. IEEE Int. Conf. Robot. Automat.*, pages 5260–5265, May 2015.

[43] Benjamin Suger and Wolfram Burgard. Global outer-urban navigation with OpenStreetMaps. In *Proc. IEEE Int. Conf. Robot. Automat.*, pages 1417–1422, May 2017.

[44] Fan Yan, Olga Vysotska, and Cyrill Stachniss. Global localization on OpenStreetMap using 4-bit semantic descriptors. In *Proc. of the Europ. Conf. on Mobile Robotics*, pages 1–7, 2019.

[45] Younghun Cho, Giseop Kim, Sangmin Lee, and Jee-Hwan Ryu. OpenStreetMap-based LiDAR global localization in urban environment

without a prior LiDAR map. *IEEE Robot. Automat. Lett*, 7(2):4999–5006, Feb. 2022.

[46] Timothy D Barfoot. *State estimation for robotics*. Cambridge University Press, 2017.

[47] Ji Zhang and Sanjiv Singh. Low-drift and real-time LiDAR odometry and mapping. *Auton. Robots*, 41(2):401–416, Feb. 2017.

[48] Dieter Fox. Adapting the sample size in particle filters through KLD-sampling. *Int. J. Robot. Res.*, 22(12):985–1003, Dec. 2003.

[49] Nobuyuki Otsu et al. A threshold selection method from gray-level histograms. *Automatica*, 11(285-296):23–27, 1975.

[50] Renaud Dubé, Daniel Dugas, Elena Stumm, Juan Nieto, Roland Siegwart, and Cesar Cadena. SegMatch: Segment based place recognition in 3D point clouds. In *Proc. IEEE Int. Conf. Robot. Automat.*, pages 5266–5272. IEEE, 2017.

[51] Athanasios Chalvatzaras, Ioannis Pratikakis, and Angelos A Amanatiadis. A survey on map-based localization techniques for autonomous vehicles. *IEEE Trans. Intell. Veh.*, 8(2):1574–1596, 2022.

[52] Kristin Nielsen and Gustaf Hendeby. Multi-Hypothesis SLAM for non-static environments with reoccurring landmarks. *IEEE Trans. Intell. Veh.*, 8(4):3191–3203, 2023.

[53] Guillaume Bresson, Zayed Alsayed, Li Yu, and Sébastien Glaser. Simultaneous localization and mapping: A survey of current trends in autonomous driving. *IEEE Trans. Intell. Veh.*, 2(3):194–220, 2017.

[54] Lineng Chen, Hui Kong, Huan Wang, Wankou Yang, Jing Lou, Fenglei Xu, and Mingwu Ren. HVP-Net: A hybrid voxel- and point-wise network for place recognition. *IEEE Trans. Intell. Veh.*, 9(1):395–406, 2024.

[55] Mubariz Zaffar, Sourav Garg, Michael Milford, Julian Kooij, David Flynn, Klaus McDonald-Maier, and Shoaib Ehsan. VPR-Bench: An open-source visual place recognition evaluation framework with quantifiable viewpoint and appearance change. *Int. J. Comput. Vis.*, 129(7):2136–2174, 2021.

[56] Dingwen Xiao, Sirui Li, and Zhe Xuanyuan. Semantic loop closure detection for intelligent vehicles using panoramas. *IEEE Trans. Intell. Veh.*, 8(10):4395–4405, 2023.

[57] Weiqiang Zhao, Hang Sun, Xinyu Zhang, and Yijin Xiong. Visual SLAM combining lines and structural regularities: Towards robust localization. *IEEE Trans. Intell. Veh.*, pages 1–18, 2023.

[58] Chris D. Monaco and Sean N. Brennan. RADARODO: Ego-motion estimation from doppler and spatial data in radar images. *IEEE Trans. Intell. Veh.*, 5(3):475–484, 2020.

[59] Keenan Burnett, Yuchen Wu, David J Yoon, Angela P Schoellig, and Timothy D Barfoot. Are we ready for radar to replace lidar in all-weather mapping and localization? *IEEE Robot. Automat. Lett*, 7(4):10328–10335, 2022.

[60] Ziyang Hong, Yvan Petillot, Andrew Wallace, and Sen Wang. RadarSLAM: A robust simultaneous localization and mapping system for all weather conditions. *Int. J. Robot. Res.*, 41(5):519–542, 2022.

[61] Arthur Venon, Yohan Dupuis, Pascal Vasseur, and Pierre Merriaux. Millimeter wave FMCW radars for perception, recognition and localization in automotive applications: A survey. *IEEE Trans. Intell. Veh.*, 7(3):533–555, 2022.

[62] Nader J. Abu-Alrub and Nathir A. Rawashdeh. Radar odometry for autonomous ground vehicles: A survey of methods and datasets. *IEEE Trans. Intell. Veh.*, pages 1–18, 2023.

[63] Shouyi Lu, Guirong Zhuo, Lu Xiong, Xichan Zhu, Lianqing Zheng, Zihang He, Mingyu Zhou, Xinfei Lu, and Jie Bai. Efficient deep-learning 4D automotive radar odometry method. *IEEE Trans. Intell. Veh.*, 9(1):879–892, 2024.

[64] Jiachong Chang, Runzhi Hu, Feng Huang, Dingjie Xu, and Li-Ta Hsu. LiDAR-Based NDT matching performance evaluation for positioning in adverse weather conditions. *IEEE Sensors Journal*, 23(20):25346–25355, 2023.

[65] Bruno Ferrarini, Maria Waheed, Sania Waheed, Shoaib Ehsan, Michael J Milford, and Klaus D McDonald-Maier. Exploring performance bounds of visual place recognition using extended precision. *IEEE Robot. Automat. Lett*, 5(2):1688–1695, 2020.

[66] Radu Bogdan Rusu, Nico Blodow, and Michael Beetz. Fast point feature histograms (FPFH) for 3D registration. In *Proc. IEEE Int. Conf. Robot. Autom.*, pages 3212–3217, 2009.

[67] Martin Magnusson, Henrik Andreasson, Andreas Nüchter, and Achim J Lilienthal. Automatic appearance-based loop detection from three-dimensional laser data using the normal distributions transform. *J. Field Robot.*, 26(11-12):892–914, 2009.

[68] Yunfeng Fan, Yichang He, and U-Xuan Tan. Seed: A segmentation-based egocentric 3D point cloud descriptor for loop closure detection. In *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, pages 5158–5163, 2020.

[69] Yachen Zhu, Yanyang Ma, Long Chen, Cong Liu, Maosheng Ye, and Lingxi Li. GOSMatch: Graph-of-semantics matching for detecting loop closures in 3D LiDAR data. In *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, pages 5151–5157, 2020.

[70] Li He, Xiaolong Wang, and Hong Zhang. M2DP: A novel 3D point cloud descriptor and its application in loop closure detection. In *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, pages 231–237, 2016.

[71] Giseop Kim and Ayoung Kim. Scan Context: Egocentric spatial descriptor for place recognition within 3D point cloud map. In *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, pages 4802–4809, 2018.

[72] Giseop Kim, Sunwook Choi, and Ayoung Kim. Scan Context++: Structural place recognition robust to rotation and lateral variations in urban environments. *IEEE Trans. Robot.*, 38(3):1856–1874, 2021.

[73] Lin Li, Xin Kong, Xiangrui Zhao, Tianxin Huang, Wanlong Li, Feng Wen, Hongbo Zhang, and Yong Liu. SSC: Semantic scan context for large-scale place recognition. In *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, pages 2092–2099, 2021.

[74] Han Wang, Chen Wang, and Lihua Xie. Intensity Scan Context: Coding intensity and geometry relations for loop closure detection. In *Proc. IEEE Int. Conf. Robot. Autom.*, pages 2095–2101, 2020.

[75] Xuecheng Xu, Huan Yin, Zexi Chen, Yuehua Li, Yue Wang, and Rong Xiong. DiSCO: Differentiable scan context with orientation. *IEEE Robot. Automat. Lett*, 6(2):2791–2798, 2021.

[76] Xuecheng Xu, Sha Lu, Jun Wu, Haojian Lu, Qiuguo Zhu, Yiyi Liao, Rong Xiong, and Yue Wang. RING++: Roto-translation invariant gram for global localization on a sparse scan map. *IEEE Trans. Robot.*, 39(6):4616–4635, 2023.

[77] Ying Wang, Zezhou Sun, Cheng-Zhong Xu, Sanjay E Sarma, Jian Yang, and Hui Kong. LiDAR Iris for loop-closure detection. In *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, pages 5769–5775, 2020.

[78] Mikaela Angelina Uy and Gim Hee Lee. PointNetVLAD: Deep point cloud based retrieval for large-scale place recognition. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, pages 4470–4479, 2018.

[79] Relja Arandjelovic, Petr Gronat, Akihiko Torii, Tomas Pajdla, and Josef Sivic. NetVLAD: CNN architecture for weakly supervised place recognition. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, pages 5297–5307, 2016.

[80] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. PointNet: Deep learning on point sets for 3D classification and segmentation. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, pages 652–660, 2017.

[81] Jacek Komorowski. MinkLoc3D: Point cloud based large-scale place recognition. In *Proc. IEEE Winter Conf. Appl. Comput. Vis.*, pages 1790–1799, 2021.

[82] Kamil Żywanowski, Adam Banaszczyk, Michał R Nowicki, and Jacek Komorowski. MinkLoc3D-SI: 3D LiDAR place recognition with sparse convolutions, spherical coordinates, and intensity. *IEEE Robot. Automat. Lett*, 7(2):1079–1086, 2021.

[83] Filip Radenović, Giorgos Tolias, and Ondřej Chum. Fine-tuning CNN image retrieval with no human annotation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 41(7):1655–1668, 2018.

[84] Matthew Gadd, Daniele De Martini, and Paul Newman. Look Around You: Sequence-based radar place recognition with learned rotational invariance. In *IEEE/ION Position Locat. Navig. Symp.*, pages 270–276, 2020.

[85] Matthew Gadd, Daniele De Martini, and Paul Newman. Contrastive learning for unsupervised radar place recognition. In *Proc. IEEE Int. Conf. Advanc. Robot.*, pages 344–349, 2021.

[86] Ştefan Săftescu, Matthew Gadd, Daniele De Martini, Dan Barnes, and Paul Newman. Kidnapped Radar: Topological radar localisation using rotationally-invariant metric learning. In *Proc. IEEE Int. Conf. Robot. Autom.*, pages 4358–4364, 2020.

[87] Farid Alijani, Jukka Peltomäki, Jussi Puura, Heikki Huttunen, Joni-Kristian Kämäräinen, and Esa Rahtu. Long-term visual place recognition. In *2022 26th International Conference on Pattern Recognition (ICPR)*, pages 3422–3428, 2022.

[88] Will Maddern, Geoffrey Pascoe, Chris Linegar, and Paul Newman. 1 year, 1000 km: The oxford robotcar dataset. *Int. J. Robot. Res.*, 36(1):3–15, 2017.

[89] Jukka Peltomäki, Farid Alijani, Jussi Puura, Heikki Huttunen, Esa Rahtu, and Joni-Kristian Kämäräinen. Evaluation of long-term LiDAR place recognition. In *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, pages 4487–4492, 2021.

[90] Filip Radenović, Giorgos Tolias, and Ondřej Chum. CNN image retrieval learns from BoW: Unsupervised fine-tuning with hard examples. In *Proc. Eur. Conf. Comput. Vis*, pages 3–20. Springer, 2016.

[91] Kamil Żywanowski, Adam Banaszczyk, and Michał R Nowicki. Comparison of camera-based and 3D LiDAR-based place recognition across weather conditions. In *Proc. Int. Conf. Control Autom. Robot. Vis.*, pages 886–891, 2020.

[92] Fengkui Cao, Fei Yan, Sen Wang, Yan Zhuang, and Wei Wang. Season-invariant and viewpoint-tolerant LiDAR place recognition in GPS-denied environments. *IEEE Trans. Ind. Electron.*, 68(1):563–574, 2020.

[93] Fengkui Cao, Hao Wu, and Chengdong Wu. An end-to-end localizer for long-term topological localization in large-scale changing environments. *IEEE Trans. Ind. Electron.*, 70(5):5140–5149, 2022.

[94] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The kitti dataset. *Int. J. Robot. Res.*, 32(11):1231–1237, 2013.

[95] Dan Barnes, Matthew Gadd, Paul Murcutt, Paul Newman, and Ingmar Posner. The oxford radar robotcar dataset: A radar extension to the oxford robotcar dataset. In *Proc. IEEE Int. Conf. Robot. Autom.*, pages 6433–6438, 2020.

[96] Giseop Kim, Yeong Sang Park, Younghun Cho, Jinyong Jeong, and Ayoung Kim. Mulran: Multimodal range dataset for urban place recognition. In *Proc. IEEE Int. Conf. Robot. Autom.*, pages 6246–6253, 2020.

[97] Keenan Burnett, David J Yoon, Yuchen Wu, Andrew Z Li, Haowei Zhang, Shichen Lu, Jingxing Qian, Wei-Kang Tseng, Andrew Lambert, Keith YK Leung, et al. Boreas: A multi-season autonomous driving dataset. *Int. J. Robot. Res.*, 42(1-2):33–42, 2023.

[98] Jacek Komorowski. Improving point cloud based place recognition with ranking-based loss and large batch training. In *2022 26th International Conference on Pattern Recognition (ICPR)*, pages 3699–3705, 2022.

[99] Junyi Ma, Jun Zhang, Jintao Xu, Rui Ai, Weihao Gu, and Xieyuanli Chen. OverlapTransformer: An efficient and yaw-angle-invariant transformer network for LiDAR-based place recognition. *IEEE Robot. Automat. Lett*, 7(3):6958–6965, 2022.

[100] Sarah H Cen and Paul Newman. Precise ego-motion estimation with millimeter-wave radar under diverse and challenging conditions. In *Proc. IEEE Int. Conf. Robot. Autom.*, pages 6045–6052, 2018.

[101] Xiaqing Ding, Xuecheng Xu, Sha Lu, Yanmei Jiao, Mengwen Tan, Rong Xiong, Huanjun Deng, Mingyang Li, and Yue Wang. Translation invariant global estimation of heading angle using sinogram of lidar point cloud. In *Proc. IEEE Int. Conf. Robot. Autom.*, pages 2207–2214, 2022.

[102] Sha Lu, Xuecheng Xu, Huan Yin, Zexi Chen, Rong Xiong, and Yue Wang. One ring to rule them all: Radon sinogram for place recognition, orientation and translation estimation. In *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, pages 2778–2785, 2022.

[103] X. Chen, T. Läbe, A. Milioto, T. Röhling, J. Behley, and C. Stachniss. OverlapNet: A Siamese Network for Computing LiDAR Scan Similarity with Applications to Loop Closing and Localization. *Autonomous Robots*, 46:61–81, 2021.

[104] Lukas Schaupp, Mathias Bürki, Renaud Dubé, Roland Siegwart, and Cesar Cadena. Oreos: Oriented recognition of 3d point clouds in outdoor scenarios. In *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, pages 3255–3261, 2019.

[105] Tixiao Shan, Brendan Englot, Fábio Duarte, Carlo Ratti, and Daniela Rus. Robust place recognition using an imaging lidar. In *Proc. IEEE Int. Conf. Robot. Autom.*, pages 5469–5475, 2021.

[106] Luca Di Giammarino, Irvin Aloise, Cyrill Stachniss, and Giorgio Grisetti. Visual place recognition using lidar intensity information. In *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, pages 4382–4389, 2021.

[107] Daniele Cattaneo, Matteo Vaghi, and Abhinav Valada. Lcdnet: Deep loop closure detection and point cloud registration for lidar slam. *IEEE Trans. Robot.*, 38(4):2074–2093, 2022.

[108] Juan Du, Rui Wang, and Daniel Cremers. Dh3d: Deep hierarchical 3d descriptors for robust large-scale 6dof relocalization. In *European Conference on Computer Vision*, pages 744–762. Springer, 2020.

[109] Jacek Komorowski, Monika Wysoczanska, and Tomasz Trzcinski. Egonn: Egocentric neural network for point cloud based 6dof relocalization at the city scale. *IEEE Robot. Automat. Lett*, 7(2):722–729, 2021.

[110] Yunge Cui, Xieyuanli Chen, Yinlong Zhang, Jiahua Dong, Qingxiao Wu, and Feng Zhu. Bow3d: Bag of words for real-time loop closing in 3d lidar slam. *IEEE Robot. Automat. Lett*, 8(5):2828–2835, 2022.

[111] Yunge Cui, Yinlong Zhang, Jiahua Dong, Haibo Sun, Xieyuanli Chen, and Feng Zhu. Link3d: Linear keypoints representation for 3d lidar point cloud. *IEEE Robot. Automat. Lett*, 2024.

[112] Michael Bosse and Robert Zlot. Place recognition using keypoint voting in large 3d lidar datasets. In *Proc. IEEE Int. Conf. Robot. Autom.*, pages 2677–2684, 2013.

[113] Jiadong Guo, Paulo VK Borges, Chanoh Park, and Abel Gawel. Local descriptor for robust place recognition using lidar intensity. *IEEE Robot. Automat. Lett*, 4(2):1470–1477, 2019.

[114] Georgi Pramatarov, Daniele De Martini, Matthew Gadd, and Paul Newman. BoxGraph: Semantic place recognition and pose estimation from 3D LiDAR. In *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, pages 7004–7011, 2022.

[115] Radu Bogdan Rusu and Steve Cousins. 3D is here: Point cloud library (PCL). In *Proc. IEEE Int. Conf. Robot. Autom.*, pages 1–4, 2011.

[116] Xiyue Guo, Junjie Hu, Junfeng Chen, Fuqin Deng, and Tin Lun Lam. Semantic histogram based graph matching for real-time multi-robot global localization in large scale environment. *IEEE Robot. Automat. Lett*, 6(4):8349–8356, 2021.

[117] Martin A Fischler and Robert C Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.

[118] Sameer Agarwal, Keir Mierle, and The Ceres Solver Team. Ceres Solver, 10 2023.

[119] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, pages 3354–3361, 2012.

[120] Xin Kong, Xuemeng Yang, Guangyao Zhai, Xiangrui Zhao, Xianfang Zeng, Mengmeng Wang, Yong Liu, Wanlong Li, and Feng Wen. Semantic graph based place recognition for 3d point clouds. In *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, pages 8216–8223, 2020.

[121] Jens Behley, Martin Garbade, Andres Milioto, Jan Quenzel, Sven Behnke, Cyrill Stachniss, and Jurgen Gall. Semantickitti: A dataset for semantic scene understanding of lidar sequences. In *Proc. IEEE Int. Conf. Comp. Vis.*, pages 9297–9307, 2019.

[122] Andres Milioto, Ignacio Vizzo, Jens Behley, and Cyrill Stachniss. Rangenet++: Fast and accurate lidar semantic segmentation. In *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, pages 4213–4220, 2019.

[123] Christopher Choy, Jaesik Park, and Vladlen Koltun. Fully convolutional geometric features. In *Proc. IEEE Int. Conf. Comp. Vis.*, pages 8958–8966, 2019.

[124] Jacqueline Ankenbauer, Parker C Lusk, Annika Thomas, and Jonathan P How. Global localization in unstructured environments using semantic object maps built from various viewpoints. In *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, pages 1358–1365, 2023.

[125] Pengyu Yin, Haozhi Cao, Thien-Minh Nguyen, Shenghai Yuan, Shuyang Zhang, Kangcheng Liu, and Lihua Xie. Outram: One-shot global localization via triangulated scene graph and global outlier pruning. *arXiv preprint arXiv:2309.08914*, 2023.

[126] Shigemichi Matsuzaki, Kenji Koide, Shuji Oishi, Masashi Yokozuka, and Atsuhiko Banno. Single-shot global localization via graph-theoretic correspondence matching. *Advanced Robotics*, pages 1–14, 2024.

[127] Frank Dellaert, Dieter Fox, Wolfram Burgard, and Sebastian Thrun. Monte carlo localization for mobile robots. In *Proc. IEEE Int. Conf. Robot. Autom.*, volume 2, pages 1322–1328, 1999.

[128] Chongjian Yuan, Jiarong Lin, Zuhao Zou, Xiaoping Hong, and Fu Zhang. STD: Stable triangle descriptor for 3D place recognition. In *Proc. IEEE Int. Conf. Robot. Autom.*, pages 1897–1903, 2023.

[129] Haotian Tang, Zhijian Liu, Shengyu Zhao, Yujun Lin, Ji Lin, Hanrui Wang, and Song Han. Searching efficient 3D architectures with sparse point-voxel convolution. In *Proc. Eur. Conf. Comput. Vis.*, pages 685–702. Springer, 2020.

[130] Ryan A Rossi, David F Gleich, and Assefaw H Gebremedhin. Parallel maximum clique algorithms with applications to network analysis. *SIAM J. Sci. Comput.*, 37(5):C589–C616, 2015.

[131] Heng Yang, Pasquale Antonante, Vasileios Tzoumas, and Luca Carlone. Graduated non-convexity for robust spatial perception: From non-minimal solvers to global outlier rejection. *IEEE Robot. Automat. Lett*, 5(2):1127–1134, 2020.

[132] Heng Yang, Jingnan Shi, and Luca Carlone. TEASER: Fast and certifiable point cloud registration. *IEEE Trans. Robot.*, 37(2):314–333, 2020.

[133] Hyungtae Lim, Beomsoo Kim, Daebeom Kim, Eungchang Mason Lee, and Hyun Myung. Quatro++: Robust global registration exploiting ground segmentation for loop closing in lidar slam. *Int. J. Robot. Res.*, 43(5):685–715, 2024.

[134] Andrew Blake and Andrew Zisserman. *Visual reconstruction*. MIT press, 1987.

[135] Kenneth Rose. Deterministic annealing for clustering, compression, classification, regression, and related optimization problems. *Proceedings of the IEEE*, 86(11):2210–2239, 1998.

[136] Frank Dellaert and GTSAM Contributors. borglab/gtsam, May 2022.

[137] Minwoo Jung, Wooseong Yang, Dongjae Lee, Hyeonjae Gil, Giseop Kim, and Ayoung Kim. Helipr: Heterogeneous LiDAR dataset for inter-lidar place recognition under spatiotemporal variations. *Int. J. Robot. Res.*, page 02783649241242136, 2023.

[138] Renaud Dubé, Abel Gawel, Hannes Sommer, Juan Nieto, Roland Siegwart, and Cesar Cadena. An online multi-robot slam system for 3d lidars. In *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, pages 1004–1011, 2017.

[139] Renaud Dubé. *Real-Time Multi-Robot Localization and Mapping with 3D Point Clouds*. PhD thesis, ETH Zurich, 2018.

[140] Sajad Saeedi, Michael Trentini, Mae Seto, and Howard Li. Multiple-robot simultaneous localization and mapping: A review. *Journal of Field Robotics*, 33(1):3–46, 2016.

[141] Keiji Nagatani, Yoshito Okada, Naoki Tokunaga, Seiga Kiribayashi, Kazuya Yoshida, Kazunori Ohno, Eijiro Takeuchi, Satoshi Tadokoro, Hidehisa Akiyama, Itsuki Noda, et al. Multirobot exploration for search and rescue missions: A report on map building in robocuprescue 2009. *Journal of Field Robotics*, 28(3):373–387, 2011.

[142] Nathan Michael, Shaojie Shen, Kartik Mohta, Yash Mulgaonkar, Vijay Kumar, Keiji Nagatani, Yoshito Okada, Seiga Kiribayashi, Kazuki Otake, Kazuya Yoshida, et al. Collaborative mapping of an earthquake-damaged building via ground and aerial robots. *Journal of Field Robotics*, 29(5):832–841, 2012.

[143] Lars AA Andersson and Jonas Nygards. C-sam: Multi-robot slam using square root information smoothing. In *Proc. IEEE Int. Conf. Robot. Autom.*, pages 2798–2805, 2008.

[144] Isaac Deutsch, Ming Liu, and Roland Siegwart. A framework for multi-robot pose graph slam. In *2016 IEEE International Conference on Real-time Computing and Robotics (RCAR)*, pages 567–572. IEEE, 2016.

[145] Haolin Dong, Jincheng Yu, Yuanfan Xu, Zhilin Xu, Zhaoyang Shen, Jiahao Tang, Yuan Shen, and Yu Wang. MR-GMMapping: Communication efficient multi-robot mapping system via gaussian mixture model. *IEEE Robotics and Automation Letters*, 7(2):3294–3301, 2022.

[146] Yulun Tian, Yun Chang, Fernando Herrera Arias, Carlos Nieto-Granda, Jonathan P How, and Luca Carlone. Kimera-multi: Robust, distributed, dense metric-semantic slam for multi-robot systems. *IEEE Trans. Robot.*, 38(4), 2022.

[147] Xu Liu, Jiuzhou Lei, Ankit Prabhu, Yuezhan Tao, Igor Spasojevic, Pratik Chaudhari, Nikolay Atanasov, and Vijay Kumar. Slideslam: Sparse, lightweight, decentralized metric-semantic slam for multi-robot navigation. *arXiv preprint arXiv:2406.17249*, 2024.

[148] Wei Xu, Yixi Cai, Dongjiao He, Jiarong Lin, and Fu Zhang. Fast-lio2: Fast direct lidar-inertial odometry. *IEEE Transactions on Robotics*, 38(4):2053–2073, 2022.

[149] Chunran Zheng, Wei Xu, Zuhao Zou, Tong Hua, Chongjian Yuan, Dongjiao He, Bingyang Zhou, Zheng Liu, Jiarong Lin, Fangcheng Zhu, et al. Fast-livo2: Fast, direct lidar-inertial-visual odometry. *arXiv preprint arXiv:2408.14035*, 2024.

[150] Vadim Indelman, Erik Nelson, Nathan Michael, and Frank Dellaert. Multi-robot pose graph localization and data association from unknown initial relative poses

via expectation maximization. In *Proc. IEEE Int. Conf. Robot. Autom.*, pages 593–600, 2014.