Titre: Title:	Robust Real-Time Simultaneous Localization and Mapping with LiDAR in Dynamic Environments
Auteur: Author:	Wenqiang Du
Date:	2024
Type:	Mémoire ou thèse / Dissertation or Thesis
Référence: Citation:	Du, W. (2024). Robust Real-Time Simultaneous Localization and Mapping with LiDAR in Dynamic Environments [Thèse de doctorat, Polytechnique Montréal]. PolyPublie. https://publications.polymtl.ca/59215/

URL de PolyPublie: PolyPublie URL:	https://publications.polymtl.ca/59215/
Directeurs de recherche: Advisors:	Giovanni Beltrame
Programme: Program:	Génie informatique

POLYTECHNIQUE MONTRÉAL
affiliée à l'Université de Montréal

Robust Real-Time Simultaneous Localization and Mapping with LiDAR in Dynamic Environments

WENQIANG DU
Département de génie informatique et génie logiciel

Thèse présentée en vue de l'obtention du diplôme de *Philosophiæ Doctor*
Génie informatique

Août 2024

POLYTECHNIQUE MONTRÉAL
affiliée à l'Université de Montréal

Cette thèse intitulée :

Robust Real-Time Simultaneous Localization and Mapping with LiDAR in Dynamic Environments

présentée par **Wenqiang DU**
en vue de l'obtention du diplôme de *Philosophiæ Doctor*
a été dûment acceptée par le jury d'examen constitué de :

Ettore MERLO, président
Giovanni BELTRAME, membre et directeur de recherche
Tarek OULD-BACHIR, membre
Gregory DUDEK, membre externe

DEDICATION

À tous mes amis du labos,
vous me manquerez...

ACKNOWLEDGEMENTS

Pursuing a PhD is a transformative and multifaceted experience, characterized not only by the technical complexities and obstacles that one must overcome but also by the invaluable interactions and lessons learned from others along the way.

I would like to extend my heartfelt appreciation to my supervisor, Prof. Giovanni Beltrame. Working with him over the past few years has been an immense privilege. His unwavering support, exceptional technical guidance, and steadfast belief in my abilities have been instrumental in my development as a researcher. Prof. Beltrame's exemplary leadership, characterized by his patience, clear communication, and passion for his work, has been a source of inspiration and a model for me to emulate in my future endeavors.

I am also deeply grateful to my friends and colleagues at MISTLab. The collaborative environment and the many insightful discussions we shared have been invaluable to my research. The willingness of my colleagues to offer their time and expertise selflessly has greatly contributed to the progress and completion of my thesis.

To my friends, your unwavering support and encouragement have been a cornerstone of my PhD journey. The strength of our friendships has provided me with the motivation and resolve to persevere through the toughest times. Your companionship has made this journey not only bearable but also enjoyable and fulfilling.

Lastly, I owe a profound debt of gratitude to my family. Their unconditional support, sacrifices, and constant encouragement have been the bedrock upon which I have built my academic pursuits. The unwavering belief and patience of my family have fueled my determination and commitment to see this journey through to the end. No words can fully capture the extent of my gratitude and love for my family.

Thank you all for your immeasurable contributions to my PhD journey. This achievement is as much yours as it is mine.

RÉSUMÉ

Dans des environnements dynamiques et non structurés, la Simultaneous Localization and Mapping (SLAM) précises sont essentielles pour les robots autonomes. Les systèmes d'odométrie basés sur le Light Detection and Ranging (LiDAR) traditionnels s'appuient fortement sur des caractéristiques géométriques telles que les points, les lignes et les plans. Cependant, dans des environnements où ces caractéristiques sont rares ou absentes, ces méthodes échouent souvent, rendant difficile le maintien d'une localisation et d'une cartographie précises. De plus, les environnements dynamiques introduisent des complexités supplémentaires, car les objets en mouvement peuvent entraîner des erreurs d'appariement des points caractéristiques et des dérives d'odométrie. Ces objets dynamiques doivent être détectés et suivis en temps réel pour garantir une navigation sûre et efficace.

Cette thèse aborde ces deux défis en proposant deux méthodologies SLAM complémentaires. La première méthode vise à améliorer le SLAM LiDAR dans des environnements non structurés, tandis que la deuxième méthode se concentre sur la détection et le suivi des objets dynamiques à l'aide des données LiDAR.

La première partie de la thèse présente une méthode innovante de SLAM en temps réel basée sur les images d'intensité du LiDAR, conçue pour surmonter la dégénérescence géométrique dans les environnements non structurés. Les systèmes d'odométrie frontale basés sur le LiDAR traditionnels s'appuient sur des caractéristiques géométriques telles que les points, les lignes et les plans. Dans des environnements dépourvus de ces caractéristiques, le système d'odométrie peut échouer. Pour résoudre ce problème, nous extrayons des points caractéristiques des nuages de points générés par le LiDAR qui correspondent aux caractéristiques identifiées dans les images d'intensité du LiDAR. Ces points caractéristiques sont utilisés pour l'enregistrement des scans et l'estimation du mouvement propre. Notre méthode combine la robustesse du SLAM LiDAR avec les avantages du SLAM visuel, minimisant les problèmes liés au flou et aux changements d'éclairage. Le système comprend un front-end léger et un back-end qui optimise les distances entre les points caractéristiques correspondants et les distances point-à-plan pour les plans identifiés dans la carte. De plus, les caractéristiques extraites des images d'intensité sont utilisées pour détecter les boucles de fermeture, suivies de l'optimisation du graphe de poses.

Les résultats expérimentaux de cette méthode montrent des performances en temps réel avec une grande précision dans divers environnements difficiles, y compris ceux avec des changements d'éclairage et des surfaces à faible texture. L'efficacité de ce système SLAM

est validée par des tests rigoureux en intérieur et en extérieur, montrant des améliorations significatives par rapport aux méthodes existantes pour gérer la dégénérescence géométrique.

La deuxième partie de la thèse traite de la nécessité cruciale de détecter et de suivre les objets en mouvement dans des environnements dynamiques en utilisant les données LiDAR. Les méthodes traditionnelles de détection d'objets dynamiques reposent sur une odométrie et des cartes de haute précision, ce qui n'est pas adapté à une opération à long terme dans des environnements en constante évolution. Nous proposons un système novateur qui met l'accent sur l'extraction des composants à basse fréquence des données LiDAR comme points caractéristiques pour les objets de premier plan. Cette approche réduit considérablement le temps nécessaire pour le regroupement des objets et l'analyse des mouvements. Notre système utilise la méthode d'estimation du mouvement propre basée sur l'intensité introduite dans la première partie, ainsi qu'une technique de fenêtre glissante pour évaluer les mouvements des objets, améliorant ainsi la résilience aux dérives d'odométrie.

Les expériences montrent que ce système peut détecter et suivre des objets dynamiques en temps réel avec une grande précision et des taux de rappel élevés. La capacité à identifier précisément les objets en mouvement et à améliorer la résilience du système aux dérives d'odométrie est cruciale pour la navigation autonome dans des environnements dynamiques tels que les zones urbaines, les entrepôts et les chantiers de construction.

En résumé, cette thèse présente deux méthodologies innovantes pour relever les défis du SLAM dans des environnements non structurés et dynamiques. La première méthode améliore la précision et la robustesse du SLAM dans des environnements avec peu de caractéristiques géométriques en utilisant des images d'intensité LiDAR. La deuxième méthode améliore la capacité de détection et de suivi des objets dynamiques en utilisant les données LiDAR. Ensemble, ces contributions offrent des avancées substantielles dans la navigation autonome, avec des applications potentielles en robotique et dans les véhicules autonomes nécessitant des capacités SLAM robustes et précises.

ABSTRACT

In dynamic and unstructured environments, achieving precise simultaneous localization and mapping (SLAM) is crucial for autonomous robots. Traditional LiDAR-based odometry systems rely heavily on geometric features such as points, lines, and planes. However, in environments where these features are sparse or absent, these methods often fail, leading to significant challenges in maintaining accurate localization and mapping. Furthermore, dynamic environments introduce additional complexities as moving objects will lead to error feature points matching and odometry drift. These dynamic objects must be detected and tracked in real time to ensure safe and effective navigation.

This thesis addresses these dual challenges by presenting two complementary SLAM methodologies. The first method focuses on enhancing LiDAR SLAM in unstructured environments, while the second method is dedicated to detecting and tracking dynamic objects using LiDAR data.

The first part of the thesis introduces an innovative real-time LiDAR intensity image-based SLAM method designed to overcome geometric degeneracy in unstructured environments. Traditional LiDAR-based front-end odometry systems rely on geometric features such as points, lines, and planes. In environments lacking these features, the entire odometry system can fail. To address this, we extract feature points from LiDAR-generated point clouds that match features identified in LiDAR intensity images. These feature points are used for scan registration and ego-movement estimation. Our method combines the robustness of LiDAR SLAM with the advantages of visual SLAM, minimizing issues related to blurring and illumination changes. The system includes a lightweight front-end and a back-end that optimizes distances between corresponding feature points and point-to-plane distances for planes identified in the map. Additionally, features extracted from intensity images are used to detect loop closure candidates from previous scans, followed by pose graph optimization.

Experimental results for this method demonstrate real-time performance with high accuracy in various challenging environments, including those with illumination changes and low-texture surfaces. The effectiveness of this SLAM system is validated through rigorous testing in both indoor and outdoor scenarios, showing significant improvements over existing methods in handling geometric degeneracy.

The second part of the thesis addresses the critical need for detecting and tracking moving objects in dynamic environments using LiDAR data. Traditional methods for dynamic object detection rely on high accuracy odometry and maps, which are not suitable for long-term

operation in constantly changing environments. We propose a novel system that emphasizes the extraction of low-frequency components from LiDAR data as feature points for foreground objects. This approach significantly reduces the time required for object clustering and movement analysis. Our system utilizes the intensity-based ego-motion estimation method introduced in the first part, along with a sliding window technique to assess object movements, thereby enhancing resilience to odometry drift.

Experiments demonstrate that this system can detect and track dynamic objects in real time with high accuracy and recall rates. The ability to precisely identify moving objects and enhance the system's resilience to odometry drift is crucial for autonomous navigation in dynamic environments such as urban areas, warehouses, and construction sites.

In summary, this thesis presents two innovative methodologies to address the challenges of SLAM in unstructured and dynamic environments. The first method improves SLAM accuracy and robustness in environments with sparse geometric features by leveraging LiDAR intensity images. The second method enhances the capability of detecting and tracking dynamic objects using LiDAR data. Together, these contributions offer substantial advancements in autonomous navigation, with potential applications in robotics and autonomous vehicles requiring robust and precise SLAM capabilities.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS AND ACRONYMS

SLAM	Simultaneous Localization and Mapping
LiDAR	Light Detection and Ranging
DOF	Degrees of Freedom
IMU	inertial measurement unit
CMU	Carnegie Mellon University
LOAM	Lidar Odometry and Mapping
NASA	National Aeronautics and Space Administration
JPL	Jet Propulsion Laboratory
TLS	Truncated Least Squares
RANSAC	Random Sample Consensus
ESIKF	Error-State Iterated Extended Kalman Filter
GNC	Graduated Non-Convexity
rIFL	robust Incremental Fixed Lag Smoother
P-GAT	Pose-Graph Attentional Graph Neural Network
iEKF	Iterated Extended Kalman Filter
ikd-Tree	Incremental k-dimensional Tree
CPU	Central Processing Unit
GPU	Graphics Processing Unit
ViTs	Vision Transformers
ICPSC	intensity cylindrical-projection shape context
ISC	Intensity Scan Context
IA-LIO-SAM	Intensity and Ambient Enhanced Lidar Inertial Odometry via Smoothing and Mapping
KNN	K-Nearest Neighbors
SIFT	Scale-Invariant Feature Transform
ORB	Oriented FAST and rotated BRIEF
SURF	Speed-Up Robust Features
GPU	graphics processing unit
FoV	field of view
BoW	bag of words
DSO	Dircet Sparse Odometry
CNN	Convolutional Neural Network
EKF	Extend Kalman Filter

TSDF	Truncated Signed Distance Field
APE	Absolute Pose Error
RPE	Relative Pose Error
AR	Augmented Reality

LIST OF APPENDICES

CHAPTER 1 INTRODUCTION

1.1 Context and Motivation

In order to achieve fully autonomous exploration in unknown environments, it is crucial for robots to perform accurate localization and mapping. Without proper localization, a robot cannot determine its relative position within the environment, nor can it autonomously explore unknown areas. SLAM [1–4] is a process where a robot builds a map of an environment while simultaneously keeping track of its own location within that map in real-time. SLAM involves several interdependent components. Localization refers to the robot's ability to determine its position and orientation within the map. This is achieved through the integration of various sensor inputs, such as LiDAR, cameras, and inertial measurement unit (IMU). Mapping [5–7], on the other hand, involves building a representation of the environment, which may include obstacles, landmarks, and other features. The accuracy of the map is critical for the robot to navigate effectively. SLAM is essential for autonomous robots to navigate and understand their surroundings.

There are various types of unknown environments, some of which present significant challenges for accurate SLAM. In environments with minimal structural information but rich in texture, such as long corridors, commonly used SLAM algorithms often fail. This issue is commonly referred to as the problem of geometry degeneracy. The lack of distinct geometric features makes it difficult for LiDAR-based systems to perform reliable scan registration and pose estimation. Methods that heavily rely on geometric features such as points, lines, and planes may not find enough distinctive features to estimate the relative motion between consecutive scans accurately. This problem occurs when the environment does not provide sufficient geometric constraints for accurate pose estimation. For example, in a long corridor, there might be enough planar features, but too few edge features to accurately estimate the robot's movement in all six Degrees of Freedom (DOF). This leads to the robot potentially losing its sense of direction, causing the SLAM system to fail. Furthermore, in natural or unstructured environments, such as forests or caves, the irregularity and complexity of the surroundings pose additional challenges. The uneven terrain and the presence of irregular objects make it difficult to extract consistent geometric features, which are essential for typical SLAM systems. The variability and unpredictability of these environments require more sophisticated approaches to ensure reliable localization and mapping.

Additionally, the problem of odometry drift exacerbates these challenges. Over time, small errors in motion estimation can accumulate, leading to significant deviations from the true

path of the robot. This drift can be particularly problematic in long-duration missions or in environments where GPS signals are unavailable or unreliable. To mitigate odometry drift, robust methods for loop closure detection and map optimization are necessary, allowing the robot to correct its trajectory by recognizing previously visited locations and adjusting its map accordingly.

In addition to the inherent challenges posed by the environment itself, the presence of moving objects can further complicate the SLAM process. Traditional SLAM methods typically assume that all objects in the environment are static. This assumption allows the extraction of feature points and the optimization of the distances between these features across two frames to estimate the movement of the robot. However, when there are dynamic objects in the environment, this assumption is violated. The presence of moving objects can lead to significant discrepancies in the optimization process, causing the estimated motion to deviate from the actual motion and resulting in the failure of the SLAM system.

For example, in urban environments where pedestrians, cyclists, and other vehicles are constantly in motion, the dynamic nature of these objects introduces errors in feature matching and pose estimation, which can severely affect the SLAM performance. These dynamic elements create a constantly changing environment, making it difficult to maintain an accurate and consistent map. The robot must be able to distinguish between static and dynamic objects and update its map accordingly to navigate safely and effectively.

In conclusion, achieving accurate and reliable real-time SLAM in unknown, unstructured, and dynamic environments remains a significant challenge in robotics. Addressing the issues of geometry degeneracy, dynamic objects, and odometry drift requires innovative approaches that combine robust feature extraction and advanced optimization techniques. By overcoming these challenges, we can significantly enhance the capabilities of autonomous robots, enabling them to navigate and explore complex environments more effectively.

1.2 Problem Statement

In this thesis, we are concerned with achieving accurate and reliable SLAM in unknown, unstructured, and dynamic environments. Endowing autonomous robots with such capability paves the way for more effective navigation and exploration in complex environments. In this thesis, we attempt to tackle the following core challenges:

1. **Geometry Degeneracy:** The lack of sufficient geometric constraints in environments with minimal structural information but rich in texture, such as long corridors or open fields, leads to unreliable scan registration and pose estimation. Traditional SLAM

methods often fail in these scenarios, where the robot can lose its sense of direction, causing the SLAM system to fail. Furthermore, in natural or unstructured environments, such as forests or caves, the irregularity and complexity of the surroundings pose additional challenges. The uneven terrain and the presence of irregular objects make it difficult to extract consistent geometric features, which are essential for typical SLAM systems. The variability and unpredictability of these environments require more sophisticated approaches to ensure reliable localization and mapping.

2. **Odometry Drift:** Over time, small errors in motion estimation can accumulate, leading to significant deviations from the true path of the robot. This problem, known as odometry drift, can be particularly problematic in long-duration missions or in environments where GPS signals are unavailable or unreliable. Odometry drift can cause the robot to lose track of its position, resulting in an inaccurate and inconsistent map. To mitigate odometry drift, robust methods for loop closure detection and map optimization are necessary. These methods allow the robot to recognize previously visited locations and correct its trajectory, ensuring the accuracy of the constructed map.

3. **Dynamic Objects:** Traditional SLAM methods typically assume that all objects in the environment are static. This assumption simplifies the extraction of feature points and the optimization of the distances between these features across two frames to estimate the movement of the robot. However, in dynamic environments such as urban areas, where pedestrians, cyclists, and vehicles are constantly in motion, this assumption is violated. The presence of moving objects introduces errors in feature matching and pose estimation, leading to significant discrepancies in the optimization process. These errors can cause the estimated motion to deviate from the actual motion, resulting in the failure of the SLAM system. The ability to distinguish between static and dynamic objects and update the map accordingly is essential for reliable navigation and mapping.

Given these challenges, it is evident that current SLAM systems are inadequate for handling the complexities of unstructured and dynamic environments. There is a need for innovative approaches that can address geometry degeneracy, effectively manage dynamic objects, and mitigate odometry drift. Developing such approaches will significantly enhance the capabilities of autonomous robots, enabling them to navigate and explore complex environments with greater reliability and accuracy.

This thesis aims to investigate and develop solutions to these problems, providing a more robust and adaptable SLAM system capable of operating in challenging environments. By ad-

dressing these critical issues, the research will contribute to the advancement of autonomous robotics, facilitating their deployment in a wide range of real-world applications.

1.3 Research Objectives

In light of the previous challenges, and motivated by the need for robust and reliable SLAM systems in complex environments, the research objectives we adopt in this thesis are:

O1: Ensure accurate and reliable real-time SLAM system in environments suffering from geometry degeneracy by achieving the following goals:

- Develop methods that can operate effectively in environments with minimal structural information and rich textures, such as long corridors or open fields.
- Implement robust feature extraction techniques to handle the irregularity and complexity of natural or unstructured environments.
- Deploy the proposed system on the robot onboard computer.

O2: Mitigate odometry drift to ensure long-term accuracy in SLAM by achieving the following goals:

- Develop robust loop closure detection methods to recognize previously visited locations and correct the robot's trajectory.
- Implement map optimization techniques that adjust the robot's map based on loop closure information, ensuring the accuracy of the constructed map.

O3: Effectively manage dynamic objects in SLAM processes by distinguishing between static and dynamic objects and tracking the movement of dynamic objects in real-time.

These objectives aim to address the core challenges faced by current SLAM systems, enhancing their capability to operate in unknown, unstructured, and dynamic environments. By achieving these goals, this research will contribute to the advancement of autonomous robotics, enabling more reliable and effective navigation and exploration in a variety of real-world applications.

1.4 Research Contributions

1. **FW1:** A real-time SLAM framework that addresses both geometry degeneracy and odometry drift. This framework includes methods capable of operating effectively in

environments with minimal structural information and rich textures, such as long corridors or open fields. It also implements robust feature extraction techniques to handle the irregularity and complexity of natural or unstructured environments, such as forests or caves. Additionally, the framework incorporates robust loop closure detection methods and map optimization techniques to mitigate odometry drift, ensuring long-term accuracy in SLAM. This work is described in detail in Chapter 4, where we develop the base framework with enhanced feature extraction techniques and loop closure methods, and we test the proposed SLAM system on the onboard computer (Jetson AGX Xavier). This work was published as:

- ***W. Du**, and G. Beltrame. "Real-time simultaneous localization and mapping with LiDAR intensity." In 2023 IEEE International Conference on Robotics and Automation (ICRA), pp. 4164-4170. IEEE, 2023.*

2. **FW2:** A dynamic objects detection and tracking framework that effectively manages dynamic objects. This framework includes methods to distinguish between static and dynamic objects in the environment, ego-motion estimation and objects tracking. These contributions are detailed in Chapter 5, where we present our dynamic object detection and tracking methods. This work was published as:

   - ***W. Du**, and G. Beltrame. "LiDAR-based Real-Time Object Detection and Tracking in Dynamic Environments." submitted to IEEE Transactions on Robotics (T-RO), 2024.*

These contributions collectively aim to enhance the robustness and adaptability of SLAM systems in unknown, unstructured, and dynamic environments. By addressing the core challenges of geometry degeneracy, odometry drift, and dynamic object management, this research advances the field of autonomous robotics, enabling more reliable and effective navigation and exploration in a variety of real-world applications.

1.5 Research Impact

We envision that our proposed methods will have a significant impact on the field of autonomous robotics, particularly in enhancing the robustness and reliability of SLAM systems in complex environments. The ability of robots to navigate and map unknown, unstructured, and dynamic environments reliably is crucial for various applications, including urban navigation, search and rescue missions, and environmental monitoring.

The framework **FW1**, which addresses geometry degeneracy and odometry drift, provides a solution to the fundamental challenges faced by SLAM systems in environments with minimal structural information and high texture. By ensuring accurate localization and mapping, **FW1** enhances the operational capabilities of autonomous robots in settings such as long corridors, open fields, forests, and caves. This framework's robust feature extraction and loop closure detection techniques ensure long-term accuracy and reliability, making it invaluable for extended missions in unstructured environments. For example, in urban search and rescue operations, where robots need to navigate through rubble and debris, **FW1** can provide the necessary accuracy to locate survivors and map hazardous areas.

On a larger scale, the ability to manage dynamic objects, as provided by **FW2**, is critical for the deployment of autonomous robots in urban environments and other settings with frequent movement of people and vehicles. This framework ensures that robots can distinguish between static and dynamic objects and update their maps in real-time to reflect these changes. Such capabilities are essential for applications like autonomous delivery services, where robots must navigate busy streets and avoid collisions with pedestrians and other vehicles. Additionally, **FW2** can be applied to autonomous vehicle navigation, enhancing safety and efficiency by accurately tracking and responding to the movement of surrounding objects.

Furthermore, in the context of general field applications, the enhanced SLAM system is crucial for the development of self-driving cars, enabling these vehicles to understand their surroundings in real time, navigate through dynamic environments, and avoid obstacles. In the field of aerial robotics, SLAM assists drones in navigating and mapping areas that are difficult to access by humans, such as in search and rescue missions, environmental monitoring, and agricultural surveying. In the field of Augmented Reality (AR), our methods can be used to create accurate and realistic virtual environments synchronized with the real world, enhancing user experiences and applications. Additionally, they can detect and analyze moving objects virtually, further enriching the AR experience.

In summary, the research contributions of this thesis are expected to enhance the reliability and adaptability of real-time SLAM systems in a wide range of applications, from urban navigation and search and rescue to environmental monitoring and agricultural management. By addressing the core challenges of geometry degeneracy, odometry drift, and dynamic object management, this research advances the field of autonomous robotics, facilitating their deployment in increasingly complex and dynamic real-world environments.

CHAPTER 2 LITERATURE REVIEW

This chapter will provide a comprehensive review of literature related to the key themes of this thesis. The chapter will be presented in following topics:

- SLAM Systems:
  - LiDAR-based SLAM systems: An in-depth discussion on cutting-edge LiDAR SLAM techniques, highlighting their strengths and limitations in different environments.
  - Intensity Aided LiDAR SLAM systems: Discussion on the incorporation of LiDAR intensity data to overcome the shortcomings of traditional LiDAR SLAM methods, and the recent development of intensity aided SLAM.
  - Visual SLAM systems: An exploration of Visual SLAM techniques, their feature extraction methods, and their applications in robotics.
  - LiDAR and Visual integration SLAM systems: An analysis of the integration of LiDAR and visual data for improved SLAM performance, current imporvements, and limitations.
- Object segmentation in Robotic perception:
  - Dynamic Object Detection and Tracking: Exploration of methods for dynamic objetc detection and tracking in various environments, and their importance in autonomous navigation systems.

2.1 SLAM Systems

SLAM is employed to enable robots to navigate through unknown environments while simultaneously creating a map of the explored area. Over the past few decades, significant advancements have been made in this field [1, 2, 8]. In 1990, Smith et al. [9] introduced the use of the Extended Kalman Filter (EKF) for estimating the relative positions of objects in an unknown environment, building a stochastic map to store these spatial relationships and their uncertainties. This pioneering work sparked widespread interest in the topics of localization and mapping among researchers.

Today, a variety of SLAM systems exist, including LiDAR-based SLAM, Visual SLAM, and systems that combine both LiDAR and visual data [10]. Many of these systems also

incorporate IMUs to enhance their accuracy and robustness. Given our focus on LiDAR and visual collaborative SLAM, we will first provide a brief overview of both LiDAR-based and Visual SLAM systems before delving into their integration.

2.1.1 LiDAR-based SLAM systems

The LiDAR-based SLAM system has seen extensive research and development in both theoretical and industrial contexts in recent years [11, 12]. LiDAR serves as a crucial sensor in these systems, providing essential data for mapping and localization. Additionally, some researchers incorporate Inertial Measurement Units (IMUs) as supplementary sensors [13] to enhance the accuracy of their SLAM systems. The primary task in these systems involves processing the point clouds collected by LiDAR to estimate the robot's trajectory.

A notable example of a LiDAR SLAM system is Cartographer, developed by Google. Cartographer utilizes the scan-to-submap strategy, incorporating loop closure detection and graph optimization to achieve high accuracy in real-time operations [14]. Another widely recognized system is Lidar Odometry and Mapping (LOAM) [11], developed by Zhang et al. from Carnegie Mellon University (CMU). LOAM is designed for real-time odometry and mapping, processing LiDAR point clouds to extract edge and planar points, which are used to register two consecutive scans at a frequency of 10 Hz for odometry and 1 Hz for mapping.

LeGO-LOAM [12] is another real-time odometry and mapping SLAM method that utilizes only the LiDAR of ground vehicles as a front-end sensor. LeGO-LOAM extracts ground planar and edge features, assigning them different labels to significantly reduce the number of features and enhance the speed of ego-estimation.

Additionally, Shan et al. [13] proposed LIO-SAM, a tightly-coupled LiDAR-inertial odometry system. LIO-SAM estimates odometry by integrating LiDAR data with a 9-axis high-frequency IMU. Results demonstrate that LIO-SAM is significantly more accurate than LOAM and can operate in real-time on platforms with limited computational resources.

DLO [15] is a direct LiDAR odometry algorithm developed by National Aeronautics and Space Administration (NASA) Jet Propulsion Laboratory (JPL) Team CoSTAR. It processes point clouds directly, achieving high accuracy in real-time, even on computationally-limited robot platforms. However, it is sensitive to dynamic objects.

KISS-ICP [16] presents a robust and accurate pose estimation system for robotic platforms, emphasizing simplicity and effectiveness. Contrary to many odometry systems that increase complexity for ego-motion estimation, KISS-ICP focuses on core elements, removing most parts to achieve a highly effective system. The approach relies on point-to-point ICP com-

bined with adaptive thresholding for correspondence matching, a robust kernel, a straightforward motion compensation method, and a point cloud subsampling strategy. This results in a system with minimal parameters that typically require no tuning for specific LiDAR sensors. The system performs comparably to state-of-the-art methods across various platforms.

TEASER [17] represents a significant advancement in robust 3D point cloud registration algorithms, addressing outlier correspondences with its Truncated Least Squares (TLS) cost function and robust graph-theoretic framework. It sequentially estimates scale, rotation, and translation despite their non-convex nature, backed by theoretical error bounds crucial for robustness. TEASER++ builds on this foundation, achieving speed and certifiability, outperforming traditional methods like Random Sample Consensus (RANSAC) [18] and heuristics in challenging scenarios. Its open-source implementation supports practical applications, fostering further research in certifiable perception for robotics and computer vision. However, the speed of TEASER++ will be affected by the increasing number of points. Specifically, as the number of points increases, the speed of TEASER++ decreases significantly.

In comparison with multi-line spinning LiDAR (Fig 2.1), Zhang et al [19] [20] developed robust and accurate localization and mapping algorithms using solid-state LiDAR with a limited field of view (FoV). FAST-LIO [19] is a computationally efficient and robust LiDAR-inertial odometry framework. It fuses LiDAR feature points with IMU data using a tightly-coupled Iterated Extended Kalman Filter (iEKF) [21,22] for robust navigation in fast-motion, noisy, or cluttered environments. A new Kalman gain formula reduces computational load by depending on the state dimension instead of the measurement dimension. Their experiments show that FAST-LIO outperforms state-of-the-art methods in terms of accuracy and robustness, even in challenging indoor environments.

As an imporvement, FAST-LIO2 is a fast, robust, and versatile LiDAR-inertial odometry framework. It improves accuracy and efficiency by registering raw points directly to the map without feature extraction and using an incremental k-dimensional tree (Incremental k-dimensional Tree (ikd-Tree)) for dynamic updates and rebalancing. In their experiments, FAST-LIO2 outperforms state-of-the-art systems in accuracy and computational load. It demonstrates efficiency up to 100 Hz, robustness in cluttered environments, and versatility across various LiDAR types and platforms.

FF-LINS [23] presents a frame-to-frame LiDAR-inertial navigation system for solid-state LiDARs, addressing the inconsistency of frame-to-map registration systems. Utilizing non-repetitive scanning patterns, the system constructs keyframe point-cloud maps for data association. It integrates LiDAR and IMU measurements via factor graph optimization with online calibration of extrinsic and time-delay parameters. Their experiments show that FF-

LINS achieves higher accuracy and robustness compared to state-of-the-art systems, with notable improvements in pose accuracy due to effective extrinsic and time-delay parameter estimation. Chen et al. present RELEAD [24], a LiDAR-based localization method designed to address scan-matching degradation in challenging environments. It employs constrained Error-State Iterated Extended Kalman Filter (ESIKF) [25] updates and integrates multisensor constraints via Graduated Non-Convexity (GNC) [26]-based graph optimization, even with outliers. The proposed robust Incremental Fixed Lag Smoother (rIFL) enhances optimization efficiency. Experiments show that RELEAD outperforms existing LiDAR-Inertial and LiDAR-Visual-Inertial odometry methods.

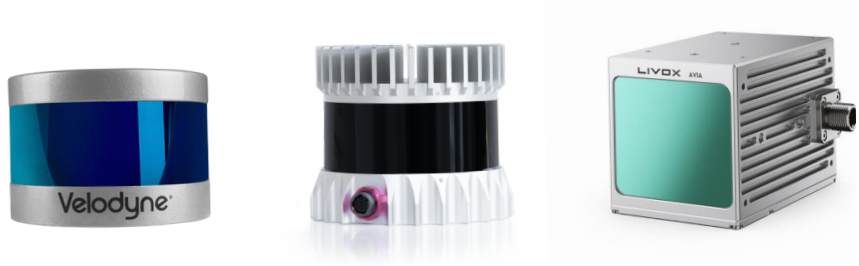


Figure 2.1 left: Velodyne LiDAR (spinning); Middle: Ouster LiDAR (spining); Right: Livox LiDAR (solid-state LiDAR)

Furthermore, loop closure detection is also a kernel part of the LiDAR SLAM system since the odometry will be drift during the long term running [27]. Most of the loop closure detection methods need to extract the features information from the sensor data [28]. Some of them will extract the semantic information from the raw sensor data. For instance, PointNet [29] and PointNet++ [30] leverage deep learning to directly learn deep point set features from 3D point clouds, enabling effective segmentation and semantic information extraction. Beyond feature extraction, SegMatch [31] offers a place recognition method by matching 3D segments. PointNetVLAD [32], which combines PointNet and NetVLAD [33], employs end-to-end learning to extract global features from point cloud frames. These global features are crucial for point cloud-based retrieval in place recognition tasks, as traditional methods often struggle to encode local features of a point cloud frame into a comprehensive global descriptor.

Ramezani et al. propose a Pose-Graph Attentional Graph Neural Network (P-GAT) for LiDAR place recognition. P-GAT compares (key)nodes between sequential and non-sequential sub-graphs, diverging from the frame-to-frame retrieval used in current state-of-the-art methods. It utilizes maximum spatial and temporal information between neighboring cloud descriptors, generated by an existing encoder, leveraging pose-graph SLAM. By incorporating

intra- and inter-attention mechanisms within a graph neural network, P-GAT connects point clouds in nearby Euclidean spaces and their feature space embeddings. Experimental results on large-scale public datasets highlight P-GAT's effectiveness in feature-sparse scenes and domain adaptation, with performance improvements over existing methods.

Scancontext [34] introduces a non-histogram-based global descriptor for global localization using 3D LiDAR scans. This method captures the 3D structure of environments directly, without prior training. Key features include a similarity score for distance calculation between scan contexts and a two-phase search algorithm for efficient, viewpoint-invariant loop detection. Evaluations on benchmark datasets show significant performance improvements.

LCDNet [35] introduces a novel approach for loop closure detection and point cloud registration in LiDAR SLAM systems. Unlike previous deep learning methods that underperform with reverse loops, LCDNet effectively detects loop closures by identifying previously visited locations and estimating six degrees of freedom transformations between current scans and maps. The network comprises a shared encoder, a place recognition head for extracting global descriptors, and a relative pose head using unbalanced optimal transport theory for end-to-end training.

OverlapNet [36] addresses loop closing in LiDAR-based SLAM systems by leveraging a deep neural network to analyze 3D laser scans from autonomous vehicles. It predicts image overlaps and relative yaw angles between scan pairs to detect loop closures. Integrating this method into an existing SLAM system improves mapping results.

LoGG3D-Net [37] is an end-to-end trainable architecture for 3D place recognition, enhancing re-localization in SLAM by learning robust global descriptors from point clouds. It introduces a local consistency loss to ensure local feature stability across revisits, improving overall descriptor repeatability.

Semantic information is not only used to the loop closure detection, but also for ego-motion estimation and mapping. Several deep learning-based LiDAR SLAM systems have achieved outstanding performance. SuMa++ [38] enhances LiDAR-based SLAM by integrating semantic information from 3D laser scans using a fully convolutional neural network. This approach labels scan points to create a semantically-enriched map, improving moving object filtering and projective scan matching. SuMa++ outperforms geometric methods, especially in dynamic environments with many moving cars.

LO-Net [39] is a deep convolutional network for real-time ego-motion estimation based on LiDAR data, trained end-to-end. It introduces a mask-weighted geometric constraint loss to learn effective feature representations and exploit sequential data dependencies. A scan-to-

map module enhances accuracy by using geometric and semantic information from LO-Net. In their experiments, LO-Net outperforms LOAM [11] and other learning-based methods.

SHINE-Mapping [40] uses sparse hierarchical implicit neural representations for large-scale 3D mapping from LiDAR data. It employs an octree structure to efficiently store features, converted to signed distance values via a shallow neural network. Based on the implicite representation, they also design a forgetting startegy for continue learning. This method improves accuracy, completeness, and memory efficiency, outperforming current 3D mapping techniques.

2.1.2 Intensity Aided LiDAR SLAM systems

Relying solely on LiDAR point cloud data may be insufficient for some SLAM systems, especially in situations where camera use is also impractical, such as in dark areas. In the meanwhile, LiDAR resolution is steadily increasing, we can generate much more clear and textured images from intensity information. Intensity data provides additional environmental information, such as surface reflectivity, which helps distinguish between different materials and objects. By integrating intensity data, SLAM systems can achieve greater accuracy and robustness in challenging environments.

In recent years, integrating intensity information into SLAM system has emerged as a novel approach among researchers [41–43]. Most of researchers tried to combine intensity information together with geometric information to improve the performance of SLAM system. Hewitt et al. [44] introduces a novel approach for incorporating intensity information into SLAM algorithms using LiDAR and time-of-flight (ToF) sensors. Unlike traditional methods that only utilize range and bearing data, this approach integrates an intensity model within a sparse bundle adjustment (SBA) framework. An observability analysis confirms the feasibility of this solution, and simulation results demonstrate its potential utility, particularly in environments where active sensors are more practical, such as mining and planetary exploration. Wang et al. [45] introduce intensity features into their SLAM system, using both intensity and geometric features to improve the performance of the SLAM system. This framework includes intensity-based front-end odometry estimation and back-end optimization. It surpasses traditional geometric-only LiDAR SLAM methods. Li et al. [46] extract intensity edge points in a solid-state LiDAR and present a robust, real-time LiDAR-inertial SLAM system SLAM system, addressing challenges posed by their small horizontal FoV and irregular scanning patterns in indoor environments. The method features a novel geometry and intensity-based feature extraction to handle degeneracy. It uses multi-weighting functions for pose optimization and an image processing-based map management module to

enhance efficiency and reduce outliers.

Minwoo et al. [47] propose an Intensity and Ambient Enhanced Lidar Inertial Odometry via Smoothing and Mapping (IA-LIO-SAM) framework, to address challenges in mapping unstructured construction environments. Building on the LIO-SAM [13] system, IA-LIO-SAM incorporates point intensity and ambient values to filter unnecessary feature points and improve the accuracy of the K-Nearest Neighbors (KNN) algorithm for point comparisons. COIN-LIO [48] is a LiDAR Inertial Odometry pipeline that enhances robustness in geometrically degenerate environments by integrating LiDAR intensity information with geometry-based point cloud registration. It projects LiDAR intensity returns into images and processes them to improve brightness consistency. A novel feature selection scheme detects uninformative directions in point cloud registration and selects complementary image patches. The method combines photometric error minimization with inertial measurements and point-to-plane registration in an iEKF. Dai et al. [49] propose an intensity-enhanced LiDAR SLAM framework tailored for unstructured environments where traditional geometric features are sparse. The method adaptively extracts intensity features and constructs intensity constraints based on degradation detection, alongside a multi-resolution intensity map construction technique.

Some researchers have also proposed using intensity data to assist in loop closure detection. Wang et al. [50] introduce a the Intensity Scan Context (ISC), a novel global descriptor for loop closure detection in SLAM, leveraging both geometry and intensity data from LiDAR scans. To enhance efficiency, the method uses a two-stage hierarchical re-identification process involving fast geometric relation retrieval and intensity structure re-identification. [42,51] proposed novel visual place recognition methods using LiDAR intensity information for LiDAR-based SLAM systems. By converting high-resolution 3D LiDAR scans and stable intensity measurements into 360° panoramic range images, visual place recognition strategies can be applied to LiDAR data. This method focuses on finding loop closures and reducing odometry drift with the help of intensity information. Zhang et al. [52] introduces a novel LiDAR-SLAM framework that incorporates intensity information to enhance accuracy in environments with sparse features. It combines geometric and intensity features for matching consecutive scans and uses a self-adaptive feature selection strategy for efficient odometry estimation. The method includes a new intensity cylindrical-projection shape context (ICPSC) descriptor for loop closure detection, supported by a row-column similarity estimation and double-value loop candidate verification.

2.1.3 Visual SLAM systems

Thanks to advancements in Central Processing Unit (CPU) and graphics processing unit (GPU) technology, we now have greater computational power to handle graphic processing. Additionally, high-resolution cameras have become more affordable and compact. As a result, visual SLAM has rapidly developed over the past decades and can now be integrated into smaller robots.

Visual SLAM systems typically use cameras as their primary sensors. These cameras can be categorized into several types, including monocular cameras, RGB-D cameras, stereo cameras, event cameras, and others [10]. Additionally, visual SLAM can be classified into dense or semi-dense SLAM and sparse SLAM, based on whether they use direct methods or feature-based methods.

ORB SLAM [53], ORB SLAM2 [54], and ORB SLAM3 [1] are sparse SLAM systems that typically begin by extracting feature points from camera data, such as Oriented FAST and rotated BRIEF (ORB) [55], Scale-Invariant Feature Transform (SIFT) [56] features. These systems can compute the camera trajectory and create a sparse 3D reconstruction of large environments with high accuracy due to their ability to detect loops in extensive scenes and perform real-time global re-localization. Notably, they can save maps offline and reload them for subsequent sessions. This global re-localization capability allows these systems to merge current maps with offline maps, enhancing the overall mapping process. VINS-Mono [57] is a robust monocular visual-inertial SLAM system using a single camera and low-cost IMU. It initializes with a robust procedure and uses nonlinear optimization to fuse IMU measurements and feature observations for accurate odometry. The system includes a loop detection module for efficient relocalization and 4-DOF pose graph optimization for global consistency. It also supports map reuse and merging via global pose graph optimization. VINS-Fusion [58] extends VINS-Mono by a sensor fusion framework for global pose estimation that combines local sensors (camera, IMU, LiDAR) with global sensors (GPS, magnetometer, barometer) to achieve accurate and drift-free state estimation. The framework uses pose graph optimization to align local estimations from VO/VIO approaches with global sensors, eliminating accumulated drifts. Effloc [59] is an efficient Vision Transformers (ViTs) [60, 61] for 6-DOF camera relocalization, using hierarchical layout and sequential group attention to enhance memory and computational efficiency. It outperforms methods like AtLoc [62] and MapNet [63] in large-scale outdoor scenarios, offering accurate, end-to-end trainable, and efficient pose estimation.

For semi-dense SLAM, EVO [64] is an event-based visual odometry algorithm that leverages the exceptional properties of event cameras to track rapid camera movements while con-

structing a semi-dense 3D map of the environment. SVO [65] employs a direct method for visual odometry, utilizing feature-based methods for joint optimization, which significantly enhances speed while maintaining competitive accuracy. Additionally, SVO can be easily adapted to various camera types, including fisheye and catadioptric cameras. CNN-SVO [66] builds upon SVO by incorporating a single-image depth prediction network, which significantly reduces the depth uncertainty of initialized map points, thus improving the reliability of SVO. Dircet Sparse Odometry (DSO) [67] calculates visual odometry directly based on sparse and direct structure and motion formulation without relying on keypoint detectors or descriptors.

Dense SLAM is useful for creating dense maps that can be directly employed in path planning. DTAM [68] is a type of dense SLAM that relies on dense, per-pixel methods rather than feature extraction. It facilitates real-time camera tracking and reconstruction using an RGB camera, GPU hardware, and a novel non-convex optimization framework. DVO [69] achieves higher pose accuracy by minimizing both photometric and depth errors across all pixels. Dynamic Fusion [70] enables real-time reconstruction of non-rigidly deforming scenes. It does not require a template or prior scene model, making it applicable to a wide range of moving objects. MLM SLAM [71] reconstructs dense 3D geometry online without the need for a GPU. DeepFactors [72] is a real-time probabilistic dense SLAM system that uses a learned compact depth map representation. ∇SLAM [73] is a fully differentiable dense SLAM system, representing a novel approach that integrates representation learning techniques with classical SLAM methods.

2.1.4 LiDAR and Visual integration SLAM systems

Fusing LiDAR and visual data for SLAM has become a significant trend in recent years. The integration of these data sources enhances the robustness and accuracy of SLAM systems, as each sensor complements the other's strengths and mitigates its weaknesses. LiDAR delivers precise depth information, while cameras provide detailed color and texture data. By combining these two types of information, SLAM systems can achieve superior performance across diverse environments. PointFusion [74] directly processes raw image and LiDAR data using a Convolutional Neural Network (CNN) and PointNet architecture, then fuses the results for 3D object detection. LIC-Fusion [75] is a tightly-coupled LiDAR-inertial-camera fusion algorithm within the MSCKF framework, combining sparse visual features with efficiently extracted LiDAR points and IMU data to detect and track edge and surface feature points from LiDAR scans. Additionally, LIC-Fusion performs online spatial and temporal sensor calibration for all three asynchronous sensors. V-LOAM [76] uses both camera and

LiDAR to estimate odometry, with two sequential processors: the first runs visual odometry at 60Hz to handle rapid motion, and the second uses LiDAR odometry at 1Hz to refine motion estimation, ensuring low drift and robustness in poor lighting conditions. This algorithm demonstrates high accuracy even without post-processing.

LVI-SAM [2] features two tightly-coupled subsystems: a visual-inertial system (VIS) and a LiDAR-inertial system (LIS). The LIS enhances VIS accuracy by providing depth information, while VIS offers an initial position guess for LIS's scan-matching. Loop closures are initially detected by VIS and further refined by LIS. This framework operates in real-time for state estimation and map-building, achieving high accuracy and robustness.

R2LIVE [77] is a real-time, tightly-coupled odometry estimation framework that fuses LiDAR, IMU, and camera data for robust and accurate state estimation. It comprises a filter-based odometry component and a factor graph optimization component. Unlike traditional multi-line spinning LiDARs, it utilizes solid-state LiDARs with a small FoV, maintaining robustness even during aggressive motion and sensor failures. R3LIVE [78], developed from R2LIVE, includes LiDAR-inertial odometry (LIO) and visual-inertial odometry (VIO) subsystems. This framework supports real-time SLAM and the reconstruction of dense, precise, RGB-colored 3D maps.

Self-VLO [79] integrates monocular images with sparse depth maps generated by 3D LiDAR points to improve ego-motion estimation. It uses a two-pathway encoder to extract and fuse features from both modalities, employing a siamese architecture with adaptively weighted flip consistency loss for self-supervised learning.

LHMap-loc [80] introduces a novel approach for monocular localization using pre-built LiDAR point cloud maps, addressing challenges like map storage and robustness in large scenes. It compresses LiDAR maps by generating offline heat point clouds and uses an online end-to-end pose regression network based on optical flow estimation and spatial attention for real-time visual localization.

2.2 Dynamic Object Detection and Tracking

2.2.1 Map-based methods

Map-based methods for the detection of dynamic objects can be categorized into three primary types: ray-tracing methods, visibility-based methods, and occupancy-based methods.

Ray-Tracing Methods

Ray-tracing methodologies simulate LiDAR sensor behavior by tracing rays from the LiDAR sensor outward into the environment [81,82]. Rays are traced through either the grid map [83] or voxel map [84] to identify dynamic objects. Should a ray traverse a grid cell or voxel, this indicates that the cell is empty. If an object obstructs the ray within this empty cell, it signifies that the object is dynamic.

Azim and Aycard [81] used an octree-based occupancy grid map to represent the dynamic surroundings of the vehicle and identified moving objects by noting discrepancies across scans. If the ray were reflected, then that volume would be occupied. If the ray were to traverse the volume, then it would be free. Peopleremover [82] efficiently removes dynamic objects from 3D point cloud data by constructing a regular voxel occupancy grid and traversing it along the sensor's line of sight to the measured points to identify differences in voxel cells between different frames, which correspond to dynamic points. It is applicable to both mobile mapping and terrestrial scan data, aiming to produce a clean point cloud devoid of dynamic objects, such as pedestrians and vehicles. This process conservatively removes volumes only when they are confidently identified as dynamic, thus preserving the integrity of static elements in the point cloud data. Khronos [85] introduces a method for detecting dynamic objects in robotic vision systems using ray tracing. Instead of relying on volumetric data, it captures changes through surface representations and background and robot poses. By creating a library of rays between observed background vertices and robot positions, stored in a global hash map for efficiency, the method dynamically detects objects without real-time free-space mapping. Object detection is achieved by computing distances to rays, determining occlusion, consistency, or absence. This technique offers a practical approach to real-time object detection, balancing computational efficiency with accuracy. However, it perform poorly in dynamic environments with large open spaces with sparse surfaces. M-detector [86] is a motion event detection system inspired by the human visual system's Magnocellular cells, known for their motion sensitivity due to larger sizes and faster processing capabilities. Using ray tracing and the principle of occlusion, it detects moving objects by observing how they interrupt the LiDAR sensor's laser rays, effectively occluding previously visible backgrounds or recursively occluding themselves. The system processes inputs either as individual points or frames, applying three parallel occlusion tests to determine motion events. This approach allows for low-latency, high-accuracy detection without requiring extensive training datasets. M-detector's design ensures it is highly generalizable, capable of detecting various object sizes, colors, and shapes across different environments with minimal computational resources, making it suitable for a wide range of robotic applications.

However, Ray-tracing methods demonstrate a significant sensitivity to the accuracy of odometry and incur substantial computational expenses during the construction of maps and and as the map size gradually increases [87–90]. Additionally, these methods lead to increased computational times [85] with prolonged operation and prove unsuitable for real-time applications.

Visibility-Based Methods

Visibility-based algorithms operate on a fundamental physical premise: light travels in straight lines, and if two points, one nearer and the other farther, are detected on the same ray, the nearer point has to be dynamic [89, 91, 92] to allow visibility of the farther point. The core concept involves recognizing dynamic points by examining the geometric variances between a query range image and its equivalent map range image. In essence, if a pixel's range value in the query image surpasses the one in the map image, it indicates an occlusion at the map image pixel's location according to these methods. Therefore, points mapped onto these pixels are deemed dynamic [88].

Removert [93] introduces a novel method for the removal of dynamic objects from LiDAR point cloud data. This method exploits the visibility of points within the LiDAR data to identify dynamic objects. By comparing the visibility of points in the current frame to that in the map, the method can detect dynamic objects and subsequently remove them from the point cloud data. This approach proves effective in removing dynamic objects from LiDAR data and enhancing the accuracy of static object detection. However, it is an offline method and is not suitable for real-time applications. ReFusion [94] creates a dense mapping that can effectively handle dynamic environmental elements without relying on specific dynamic object class detection, using Truncated Signed Distance Field (TSDF) and voxel hashing on GPU. The method detects dynamic objects by leveraging residuals from the registration process and the explicit representation of free space in the environment, achieving geometric filtering of dynamic elements. However, maintaining a dense TSDF mapping can be computationally expensive and memory-intensive.

The accuracy of the pose estimation is critical to the performance of the visibility-based methods since they need to retrieve the corresponding points in the map according to the odometry information. Moreover, false negatives, which include but are not limited to self-interference within point clouds and false positives caused by parallel point discrepancies, misjudged occluded points, and contact point errors, directly undermine the reliability of dynamic points extraction [87]. Additionally, the invisibility of static points, although less common than false negatives, poses a more challenging problem to solve. For instance, when

dynamic obstacles continuously block part or all of the LiDAR's rays of light, the LiDAR fails to detect the static objects behind these dynamic obstacles, leading to a scenario where the dynamic obstacle's point cloud is never identified and filtered out [88].

Occupancy-Based Methods

Occupancy-based methods employ statistical models to estimate the likelihood of space being occupied, facilitating robust mapping of environments with dynamic or uncertain elements.

Dynablox [95] is a real-time, map-based dynamic object detection system. It incrementally estimates high-confidence free-space areas and leverages enhanced the Truncated Signed Distance Field (TSDF) [96] method for volumetric mapping. This approach enables robust detection of dynamic objects without assumptions about their appearance or the structure of the environment. The method demonstrated an Intersection over Union (IoU) of 86% and processed data at 17 frames per second (FPS) on a laptop-grade CPU in real-world datasets, surpassing appearance-based classifiers and approximating the performance of certain offline methods. However, the study also acknowledges certain limitations, including challenges in detecting extremely thin and sparsely measured objects, such as shades, due to the voxel-based map representation. It also relies on prior odometry to place points into map spaces. ERASOR [87], developed by Lim et al., effectively removes dynamic objects from 3D point cloud maps using a novel approach centered around the concept of Region-Wise Pseudo Occupancy Descriptor (R-POD) and Region-wise Ground Plane Fitting (R-GPF). Pseudo occupancy is a type of occupancy map that expresses the occupancy of unit space and then discriminate spaces of varying occupancy. By focusing on the fundamental observation that most dynamic objects in urban environments make contact with the ground, ERASOR employs pseudo occupancy to evaluate the occupancy status of space units, distinguishing between static and dynamic areas. The method then utilizes R-GPF to accurately segregate static points from dynamic points within these identified spaces. This process allows for the efficient removal of dynamic objects from the point cloud data, ensuring a cleaner, more static map for improved navigation and localization performance. ERASOR2 [88] is an instance level dynamic object removal system that can effectively remove dynamic points from 3D point cloud maps. It uses Pseudo Occupancy Grid Map to represent whether the regions are temporally occupied or not by calculating the probability of those regions. It also employs instance segmentation [97] estimates to identify and exclude dynamic points at the instance level, ensuring that static points are preserved with minimal loss. This approach allows for the precise removal of dynamic objects from the map, enhancing the map's utility for navigation and planning tasks. ERASOR2 [88] is an instance level dynamic object

removal system that can effectively remove dynamic points from 3D point cloud maps. It uses Pseudo Occupancy Grid Map to represent whether the regions are temporally occupied or not by calculating the probability of those regions. It also employs instance segmentation [97] estimates to identify and exclude dynamic points at the instance level, ensuring that static points are preserved with minimal loss. This approach allows for the precise removal of dynamic objects from the map, enhancing the map's utility for navigation and planning tasks.

Nevertheless, these methods still require maintaining a map of the environment, which is memory-intensive and computationally demanding, especially for long runs.

2.2.2 Segmentation-Based Methods

Segmentation-based methods focus on identifying dynamic objects by semantically segmenting the point cloud and analyzing the movement of points or objects over time. These methods are particularly effective in detecting dynamic objects in complex environments with unidentified objects.

Thomas et al. [98] proposed a self-supervised learning method for dynamic object detection in LiDAR point clouds. They first projected each frame of point cloud into 2D grids. Then, they stacked the 2D frames into 3D according to the time sequence. Finally, they used a KPConv network [99] 3D back-end and a three levels U-Net [100] 2D front-end to predict the future time steps of spatiotemporal occupancy grid maps. Chen et al. [101] used the range image to represent the point cloud data and combined the current frame and residual images generated between current frame and the last few frames. After then, they used a range projection-based segmentation CNNs [102–104] to predict the moving objects.

As an upgrade to [98], Sun et al. [105] extended the SalsaNext [104] by adopting a dual-branch (a range image branch and a residual image branch) and dual-head (an image head and a point head) architecture to separately deal with spatial and temporal information. After then, they used a 3D sparse convolution to refine the prediction results. In this coarse-to-fine framework, the network can predict the moving objects more accurately and efficiently. Mersch et al. [106] combined a sequence of LiDAR point cloud into a voxelized se 4D point cloud. Then, they feeded the combined 4D occupancy grid to a modified MinkUNet14 [107] to extract spatio-temporal features and predicted the confidence score for each point.

These methods are effective in detecting dynamic objects; however, they predominantly depend on a substantial volume of annotations for the dynamic objects in the training data, which may not always be accessible. Furthermore, segmentation-based methods incur signif-

icant computational costs and frequently yield unsatisfactory results in unfamiliar environments. Consequently, a need persists for a lightweight and label-free method for dynamic object detection in real-time.

CHAPTER 3 RESEARCH APPROACH AND THESIS ORGANIZATION

In this thesis, we develop two frameworks for LiDAR-based SLAM system that come together to form a unified, robust solution for complex unknown environments. The integration of these frameworks addresses both the challenges of maintaining real-time localization in degenerate scenarios and the real-time identification of dynamic objects. This combined approach results in a highly reliable SLAM system capable of operating in complex and ever-changing environments.

In this chapter, we provide an overview of the key deliverables of the thesis and a succinct outline of the primary concepts discussed. Furthermore, we align these deliverables with the research objectives established in Chapter 1. In the last, we conclude with a detailed description of the overall thesis structure.

In this thesis, we develop two frameworks for LiDAR-based SLAM system:
(**FW1**) Maintain Localization in Degenerate Scenarios
(**FW2**) Identify Dynamic Objects in Real-Time

FW1: Maintain Localization in Degenerate Scenarios

In this framework, we are interested in scenarios in which the environment lacks sufficient structural features or is poorly illuminated, making it challenging to estimate the movement of robots. The purpose of this framework is:

- To propose a method that utilizes intensity information only from LiDAR to extract texture features from the environment. These extracted features are then employed to estimate the robot's ego-motion, providing a more reliable localization solution in such challenging scenarios. (**O1**)
- Provide a full SLAM pipeline that take the advantages of intensity information from LiDAR sensors to improve the loop closure detection and pose graph optimization. (**O2**)

O1 The development of **FW1** should ensure robot localization in challenging environments, where traditional methods may fail due to the lack of structural features or poor illumination. In developing **FW1**, we begin by creating a basic version of the framework in Chapter 4, where we use LiDAR-generated point clouds and intensity images to extract features. These features are used for scan registration to estimate robot movement. The system is designed to handle geometric degeneracy by leveraging intensity-based features to ensure robust localization.

O2 The chapter 4 will also discuss the integration of these methods into a full SLAM pipeline, demonstrating how the extracted intensity features improve loop closure detection and pose graph optimization. This ensures continuous and accurate localization, even in environments with minimal geometric features.

FW2: Identify Dynamic Objects in Real-Time

In this framework, we focus on scenarios where autonomous robots navigate through dynamic environments, requiring robust detection and tracking of moving objects. The purpose of this framework is:

O3
- To develop advanced techniques for real-time dynamic object detection and tracking, utilizing intensity-based ego-motion estimation to assist identify and track moving objects such as pedestrians.
- To maintain the accuracy of detecting dyanmic objects despite odometry drift..

For **FW2**, we design and implement techniques that use intensity information from LiDAR to detect and track dynamic objects in real-time. This involves extracting foreground features from the intensity data and clustering them to identify moving objects. The framework is built to be efficient, utilizing parallel computation to ensure reliable performance in complex and dynamic environments.

In Chapter 5, we demonstrate the application of **FW2** methods in dynamic environment. We show how the intensity-based foreground objects features extraction improves the detection and tracking of dynamic objects, enhancing the robots' ability to navigate safely and efficiently in dynamic environment.

Document Layout

This thesis is organized as follows: in Chapter 1, we provide an introduction in which we discuss the context and motivation of our work, as well as the problem statement and research objectives. In Chapter 2, we review the literature on several relevant topics, including vari-

ous methods for LiDAR-based SLAM and dynamic object detection. Chapter 4 focuses on the development of **FW1**, which aims to maintain localization in degenerate scenarios. We introduce techniques for using intensity information from LiDAR to extract texture features and ensure robust localization. This chapter details the methods for feature extraction, scan registration, and pose graph optimization. Chapter 5 presents the development of **FW2**, focusing on detecting dynamic objects in real-time. We detail the methods for extracting and clustering foreground features from LiDAR intensity data and showcase the system's performance in dynamic environments. Experimental results using dynamic scenarios are presented to validate the effectiveness of **FW2**. In Chapter 6, we provide a discussion of the work presented in the thesis, highlighting the strengths and contributions of the developed frameworks. We also address some of the challenges encountered during the development of **FW1** and **FW2** and provide insights into the solutions implemented. Chapter 7 concludes the thesis, summarizing the main findings and contributions. We discuss the current limitations of the frameworks and propose potential directions for future research to further enhance the capabilities of LiDAR-based SLAM and dynamic object detection systems.

CHAPTER 4 ARTICLE 1: REAL-TIME SIMULTANEOUS LOCALIZATION AND MAPPING WITH LIDAR INTENSITY

Preface: In order to enable robots to localize themselves in unknown environments, they must often rely on different sensors depending on the availability of environmental information. In cases where the environment lacks sufficient structural features or is poorly illuminated, cameras may not be viable for localization. Under these conditions, LiDAR sensors become crucial. Typically, the distance data from LiDAR is used to estimate the robot's movement. However, this approach often fails in less structured environments because it lacks sufficient edge and plane features. To address this challenge, we propose a method that utilizes intensity information only from LiDAR to extract texture features from the environment. These extracted features are then employed to estimate the robot's ego-motion, providing a more reliable localization solution in such challenging scenarios.
Based on this concept, we developed a comprehensive SLAM system. This system encompasses several key components: intensity feature extraction, feature matching, intensity-based loop closure detection, and pose graph optimization. We evaluated our system using both public datasets and the Spot robot. The experimental results demonstrate that our system achieves real-time performance and outperforms state-of-the-art methods in terms of accuracy and reliability. We believe that utilizing the intensity information from LiDAR sensors is a promising approach for SLAM in challenging environments.

Full Citation: Du, Wenqiang, and Giovanni Beltrame. "Real-time simultaneous localization and mapping with LiDAR intensity." In 2023 IEEE International Conference on Robotics and Automation (ICRA), pp. 4164-4170. IEEE, 2023. Published on July 4, 2023.

DOI: 10.1109/ICRA48891.2023.10160713

Abstract: We propose a novel real-time LiDAR intensity image-based simultaneous localization and mapping method, which addresses the geometry degeneracy problem in unstructured environments. Traditional LiDAR-based front-end odometry mostly relies on geometric features such as points, lines and planes. A lack of these features in the environment can lead to the failure of the entire odometry system. To avoid this problem, we extract feature points from the LiDAR-generated point cloud that match features identified in LiDAR intensity images. We then use the extracted feature points to perform scan registration and estimate

the robot ego-movement. For the back-end, we jointly optimize the distance between the corresponding feature points, and the point to plane distance for planes identified in the map. In addition, we use the features extracted from intensity images to detect loop closure candidates from previous scans and perform pose graph optimization. Our experiments show that our method can run in real time with high accuracy and works well with illumination changes, low-texture, and unstructured environments.

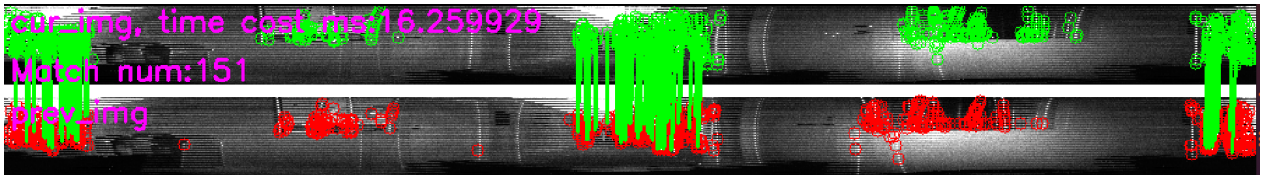
4.1 Introduction

Simultaneous Localization and Mapping (SLAM) is a fundamental problem in robotics. SLAM is the process of building a map of the environment and in the meanwhile, tracking the robot's pose on the generated map. There are many kinds of SLAM methods, such as visual SLAM [57], Light Detection and Ranging (LiDAR) SLAM [11], visual and LiDAR SLAM [2], visual-intertial and ranging SLAM (VIR-SLAM) [108] and so on. In this paper, we focus on the LiDAR SLAM.

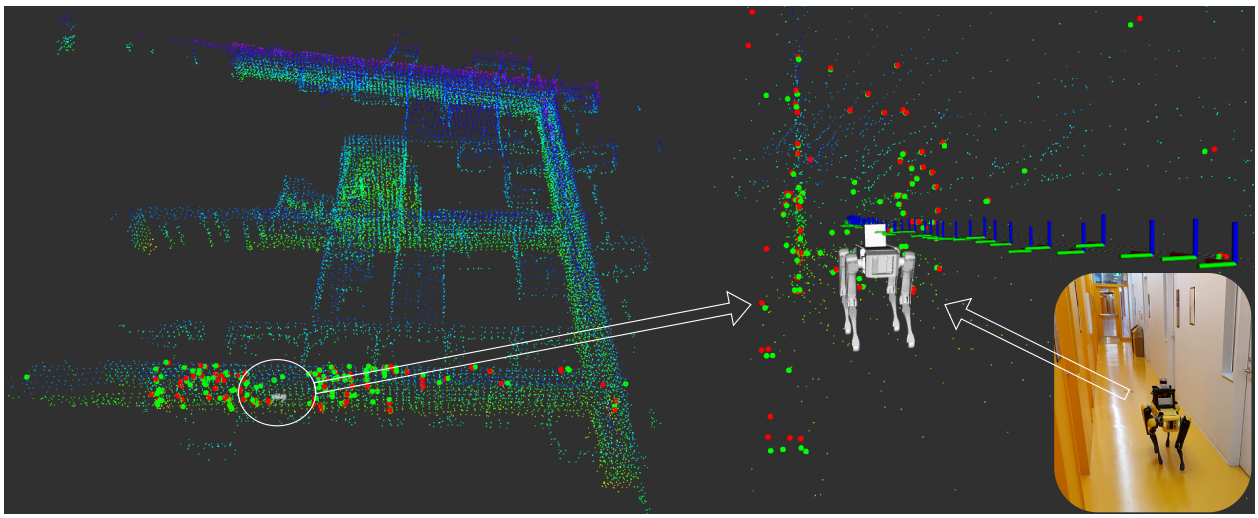
Numerous SLAM methods based on LiDAR [2, 12, 13, 109] have been proposed over the past few years. Most of them are based on the geometric features (e.g., edges and planes [11,12]) of LiDAR point clouds. These methods are robust and accurate in most structured environment. However, when the environment has little structure, these methods can suffer from geometric degeneracy and fail [8, 110]. For example, in a long corridor or a cave environment, there might be enough plane features, but too few edge features to estimate the relative ego-motion movement between two consecutive scans. Unfortunately, aligned plane features (like in the corridor) alone are not sufficient to estimate the robot pose in six degrees of freedom (6DOF), meaning the robot could lose the sense of forward or backward movement.

To solve this problem, one needs an additional reference to constrain the sixth degree of freedom. One possible way is to extract feature points from textural information. Recently, many researchers tried to add intensity information to augment [46] or assist [45] LiDAR SLAM systems, improving their performance. However, these feature points from textural information are already presented in 3D space and theoretically are enough to constrain the 6DOF movement. Why not directly perform scan registration with these feature points? We propose a pure LiDAR intensity SLAM method which directly extracts feature points from intensity images and perform scan registrations to estimate the robot's ego-movement. Our contributions are:

- A novel real-time LiDAR intensity image-based SLAM system, aiming at solving the geometric degeneracy problem;



(a) Feature points in the current scan (top) and the previous scan (bottom) are matched with ORB features from intensity images.



(b) The matched feature points are used to perform the scan registration.

Figure 4.1 The matched 3D points from two consecutive scans and their corresponding feature points. The points in **(b)** are the 3D points that are extracted from point clouds according to the indexes of matched features from **(a)**. The red points represent matched points of the previous frame, and the green points stand for matched points in the current scan. Those points are then used for scan registration to estimate the relative poses between two consecutive frames.

- Combining the benefits of visual SLAM systems with those of LiDAR SLAM systems, without suffering from blurring or illumination changes;
- A lightweight front-end due to fewer feature points and adding ground plane constraint and LiDAR Bundle Adjustment (BA) to the back-end;
- Intensity-based loop closure detection and pose graph optimization;

4.2 Related Work

In the past decades, many researchers worked in SLAM and achieved many great results. In 1990, Smith et al. [9] firstly used extended Kalman filter (EKF) to estimate the relative position among objects in an unknown environment and build a stochastic map to store these estimations of spatial relationships and their uncertainties. Nowadays, there are many kinds of SLAM systems [10], many using an Inertial measurement unit (IMU), cameras, and/or LiDARs.

LiDAR-based SLAM systems have been well studied in both theory and industrial applications in recent years. One representative work is LOAM [11], a real-time method for estimating odometry and mapping. LOAM only consumes a LiDAR's point cloud and then extracts edge and planar points from this point cloud to register consecutive scans at 10Hz and providing map updates at 1Hz. Another popular work is Cartographer [14], which uses a scan-to-submap matching strategy with loop closure detection and graph optimization, achieving high accuracy even in real-time. LeGO-LOAM [12] is also a real-time odometry and mapping SLAM method, which uses only the LiDAR of ground vehicles as a front-end sensor. LeGO-LOAM extracts ground planar and edge features and assigns them different labels, which significantly decreases the number of features. Shang et al. [13] proposed LIO-SAM, which is a tightly-coupled LiDAR-inertial odometry system. LIO-SAM can estimate the odometry by using LiDAR and a 9-axis high-frequency IMU. Shang et al. results showed that LIO-SAM is much more accurate than LOAM and can be run in real-time on a computationally limited platform.

Differentiating from these feature-based odometry methods, Chen et al. [111] proposed a direct method to estimate the odometry using LiDAR and IMU. Their results showed that their method is more accurate than a feature-based approach. DLO [111] is another direct LiDAR odometry algorithm that can match consecutive frames of point cloud directly with high accuracy in real-time for computationally-limited robot platforms, but it is sensitive to dynamic objects. Lin et al. [112] [20] developed a robust and accurate SLAM system using solid-state LiDAR with small field of view (FoV).

In addition to traditional algorithms, some deep learning-based LiDAR SLAM [35, 113, 114] systems also achieved good performance. PointNet [29] and PointNet++ [30] leverage deep learning techniques to directly extract semantic features from 3D point clouds, enabling their use in segmentation and extraction of semantic information. In addition to feature extraction, deep learning was used in SegMatch [31] for place recognition matching 3D segments. PointNetVLAD [32] is a combination of PointNet and NetVLAD [33], which uses end-to-end learning to extract global features from a frame of point cloud, which are very useful for place recognition.

Recently, as LiDAR resolution is steadily increasing, we can generate much more clear and textured images (see Fig. 4.1a) from intensity information. In recent years, integrating intensity information into their SLAM system has emerged as a novel approach among researchers [41–43]. Wang et al. [45] introduce intensity features into their SLAM system, using both intensity and geometric features to improve the performance of the SLAM system. Li et al. [46] extract intensity edge points in a solid-state LiDAR-inertial SLAM system, while [42, 51] proposed a novel visual place recognition method using LiDAR intensity information.

In this paper, we propose a novel lightweight LiDAR odometry method which directly matches 3D feature points extracted from intensity images. This method can merge both the feature tracking ability of visual SLAM and LiDAR's high accuracy. We also propose an intensity image-based back-end, including additional constraints between intensity features and using intensity information to detect loop closures.

4.3 Method

The pipeline of our proposed method is shown in Fig. 4.2. In our system, a LiDAR generates a point cloud within 100ms that is called a frame or a scan. To estimate the movement of the robot, we need to calculate the relative pose between consecutive frames. We use intensity odometry to implement this procedure in the front-end. In addition, we use the term "odometry" to describe the relative pose between the current frame and the initial frame. The odometry from the front-end is generally inaccurate, so we need to use a back-end to optimize the odometry. In the back-end, we use the scan-to-map method [11] and LiDAR BA [115] to correct the drift. However, map optimization is generally not enough for removing the accumulated drift, and we add loop closure detection as an additional means to reduce drift. In our case, we perform loop closure detection on the LiDAR intensity image and use pose graph optimization to update the trajectory and generate the final trajectory of the robot.

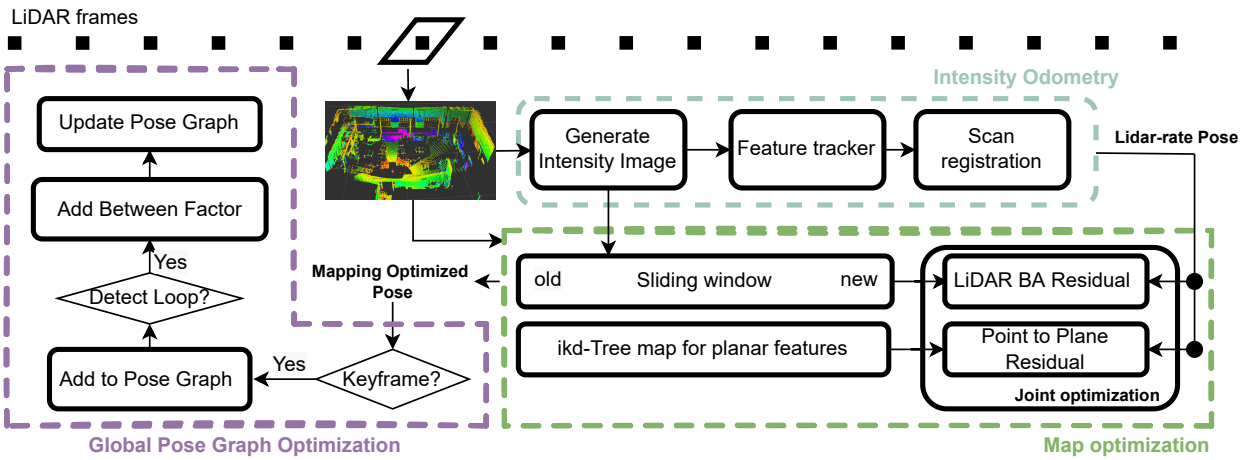


Figure 4.2 System overview of the proposed method. The whole system consists of three parts, including intensity odometry, map optimization, and pose graph optimization. The intensity odometry part is the core of the proposed method. It consists of intensity image generation, feature tracking, and scan registration. The map optimization corrects the drift by jointly minimizing both LiDAR BA residual and point to map plane residual. Pose graph optimization corrects the whole trajectory by adding loop constraints.

4.3.1 Intensity Odometry

Assume that we have two consecutive frames of point clouds $\mathcal{X} = \{\mathbf{X}_1, \mathbf{X}_2, \cdots, \mathbf{X}_J\}$ and $\mathcal{Y} = \{\mathbf{Y}_1, \mathbf{Y}_2, \cdots, \mathbf{Y}_I\}$ from a LiDAR: one direct way of estimating the relative pose is directly applying an Iterative Closest Point (ICP) algorithm [116] to calculate the rotation matrix $\mathbf{R}$ and $\mathbf{T}$:

$$\arg\min_{\mathbf{R},\mathbf{T}} \sum_{\mathbf{X}_j \in \mathcal{X}, \mathbf{Y}_i \in \mathcal{Y}} \| \mathbf{Y}_i - \mathbf{R}\mathbf{X}_j - \mathbf{T} \|^2 \tag{4.1}$$

However, this method usually consumes significant time and computation resources [11]. To reduce the computation cost, we need to extract representative points for the scan registration, thereby reducing the number of points used for optimization (up to a limit: we still need to maintain the original relationships between two frames).

In order to decrease the number of points used for optimization, Zhang et al. [11] tried to extract edge and plane features. The points of the edge feature of the current frame can be matched with the edges in the map. The same goes for plane features.

With edge and plane features, we can optimize the point to line distance, and point to plane

distance jointly, and estimate $\mathbf{R}$ and $\mathbf{T}$. However, sometimes we cannot extract edge features accurately enough, like in the long corridor or cave environment. In this case, we will lose the ability to estimate 6DOF movement.

To solve this issue, we extract and track features directly from intensity images. Fig. 4.1a shows the intensity images generated from an Ouster-64 LiDAR, where the image resolution is 1024×64. Even if the vertical resolution is low, we can still extract enough features (the red and green circles are the ORB features [1, 53, 54]) for estimating movement. We extract 3D points, $\mathcal{Y}_2 = \{\mathbf{Y}_1, \mathbf{Y}_2, \cdots, \mathbf{Y}_k\}$ directly from the point cloud according to the index of the matched ORB feature points from the intensity image. Each 3D feature point is assigned a score $\mathcal{S} \in \{\mathbf{S}_1, \mathbf{S}_2, \cdots, \mathbf{S}_n\}$ obtained during feature extraction. Similarly, we can extract the corresponding points $\mathcal{X}_2 = \{\mathbf{X}_1, \mathbf{X}_2, \cdots, \mathbf{X}_k\}$ in the following scan, and we scan match as a least squared estimation problem:

$$\bar{\mathbf{X}}_n = \mathbf{R}\mathbf{X}_n + \mathbf{T} \tag{4.2}$$

$$\arg\min_{\mathbf{R},\mathbf{T}} \sum_{n \in \mathcal{N}} \| \left(\mathbf{Y}_n - \bar{\mathbf{X}}_n\right) \cdot \mathbf{S}_n \|^2 \tag{4.3}$$

where $\mathcal{N} = [1, 2, \cdots, k]$, and $\mathbf{S}_n$ is the score of the nth matched feature point according to Hamming distance. This way, we can limit the number of features to around 200 points.

4.3.2 Map Optimization

The LiDAR intensity odometry generates a transformation matrix

$$\hat{\mathbf{T}} = \begin{bmatrix} \mathbf{R} & \mathbf{T} \\ \mathbf{0} & 1 \end{bmatrix}$$

between current sensor frame and the map frame. In this module, we jointly optimize the scan-to-map residual and LiDAR BA residual to correct the pose drift.

LiDAR Bundle Adjustment (BA)

Similar to visual SLAM BA, we can use the LiDAR BA (a non-linear optimization problem) to correct the drift. With this strategy, the last k frames are used in a residual function. We remove the oldest frame if the number of frames is larger than k, and add the newest frame to the sliding window as Fig. 4.3 showed. We can then match the current frame with the last k frames in the window and store the matched 3D feature points in $\mathcal{P}^c = \{\mathbf{P}_0^c, \mathbf{P}_1^c, \cdots, \mathbf{P}_k^c\}$, and $\mathcal{F}^l = \{\mathbf{F}_0^l, \mathbf{F}_1^l, \cdots, \mathbf{F}_k^l\}$ for current and last k frames, respectively.

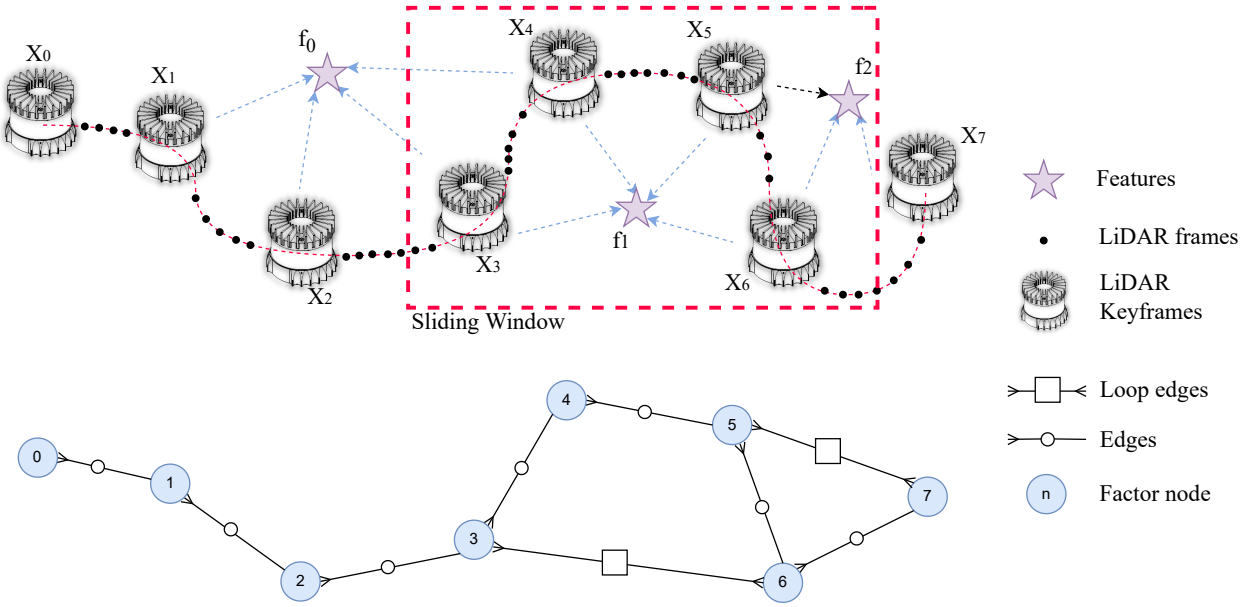


Figure 4.3 Illustration of the sliding window strategy used for LiDAR BA (top) and Pose graph (bottom) with loop closure contraints.

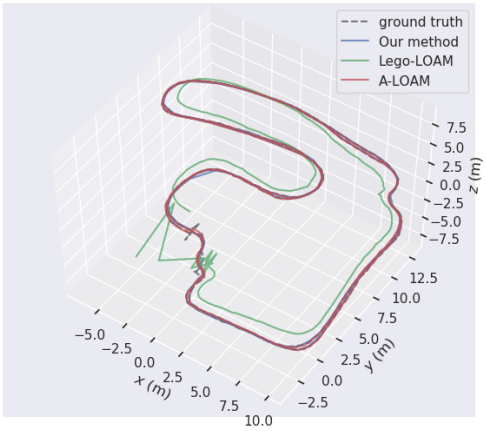
Where $\mathbf{P}_i^c = \{\mathbf{P}_{i0}^c, \mathbf{P}_{i1}^c, \cdots, \mathbf{P}_{im}^c\}$, $\mathbf{P}_{ij}^c = [p_x^{ij}, p_y^{ij}, p_z^{ij}, 1]^T$, The transformation matrix of the last k frames are calculated and stored in $\hat{\mathcal{T}}^l = \{\hat{\mathbf{T}}_0, \hat{\mathbf{T}}_1, \cdots, \hat{\mathbf{T}}_k\}$ by the previous map optimization step. The residual function is therefore:

$$\boldsymbol{f}_b = \sum_{i=0}^{k} \sum_{j=0}^{m} (\hat{\mathbf{T}} \mathbf{P}_{ij}^c - \hat{\mathbf{T}}_i \mathbf{F}_{ij}^l) \tag{4.4}$$

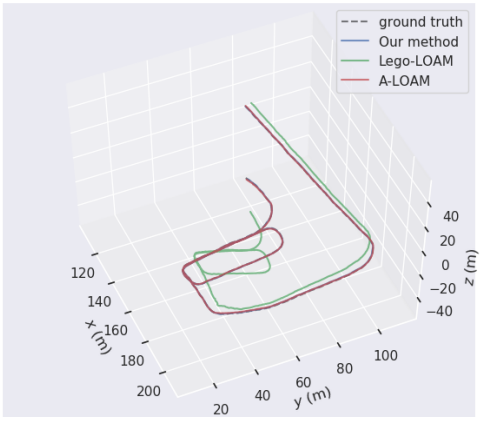
where $\boldsymbol{f}_b$ is the residual of LiDAR BA. In addition, we treat $\hat{\mathbf{T}}$ from the intensity odometry as an initial pose of current frame.

Scan-to-Map

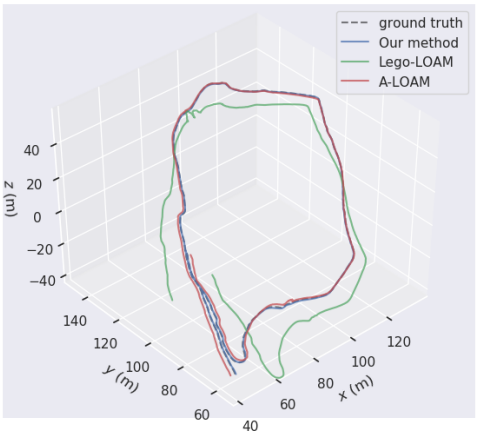
To match with previous plane feature points, we create an ikd-Tree [117] map, which is a incremental 3D k-d Tree. This map incrementally updates a k-d tree with incoming plane feature points. Compared to static k-d trees, this method has lower computation cost. For plane features, we divide the plane into ground plane and general plane features. When meeting a flat area, adding ground plane constraints is better than plane only method since the ground plane can constrain the roll and pitch direction more accurately. We extract normal plane features like in [11]. As for the ground plane, we first give an initial height of robot to extract possible ground planes and then use RANSAC to segment the ground



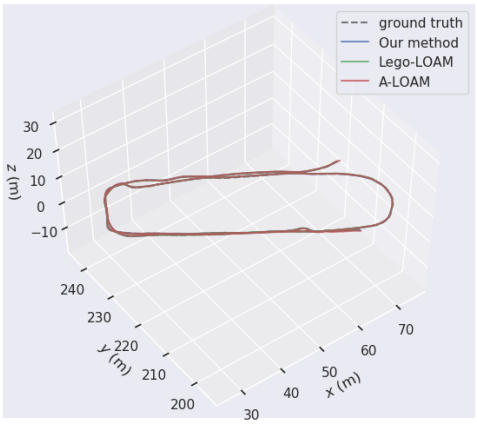
(a) Trajectories in outdoor environment with up and down stairs (191m).



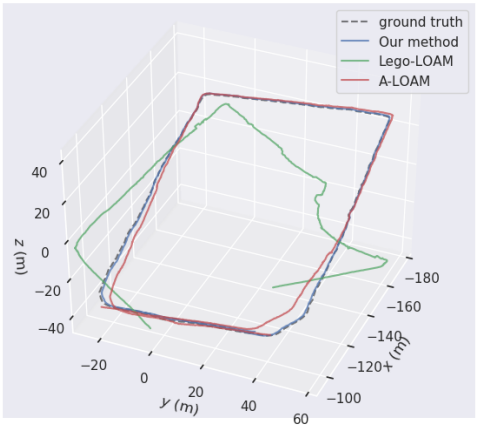
(b) Trajectories in outdoor environment with steep slope (414m).



(c) Trajectories on mountain with a long slope (507m).



(d) Trajectories in parking lot with flat ground (249m).



(e) Trajectories on the street with revisiting the start point (478m).

Figure 4.4 Trajectories results in multiple environments. The experimental results prove that our method is able to estimate the position accurately in various scenarios. LeGO-LOAM's algorithm works well in flat environments, but not in environments with slopes.

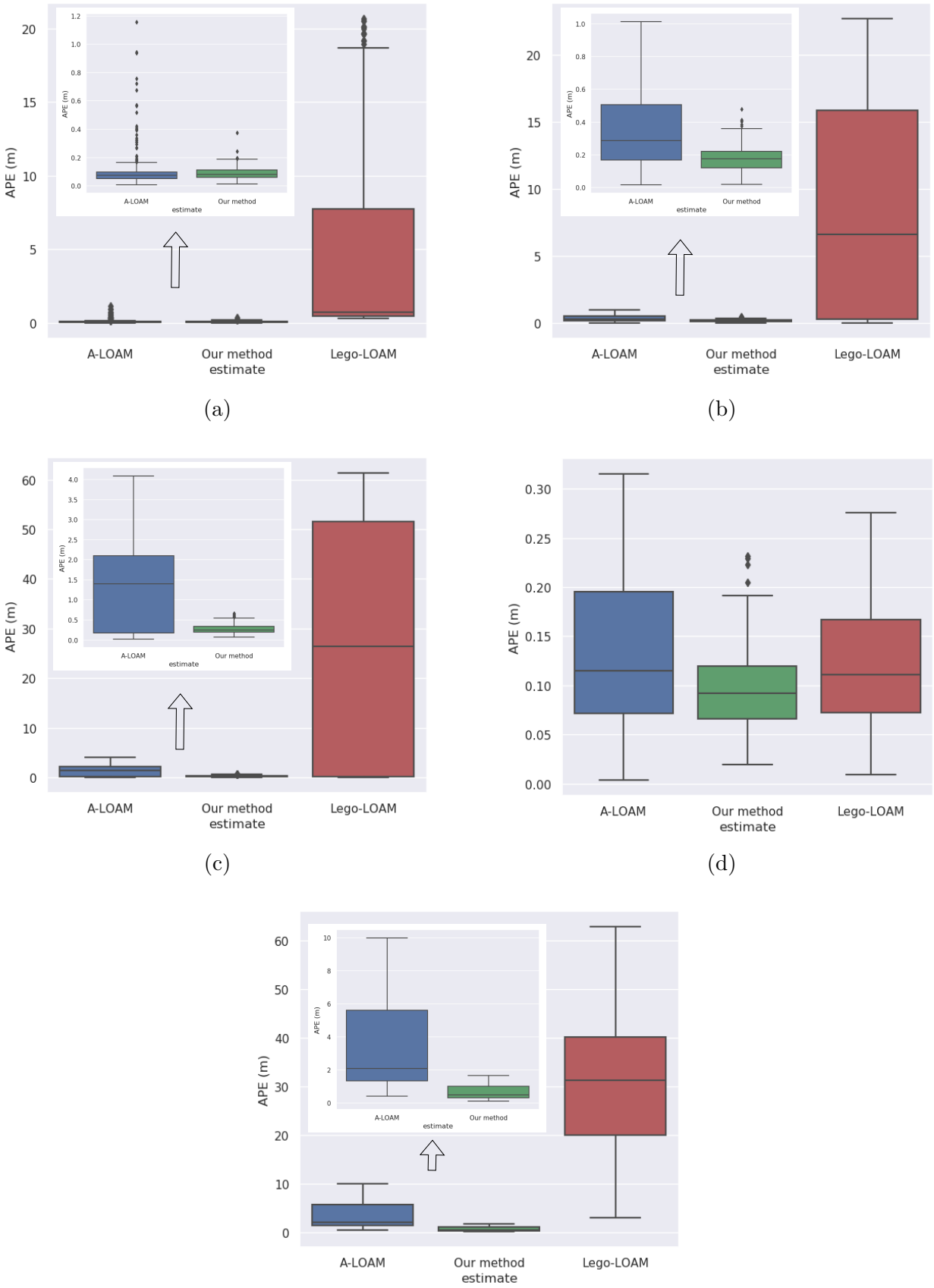


Figure 4.5 Absolute Pose Error in multiple environments. The APE of our method is the smallest in all these scenarios, followed by A-LOAM.

plane points. The plane features are extracted from the point cloud, and are stored in $\mathcal{P}^m = \{\mathbf{P}_0^m, \mathbf{P}_1^m, \cdots, \mathbf{P}_n^m\}$, where $\mathbf{P}_0^m = \{p_x, p_y, p_z\}^T$. For each plane feature point $\mathbf{P}_i^p$, we can find 3 nearby plane points from the ikd-Tree map, and store them in $\mathcal{F}^m = \{\mathbf{F}_0^m, \mathbf{F}_1^m, \cdots, \mathbf{F}_n^m\}$, where $\mathbf{F}_i^m = \{\mathbf{P}_i^{m0}, \mathbf{P}_i^{m1}, \mathbf{P}_i^{m2}\}$. The residual function can be formulated as:

$$\hat{\mathbf{P}}_i^m = \mathbf{R}\mathbf{P}_i^m + \mathbf{T} \tag{4.5}$$

$$\boldsymbol{f}_s = \sum_{i=0}^{n} (\hat{\mathbf{P}}_i^m - \mathbf{P}_i^{m0})^T \cdot \left(\frac{(\mathbf{P}_i^{m1} - \mathbf{P}_i^{m0}) \times (\mathbf{P}_i^{m2} - \mathbf{P}_i^{m0})}{|(\mathbf{P}_i^{m1} - \mathbf{P}_i^{m0}) \times (\mathbf{P}_i^{m2} - \mathbf{P}_i^{m0})|} \right) \tag{4.6}$$

where $\hat{\mathbf{P}}_i^m$ is the 3D mapped point which was converted by $\mathbf{P}_i^m$ from the sensor to the map coordinate system through (4.5). $\mathbf{P}_i^{m0}$, $\mathbf{P}_i^{m1}$, and $\mathbf{P}_i^{m2}$ are nearby points of $\hat{\mathbf{P}}_i^m$ on the map.

After getting the residual function of LiDAR BA and scan-to-map, we can formulate the whole optimization problem as:

$$\arg\min_{\mathbf{R},\mathbf{T}} (\boldsymbol{f}_b + \boldsymbol{f}_s) \tag{4.7}$$

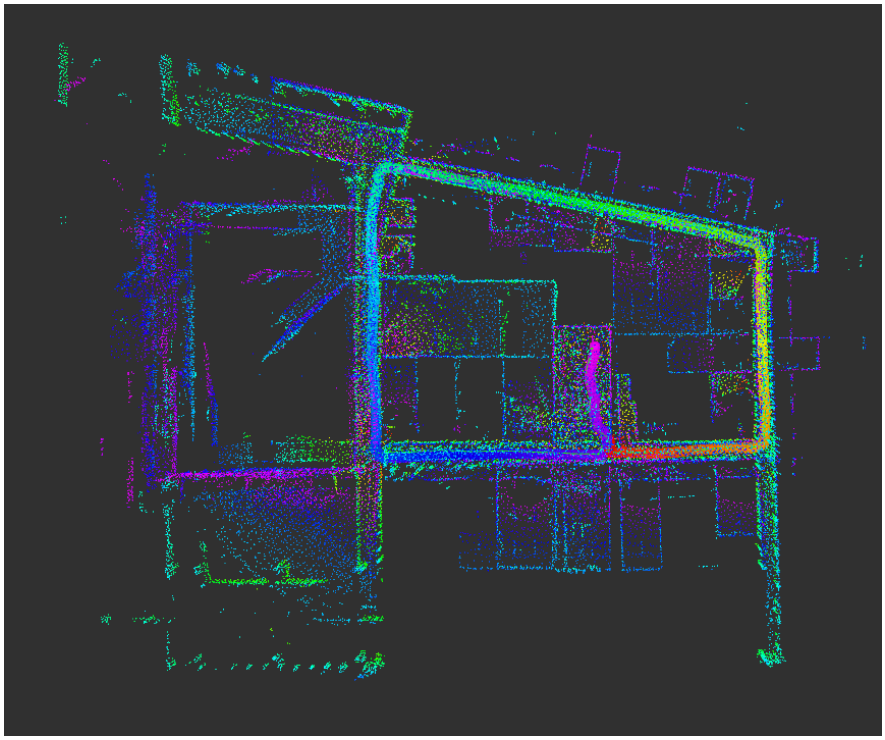
4.3.3 Pose Graph Optimization

During map optimization, we can get a better pose estimation of the current frame. Once done, we can use optimized results to correct the drift for the future frames and publish high-frequency optimized odometry in real time. In the backend, we build a pose graph on top of map optimization with LiDAR keyframes. First of all, we extract keyframes from whole LiDAR frames with three criteria:

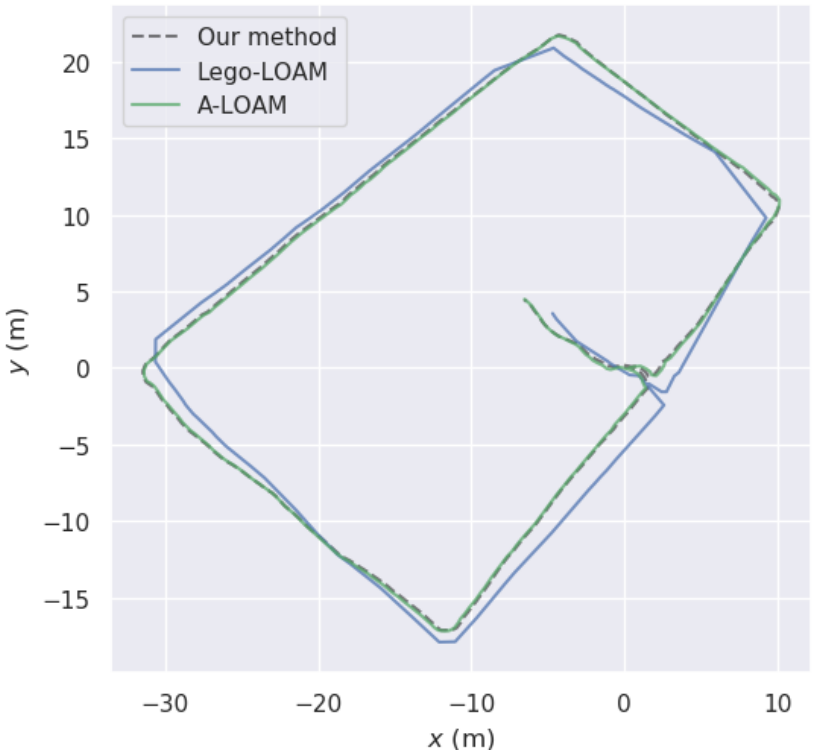
- The distance between current frame and last keyframe is larger than a threshold.
- The angle between two keyframes is larger than a threshold.
- The number of matched feature points is less than a threshold.

We can use the optimized poses of keyframes as the vertexes of the pose graph, and the relative poses between two keyframes as the edges of pose graph.

We also add loop constraints to the pose graph, as Fig. 4.3 shows, we use the latest keyframe as the anchor frame. With a trained vocabulary, we can compare descriptors of the current keyframe with a database where the history descriptors are stored. If we can not match it with history descriptors, then no loop closure is found for this keyframe [42]. If we successfully match a previous keyframe, we can then put it into the outlier rejection procedure to test whether it is an incorrect loop closure. If it is positive, we can add this loop constraint

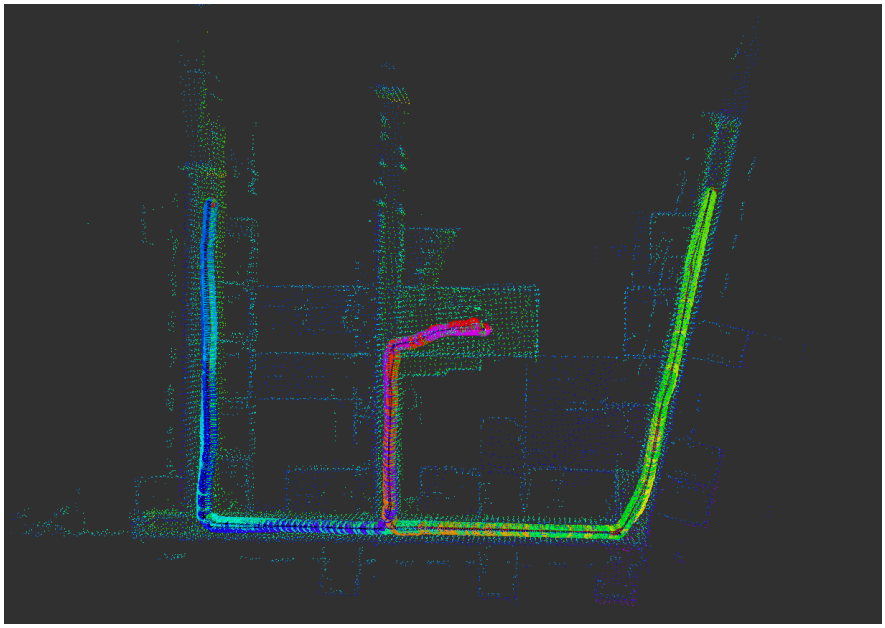


(a) Map generated by our method in the long corridor environment.

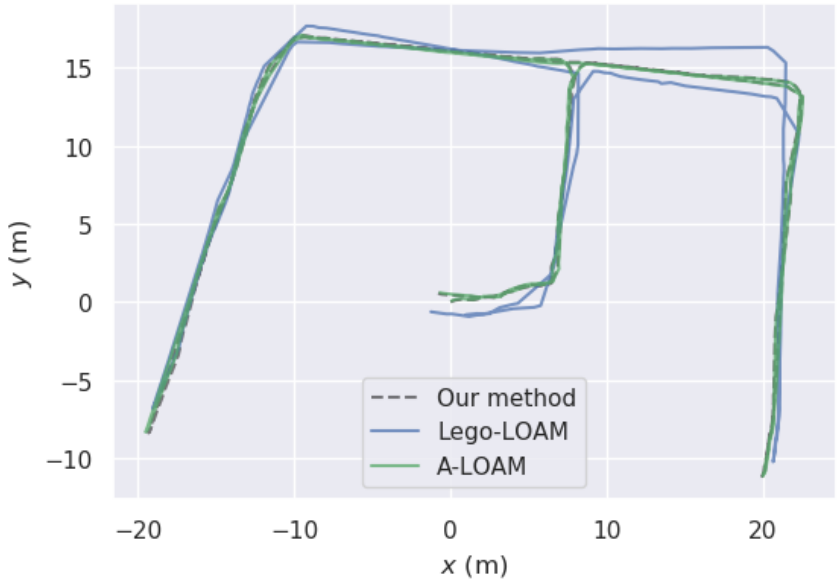


(b) Trajectories of A-LOAM, LeGO-LOAM and our method.

Figure 4.6 Map and trajectories of the Spot robot in a building with long corridors. In this scene, we walked along the corridor back to the starting point. In this experiment, the drift of LeGO-LOAM is relatively large, and the performance of A-LOAM is close to our method, but it is not clear from these two graphs which method is better.



(a) Map result of our method with a different path.



(b) Trajectories of different methods.

Figure 4.7 Instead of going back around in a circle to the starting point, we went back and forth through all the corridors for this test and then returned to the starting point. This situation is extreme because there is no good loop to correct the drift in the pitch direction. The experimental results show that LeGO-LOAM drifts more than in the previous scenario. A-LOAM and our method are almost the same.

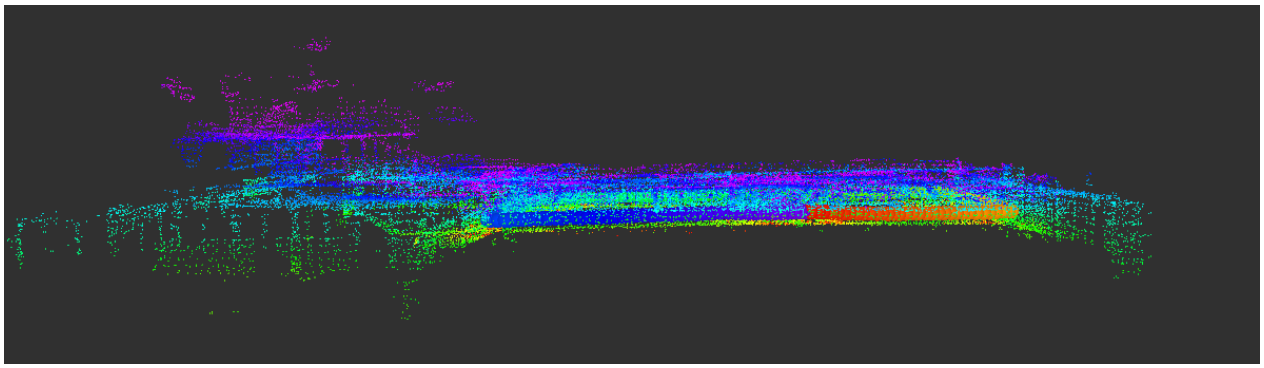
between the current and the loop candidate factor node in the pose graph. Finally, we use g2o [118] to solve the pose graph optimization problem.

4.4 Experimental Results

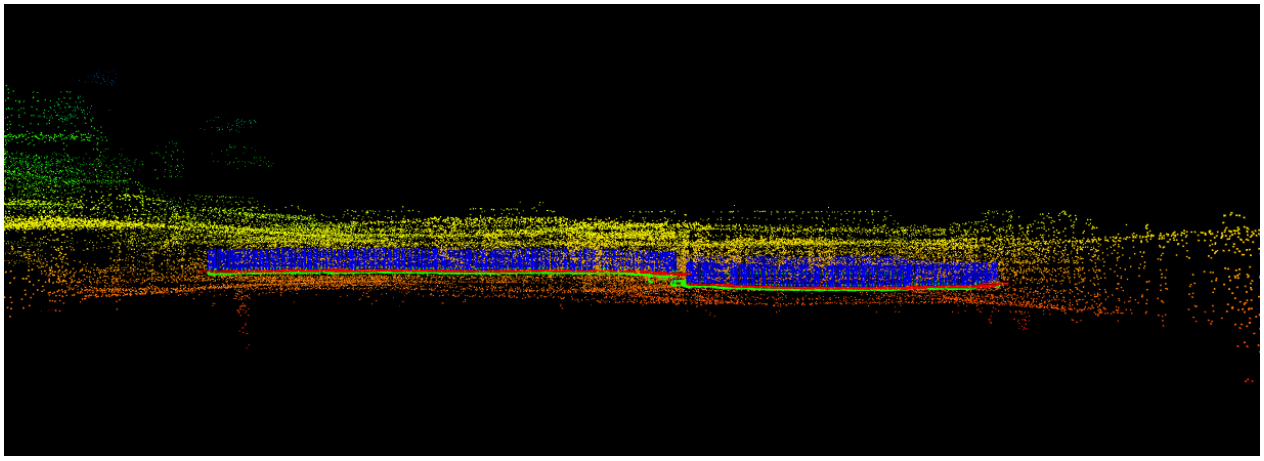
To prove our algorithm's reliability, we present our experimental results in an indoor environment with long corridors (Fig. 4.9b), a multi-storey indoor environment, a mountain, and a street environment. The reason why we chose these environment is that they are all different from each other, and they are challenging for pure LiDAR SLAM. In the indoor environment, we have long corridors and narrow passages. In the mountain environment, we have steep slopes and narrow passages. In the street environment, we have many obstacles and many turns. Those scenarios are difficult for pure LiDAR SLAM to handle. In the following, we will introduce the details of our experiments.

In our experiments, we compared our approach with two other popular pure LiDAR SLAM systems: LeGO-LOAM [12] and A-LOAM [109]. The A-LOAM and LeGO-LOAM algorithms were obtained from their open-source code available on Github. The attempt to compare the open-source Intensity-SLAM [45] was unsuccessful as we could not run it. Our intensity based SLAM system shows competitive performance in different environments, including indoor, outdoor, and some extreme scenarios, such as long corridors. Most other LiDAR SLAM systems fail in such extreme environment with fewer edge features. We first tested our method with a public dataset provided by Shan et al. [42] which was collected by Ouster OS1-128 LiDAR. LIO-SAM's trajectory [13] was treated as the ground truth since it is estimated based on LiDAR, 9-axis IMU and GPS, which is much more accurate than LiDAR only SLAM. In Fig. 4.4, we present the trajectories of our method, LeGO-LOAM, and A-LOAM in various terrains, while Fig. 4.5 shows the corresponding Absolute Position Error (APE) of Fig. 4.4. From Fig. 4.5, we can say our method has a significantly lower (T-test [119], $p<1e\text{-}5$) APE than others, except for the scenario in Fig. 4.5a where our result is not significant (T-test, $p=0.857$ compared with A-LOAM). The results in Fig. 4.4 and Fig. 4.5 prove our proposed approach achieves competitive results compared to both A-LOAM and LeGO-LOAM, especially in Fig. 4.4d where the trajectory is close to the ground truth trajectory in the end, while others drift a lot. The effectiveness of LeGO-LOAM is limited to level terrains as it relies on a ground plane constraint. It becomes challenging to extract the ground plane information in uneven terrains, thereby hindering its performance in such environments.

We also test our algorithm indoors with a Spot robot (from Boston Dynamics) equipped with Ouster Os0-64 LiDAR (Fig. 4.9a). The scene of this experiment mainly contains the



(a) The front view of the map of our method

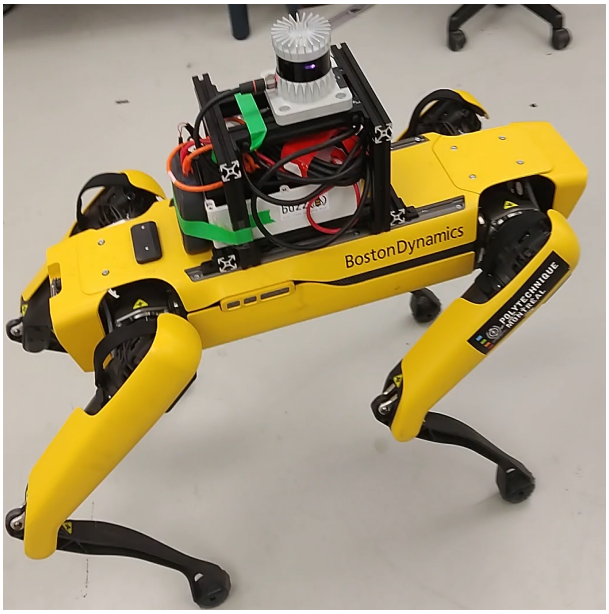


(b) The front view of the map of A-LOAM

Figure 4.8 Difference between our method and A-LOAM in long corridors. As we can see, the map generated by our method is smoothly connected to the starting position, but the map generated by A-LOAM has a clear break at the end.

Table 4.1 Time consumption of different algorithms (ms)

	A-LOAM	LeGO-LOAM	Ours
Features extraction	6.33 ± 1.99	10.78 ± 4.36	10.70 ± 1.50
Scan registration	20.04 ± 4.06	2.53 ± 3.88	3.15 ± 2.07
Map optimization	10.63 ± 2.61	21.61 ± 7.55	29.02 ± 6.23
PGO	N/A	23.48 ± 7.81	1.73 ± 10.51



(a) Spot robot for indoor experiment

(b) Indoor corridor

Figure 4.9 Indoor environment and Spot robot used for testing our algorithm. Spot with Ouster LIDAR mounted on it walking and collecting data in the corridor shown on the right.

same long corridor as Fig. 4.9b. In this scenario, we ran different algorithms for testing the ability of localization and map building in real world. Both Fig. 4.6a and Fig. 4.7a are the maps generated by our algorithm. Fig. 4.6b and Fig. 4.7b then show the trajectories of differnt algorithms in the corresponding environments. From the trajectories, we can see that LeGO-LOAM drifts a lot. A-LOAM's and our trajectories are almost the same. Due to the indoor environments, we can hardly collect the ground truth trajectories with RTK. So we try to analyze the map details (Fig. 4.8) to evaluate the algorithm strengths and weaknesses. Fig. 4.8a shows the front view of Fig. 4.6a, and Fig. 4.8b shows the front view of the same scenario generated by A-LOAM. We can see that our method can smoothly connect the start point at the end of the trip, but A-LOAM's map is disconnected. At this point, we can say that our method is more reliable in such an extreme environment. In addition, we analyzed the time consumption of different SLAM algorithms on the Intel processor with the same data collected by Os0-64 LiDAR. Table 4.1 shows that our intensity-based front-end is able to calculate the odometry within 15 ms and our method is efficient enough to meet the real-time requirements of 10 Hz LiDAR.

4.5 Conclusions

In this paper, we propose a novel intensity-based pure LiDAR SLAM method. We first proposed a novel lightweight intensity-based odometry method, which directly match 3D features point extracted from the intensity images. Then we propose a novel map optimization method, which jointly optimizes the LiDAR BA and point-to-plane residuals. Finally, we propose a novel intensity-based pose graph optimization method, which can optimize the pose graph based on the intensity image. We tested our method in both outdoor and indoor environments. The results proved that our method can achieve competitive results compared with other popular pure LiDAR SLAM methods. In the future, we will further improve our method by using more advanced feature extraction methods and loop closure detection methods. We will also test our method in more challenging environments.

CHAPTER 5 ARTICLE 2: LIDAR-BASED REAL-TIME OBJECT DETECTION AND TRACKING IN DYNAMIC ENVIRONMENTS

Preface: The ability to stably estimate a robot's pose in dynamic environments is crucial for autonomous robots. During the extraction of edge, plane, or texture features from the environment, many features may belong to moving objects, causing feature matching to diverge. Typically, the estimation of robot movement assumes a static environment. However, real-world scenarios often involve dynamic environments. To address this issue, we need to detect dynamic objects. A straightforward approach is to use LiDAR data directly to cluster and track all objects. However, the large number of points in LiDAR data results in long processing times, hindering real-time performance.
In this chapter, we propose a method to extract foreground objects from the environment and track them in real-time. Since the points associated with foreground objects are fewer than the total points in a LiDAR frame, this approach enables real-time processing. Our proposed framework consists of three main components: data preprocessing, object tracking, and dynamic object detection. We evaluated our system using the Spot robot. The experimental results demonstrate that our system achieves real-time performance and outperforms state-of-the-art methods in terms of accuracy and reliability, even with unstable odometry. We believe that utilizing this proposed approach can play a crucial role in the development of autonomous robots in dynamic environments.

Full Citation: Du, Wenqiang, and Giovanni Beltrame. "LiDAR-based Real-Time Object Detection and Tracking in Dynamic Environments." IEEE Transactions on Robotics (T-RO), 2024. Submitted on June 25, 2024.

Abstract: In dynamic environments, the ability to detect and track moving objects in real-time is crucial for autonomous robots to navigate safely and effectively. Traditional methods for dynamic object detection rely on high accuracy odometry and maps to detect and track moving objects. However, these methods are not suitable for long-term operation in dynamic environments where the surrounding environment is constantly changing. In order to solve this problem, we propose a novel system for detecting and tracking dynamic objects in real-time using only LiDAR data. By emphasizing the extraction of low-frequency components from LiDAR data as feature points for foreground objects, our method significantly reduces

the time required for object clustering and movement analysis. Additionally, we have developed a tracking approach that employs intensity-based ego-motion estimation along with a sliding window technique to assess object movements. This enables the precise identification of moving objects and enhances the system's resilience to odometry drift. Our experiments show that this system can detect and track dynamic objects in real-time with an average detection accuracy of 88.7% and a recall rate of 89.1%. Furthermore, our system demonstrates resilience against the prolonged drift typically associated with front-end only LiDAR odometry.

All of the source code, labeled dataset, and the annotation tool are available at: `https://github.com/MISTLab/lidar_dynamic_objects_detection.git`

5.1 Introduction

Typically, when estimating the movement of robots, the assumption is that the surroundings are static [1, 11, 57]. However, this assumption is now being challenged more frequently in recent research. This is due to significant progress in the area of ego-motion estimation [2, 8, 13] and an increasing need for autonomous robots to work in dynamic environments [120, 121], such as urban areas, warehouses, and construction sites. In these settings, the presence of moving objects like pedestrians, cyclists, and other vehicles can greatly affect the robot's ability to determine its location and map the area. Consequently, the robot's capability to identify and track moving objects in real time becomes a crucial task.

Detecting and tracking moving objects in real time is a complex task. It demands that the robot can distinguish between static and moving objects, and track the motion of these moving objects in real time. This task is particularly challenging for robots that depend on Light Detection and Ranging (LiDAR) data for perception, as the data is typically unordered and noisy. In contrast to camera-based systems, which can obtain real-time semantic information of objects for tracking [122–126], processing LiDAR data is not as straightforward and efficient due to the higher dimensionality, increased number of points, and the sparse nature of point clouds compared to image data.

At present, LiDAR-based techniques for identifying moving objects are categorized into two primary groups: map-based methods, and segmentation-based approaches [87, 95, 127], each coming with inherent limitations. Map-based methods demand extensive computational resources for map creation, making them unsuitable for real-time applications. Besides, those techniques struggle with complex environments and require increased computational effort as the size of the map grows. On the other hand, Segmentation-based approaches face challenges

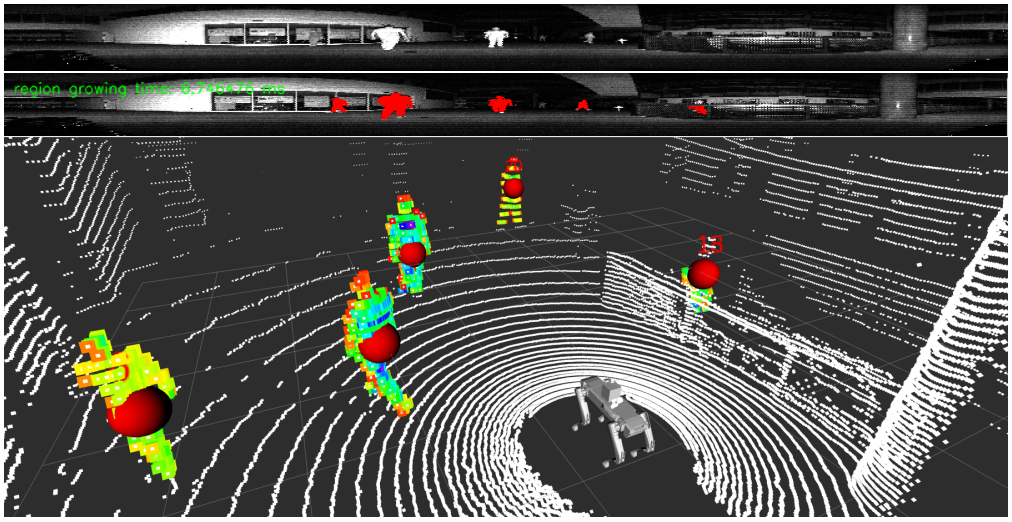


Figure 5.1 Illustration of dynamic object detection. Top: The raw intensity image. Middle: Dynamic objects are highlighted in red within the intensity image. Bottom: Dynamic points corresponding to these objects are extracted from the raw 3D point cloud, showing their spatial distribution.

in accurately detecting objects within complex environments filled with unidentified objects, and their performance heavily depends on the objects' visual characteristics.

To address these challenges, we propose a mapless dynamic object detection and tracking system. Our method first transforms the unordered point cloud data into an intensity image. This image then undergoes a process of 2D gaussian convolution to reduce its high-frequency components. This step helps in reducing noise and simplifying the distinction between foreground objects and the background. The 3D points that belong to the features of these foreground objects are retrieved from the original LiDAR data. By using intensity-based ego-motion estimation [128], our system can track various object clusters in real-time. Analyzing the movement of these clusters, the system can identify the central points of moving objects within the environment and, through region growing, reconstruct the entire set of points that make up these moving objects [86, 93].

The main contributions of this work are summarized as follows:

1. **Innovative Processing of LiDAR Data:** We propose a novel method for processing LiDAR data by converting unordered point clouds into intensity images and applying 2D convolution. This technique effectively minimizes noise and accentuates the features of foreground objects, facilitating their identification and tracking. This represents a considerable advancement over conventional methods that struggle to process LiDAR data effectively;

2. **Efficient Object Clustering and Identification:** Employing intensity-based ego-motion estimation alongside a sliding window technique, our system excels in tracking clusters of objects in real time. Analyzing the movement of these clusters enables the identification of the central points of moving objects, leading to the accurate reconstruction of their full point sets. This method significantly enhances the system's efficiency in recognizing and tracking objects within its surroundings;

3. **Real-Time Detection and Tracking of Dynamic Objects:** Our method encompasses a robust system for detecting and tracking dynamic objects in real time. Using LiDAR data, we distinguish between static and dynamic objects and track their movements with high precision. This capability is essential for navigating environments populated with frequently moving objects;

4. **Robust Reconstruction of Dynamic Objects:** Using intensity image based region growing techniques, our method reconstructs the complete set of points that make up moving objects. This enables the identification of the objects' structures and positions within the environment, which is crucial for navigation and task execution. This robust reconstruction capability marks a significant improvement over prior techniques, which offer only partial or less precise representations.

This article is structured as follows: Section 5.2 offers a review of related work, focusing on the progress and challenges in the dynamic object detection. This sets the stage for our research by providing necessary background and justification. Section 5.3 elaborates on our proposed methodology. It details the process of converting LiDAR point clouds into intensity images, applying 2D convolution for background reduction, and tracking dynamic objects using intensity-based ego-motion estimation, clustering, estimation of movement combined with a sliding window approach. Section 5.4 is dedicated to the experimental evaluation of our approach. It includes a comparative analysis with existing methods, demonstrating the effectiveness of our system in handling dynamic environments.

Section 5.5 summarizes the main contributions of our work and highlights its significance in enhancing robotic capabilities for navigating and operating in dynamic, complex environments. It also addresses the limitations of our study and proposes future research directions, suggesting advancements in ego-motion estimation and dynamic object tracking. We discuss the broader implications of our findings, the potential applications of our method, and how our work contributes to the field of robotics, particularly in terms of autonomous navigation.

5.2 Related Work

5.2.1 Map-based methods

Map-based methods for the detection of dynamic objects can be categorized into three primary types: ray-tracing methods, visibility-based methods, and occupancy-based methods.

Ray-Tracing Methods Ray-tracing methodologies simulate LiDAR sensor behavior by tracing rays from the LiDAR sensor outward into the environment [81, 82]. Rays are traced through either the grid map [83] or voxel map [84] to identify dynamic objects. Should a ray traverse a grid cell or voxel, this indicates that the cell is empty. If an object obstructs the ray within this empty cell, it signifies that the object is dynamic.

Azim and Aycard [81] used an octree-based occupancy grid map to represent the dynamic surroundings of the vehicle and identified moving objects by noting discrepancies across scans. If the ray were reflected, then that volume would be occupied. If the ray were to traverse the volume, then it would be free. Peopleremover [82] efficiently removes dynamic objects from 3D point cloud data by constructing a regular voxel occupancy grid and traversing it along the sensor's line of sight to the measured points to identify differences in voxel cells between different frames, which correspond to dynamic points. It is applicable to both mobile mapping and terrestrial scan data, aiming to produce a clean point cloud devoid of dynamic objects, such as pedestrians and vehicles. This process conservatively removes volumes only when they are confidently identified as dynamic, thus preserving the integrity of static elements in the point cloud data. Khronos [85] introduces a method for detecting dynamic objects in robotic vision systems using ray tracing. Instead of relying on volumetric data, it captures changes through surface representations and background and robot poses. By creating a library of rays between observed background vertices and robot positions, stored in a global hash map for efficiency, the method dynamically detects objects without real-time free-space mapping. Object detection is achieved by computing distances to rays, determining occlusion, consistency, or absence. This technique offers a practical approach to real-time object detection, balancing computational efficiency with accuracy. However, it perform poorly in dynamic environments with large open spaces with sparse surfaces. M-detector [86] is a motion event detection system inspired by the human visual system's Magnocellular cells, known for their motion sensitivity due to larger sizes and faster processing capabilities. Using ray tracing and the principle of occlusion, it detects moving objects by observing how they interrupt the LiDAR sensor's laser rays, effectively occluding previously visible backgrounds or recursively occluding themselves. The system processes inputs either as individual points

or frames, applying three parallel occlusion tests to determine motion events. This approach allows for low-latency, high-accuracy detection without requiring extensive training datasets. M-detector's design ensures it is highly generalizable, capable of detecting various object sizes, colors, and shapes across different environments with minimal computational resources, making it suitable for a wide range of robotic applications.

However, Ray-tracing methods demonstrate a significant sensitivity to the accuracy of odometry and incur substantial computational expenses during the construction of maps and and as the map size gradually increases [87–90]. Additionally, these methods lead to increased computational times [85] with prolonged operation and prove unsuitable for real-time applications.

Visibility-Based Methods

Visibility-based algorithms operate on a fundamental physical premise: light travels in straight lines, and if two points, one nearer and the other farther, are detected on the same ray, the nearer point has to be dynamic [89, 91, 92] to allow visibility of the farther point. The core concept involves recognizing dynamic points by examining the geometric variances between a query range image and its equivalent map range image. In essence, if a pixel's range value in the query image surpasses the one in the map image, it indicates an occlusion at the map image pixel's location according to these methods. Therefore, points mapped onto these pixels are deemed dynamic [88].

Removert [93] introduces a novel method for the removal of dynamic objects from LiDAR point cloud data. This method exploits the visibility of points within the LiDAR data to identify dynamic objects. By comparing the visibility of points in the current frame to that in the map, the method can detect dynamic objects and subsequently remove them from the point cloud data. This approach proves effective in removing dynamic objects from LiDAR data and enhancing the accuracy of static object detection. However, it is an offline method and is not suitable for real-time applications. ReFusion [94] creates a dense mapping that can effectively handle dynamic environmental elements without relying on specific dynamic object class detection, using Truncated Signed Distance Function (TSDF) and voxel hashing on GPU. The method detects dynamic objects by leveraging residuals from the registration process and the explicit representation of free space in the environment, achieving geometric filtering of dynamic elements. However, maintaining a dense TSDF mapping can be computationally expensive and memory-intensive.

The accuracy of the pose estimation is critical to the performance of the visibility-based methods since they need to retrieve the corresponding points in the map according to the

odometry information. Moreover, false negatives, which include but are not limited to self-interference within point clouds and false positives caused by parallel point discrepancies, misjudged occluded points, and contact point errors, directly undermine the reliability of dynamic points extraction [87]. Additionally, the invisibility of static points, although less common than false negatives, poses a more challenging problem to solve. For instance, when dynamic obstacles continuously block part or all of the LiDAR's rays of light, the LiDAR fails to detect the static objects behind these dynamic obstacles, leading to a scenario where the dynamic obstacle's point cloud is never identified and filtered out [88].

Occupancy-Based Methods

Occupancy-based methods employ statistical models to estimate the likelihood of space being occupied, facilitating robust mapping of environments with dynamic or uncertain elements.

Dynablox [95] is a real-time, map-based dynamic object detection system. It incrementally estimates high-confidence free-space areas and leverages enhanced the Truncated Signed Distance Field (TSDF) [96] method for volumetric mapping. This approach enables robust detection of dynamic objects without assumptions about their appearance or the structure of the environment. The method demonstrated an Intersection over Union (IoU) of 86% and processed data at 17 frames per second (FPS) on a laptop-grade CPU in real-world datasets, surpassing appearance-based classifiers and approximating the performance of certain offline methods. However, the study also acknowledges certain limitations, including challenges in detecting extremely thin and sparsely measured objects, such as shades, due to the voxel-based map representation. It also relies on prior odometry to place points into map spaces. ERASOR2 [88] is an instance level dynamic object removal system that can effectively remove dynamic points from 3D point cloud maps. It uses Pseudo Occupancy Grid Map to represent whether the regions are temporally occupied or not by calculating the probability of those regions. It also employs instance segmentation [97] estimates to identify and exclude dynamic points at the instance level, ensuring that static points are preserved with minimal loss. This approach allows for the precise removal of dynamic objects from the map, enhancing the map's utility for navigation and planning tasks.

Nevertheless, these methods still require maintaining a map of the environment, which is memory-intensive and computationally demanding, especially for long runs.

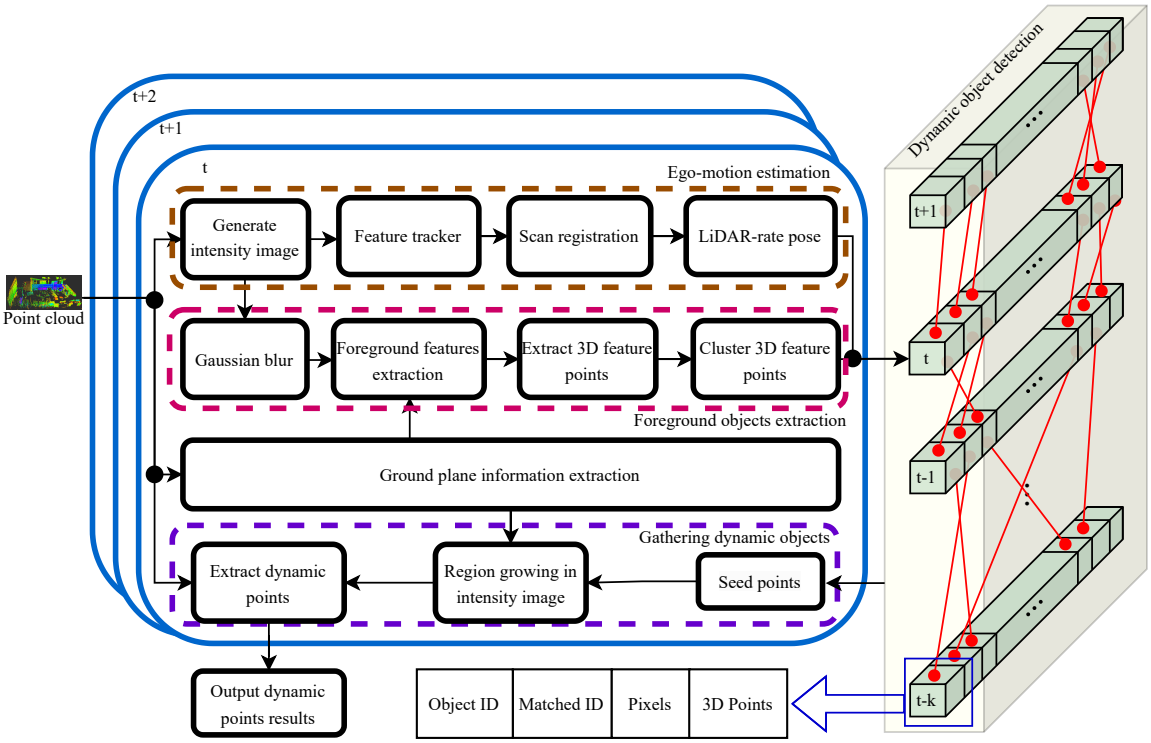


Figure 5.2 The framework of our dynamic object detection system, consisting of three main components: data preprocessing, object tracking, and dynamic object detection. The data preprocessing component converts unordered LiDAR point cloud data into intensity images and extracts low-frequency components to represent foreground objects. Object tracking involves ego-motion estimation, clustering, and cluster association. The ego-motion estimation component calculates the robot's movement between frames using intensity-based front-end odometry, providing real-time movement data despite potential noise and drift. The dynamic object detection component leverages data association results to track objects and identify seed points of dynamic objects, followed by a region growing algorithm to reconstruct the complete point set of dynamic objects from these seed points. This system processes LiDAR data in real-time to detect dynamic objects in complex environments, using the low-frequency components to represent foreground objects and enable real-time tracking across multiple frames.

5.2.2 Segmentation-Based Methods

Segmentation-based methods focus on identifying dynamic objects by semantically segmenting the point cloud and analyzing the movement of points or objects over time. These methods are particularly effective in detecting dynamic objects in complex environments with unidentified objects.

Thomas et al. [98] proposed a self-supervised learning method for dynamic object detection in LiDAR point clouds. They first projected each frame of point cloud into 2D grids. Then, they stacked the 2D frames into 3D according to the time sequence. Finally, they used a KPConv network [99] 3D back-end and a three levels U-Net [100] 2D front-end to predict the future time steps of spatiotemporal occupancy grid maps. Chen et al. [101] used the range image to represent the point cloud data and combined the current frame and residual images generated between current frame and the last few frames. After then, they used a range projection-based segmentation CNNs [102–104] to predict the moving objects.

As an upgrade to [98], Sun et al. [105] extended the SalsaNext [104] by adopting a dual-branch (a range image branch and a residual image branch) and dual-head (an image head and a point head) architecture to separately deal with spatial and temporal information. After then, they used a 3D sparse convolution to refine the prediction results. In this coarse-to-fine framework, the network can predict the moving objects more accurately and efficiently. Mersch et al. [106] combined a sequence of LiDAR point cloud into a voxelized sparse 4D point cloud. Then, they feeded the combined 4D occupancy grid to a modified MinkUNet14 [107] to extract spatio-temporal features and predicted the confidence score for each point.

These methods are effective in detecting dynamic objects; however, they predominantly depend on a substantial volume of annotations for the dynamic objects in the training data, which may not always be accessible. Furthermore, segmentation-based methods incur significant computational costs and frequently yield unsatisfactory results in unfamiliar environments. Consequently, a need persists for a lightweight and label-free method for dynamic object detection in real-time.

In this work, to increase speed and reduce computational load, we primarily concentrate on the low-frequency components of LiDAR data, which are used to represent foreground objects. Given that the size of the low-frequency components is significantly smaller than that of the original LiDAR data, our approach enables real-time tracking of multiple objects across multiple frames.

5.3 Methodology

Our proposed method is illustrated in Fig 5.2. This framework principally comprises three components: data preprocessing, object tracking, and dynamic object detection. Our method is engineered to process

5.3.1 Data Preprocessing

Point Cloud to Intensity Image

Unlike cameras, which generate an image in each cycle, LiDARs produce a frame of point cloud data per cycle. Each point $\mathbf{p}$ within this frame records spatial dimensions and intensity information, $\mathbf{p} = \{x, y, z, i\}$. The intensity of a point is determined by the signal, which constitutes the number of photons collected by the sensor for a pixel during a single cycle. The initial step of our method involves converting the unordered LiDAR point cloud data into an intensity image through spherical projection. This process entails projecting the point cloud data onto a two-dimensional grid; each grid cell therein represents the intensity of the points contained within it. The grid cell's index (u, v) can be determined by the following equation:

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} \frac{(\arctan(y,x)+\pi)}{2\pi} \cdot w \\ \frac{\left(\beta_{up} - \arcsin(zr^{-1})\right)}{\beta_{fov}} \cdot h \end{bmatrix} \tag{5.1}$$

Where the r represents the distance between the point and the sensor, denoted as $||\mathbf{p}||_2$, w and h are the width and height of the image, respectively, and β_{up} and β_{fov} are the upper bound and field of view of the LiDAR sensor, respectively.

By converting the point cloud data into an intensity image, we treat the unordered LiDAR data similarly to an image, simplifying subsequent processing steps. This transformation permits the application of image processing techniques to LiDAR data, thus facilitating the identification and tracking of dynamic objects.

Low-frequency components extraction

The intensity image generated in the previous step contains high-frequency components that can significantly slow down the clustering and tracking processes. In order to extract features of the foreground objects with fewer points, we apply a 2D gaussian convolution to the intensity image to extract low-frenquency components, see (5.3), and treat them as a representative of foreground objects. This process involves convolving the intensity image

with a 2D Gaussian kernel which effectively smooths the image and minimizes noise:

$$g(m,n) = \frac{1}{2\pi\sigma_m\sigma_n} \exp\left(-\left(\frac{m^2}{2\sigma_m^2} + \frac{n^2}{2\sigma_n^2}\right)\right) \tag{5.2}$$

Here, m and n represent the distances from the center of the kernel to the current point and can take integer values, including negative ones. m ranges from $[-a, a]$ and n ranges from $[-b, b]$. In this case, as the intensity image is quite flat, we also configure the kernel to have a flat shape, where $a > b$. This configuration ensures that the Gaussian kernel matrix's index for the center of columns is 0, with a positions on either side, and similarly, the center index for rows is 0, with b positions on each side. σ_m and σ_n denote the standard deviations along the horizontal and vertical directions, respectively. The Gaussian kernel size is $(2a+1, 2b+1)$. Then, we convolve the intensity image with the Gaussian kernel to extract the low-frequency components:

$$\begin{aligned} h(u,v) &= (f * g)(u,v) \\ &= \sum_{m=-a}^{a} \sum_{n=-b}^{b} f(u-m, v-n) \cdot g(m,n) \end{aligned} \tag{5.3}$$

where $f(u-m, v-n)$ represents the intensity value of the image at position $(u-m, v-n)$, and $g(m,n)$ represents the value of the kernel at position (m,n). The variables m and n are the distances from the center of the kernel, ranging over the kernel dimensions. The convolution operation is performed by sliding the kernel over the image and calculating the sum of the products of the kernel values and the image values at each position.

Finally, we subtract the blurred image from the original intensity image, as shown in (5.4). The resulting image retains only the low-frequency components, which depict the features of the foreground objects. By focusing on these low-frequency components, we are able to accurately identify and track dynamic objects in real-time.

$$t(u,v) = B\left(f(u,v) - h(u,v), \theta\right) \tag{5.4}$$

where $B(x,\theta)$ is a binarization function defined by:

$$B(x,\theta) = \begin{cases} 1 & \text{if } x > \theta \\ 0 & \text{otherwise} \end{cases}$$

and θ is the threshold value.

5.3.2 Object Tracking

After extracting the low-frequency components from the intensity image, the 3D points associated with the features of these foreground objects can be further extracted from the raw LiDAR data. Subsequently, these points can be clustered into distinct objects and tracked in real-time, owing to their small number.

Intensity-based Ego-Motion Estimation

The initial step in object tracking involves estimating the ego-motion of the robot. This is accomplished by extracting ORB features [55] from the intensity image and calculating the transformation matrix between consecutive frames. The resulting transformation matrix is used to estimate the robot's movement between frames, an essential factor for accurate object tracking. This technique, known as intensity-based ego-motion estimation, was proposed in our previous work [128]. Here we only use the front-end of the SLAM system to make is lightweight and real-time, but drift may occur during long-term operation. However, in the following we show that drift will not affect the object tracking accuracy.

For each frame of intensity image, we extract ORB features and match them with the features in the previous frame. At the same time, we can extract the corresponding 3D points from current and previous frames and put them into $\mathcal{X} = \{\mathbf{X}_1, \mathbf{X}_2, \cdots, \mathbf{X}_k\}$ and $\mathcal{Y} = \{\mathbf{Y}_1, \mathbf{Y}_2, \cdots, \mathbf{Y}_k\}$, respectively. Then, the scan match problem can be formulated as a least square estimation problem:

$$\arg\min_{\mathbf{R},\mathbf{t}} \sum_{n \in \mathcal{N}} \parallel \mathbf{Y}_n - (\mathbf{R}_i \mathbf{X}_n + \mathbf{t}_i) \parallel_2 \tag{5.5}$$

where $\mathcal{N} = [1, 2, \cdots, k]$, $\mathbf{X}_n = \{x_n, y_n, z_n\}$, and $\mathbf{R}_i$ and $\mathbf{t}_i$ are the rotation matrix and translation vector at current frame time i with respect to the previous frames, respectively. The solution of this optimization problem is the ego-motion of the robot between two frames. So, we need to accumulate $\mathbf{R}_i$ and $\mathbf{t}_i$ to get the robot's pose in a reference coordinate system.

Clustering

Since the number of objects varies across different frames, traditional clustering methods that require predefined cluster numbers, such as K-means, cannot be employed. Furthermore, the vertical disparity between two adjacent laser beams increases as the distance to the objects increases. Consequently, addressing this challenge demands the use of a more robust clustering method. We initially employ an Euclidean distance-based clustering method to

segment all points into groups, which are subsequently clustered into distinct objects.

As mentioned above, we can get 3D points $\mathcal{Q} = \{\mathbf{p}_0, \mathbf{p}_1, \cdots, \mathbf{p}_n\}$ that represent the features of the foreground objects. For each point in $\mathcal{Q}$, we calculate the Euclidean distance between it and all other points in horizontal and vertical plane, see (5.6).

$$\mathbf{d}(\mathbf{p}_i, \mathbf{p}_j) = \begin{bmatrix} d_{xy}(\mathbf{p}_i, \mathbf{p}_j) \\ d_z(\mathbf{p}_i, \mathbf{p}_j) \end{bmatrix} = \begin{bmatrix} \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} \\ |z_i - z_j| \end{bmatrix} \tag{5.6}$$

The points i and j can be clustered into the same group $\mathcal{C}_m = \{\mathbf{p}_i, \mathbf{p}_j, \cdots, \mathbf{p}_k\}$ if the distance between them is less than a predefined threshold, ϵ. The clustering process can be formulated as (5.7):

$$\forall \mathbf{p}_i, \mathbf{p}_j \in \mathcal{C}_m, \quad \mathbf{d}(\mathbf{p}_i, \mathbf{p}_j) \leq \epsilon \tag{5.7}$$

After clustering, we get a group of clusters $C = \{\mathcal{C}_1, \mathcal{C}_2, \cdots, \mathcal{C}_m, \cdots, \mathcal{C}_q\}$, and the total number of cluster q can be different for each frame.

Cluster Association

The clusters need to be linked between frames to ensure the continuity of the tracking process. We use the Global Nearest Neighbor (GNN) [129] method for this association. We first calculate the center point of each cluster in the current frame, $\mathbf{c}_m^i = [x_m, y_m, z_m]^T$, and the center point of each cluster in the previous frame, $\mathbf{c}_n^{i-1} = [x_n, y_n, z_n]^T$. Then, we calculate the Euclidean distance between the center points of the clusters in the current frame and the previous frame:

$$d_{\mathbf{c}}(\mathbf{c}_m^i, \mathbf{c}_n^{i-1}) = \| \mathbf{c}_m^i - (\mathbf{R}_i \mathbf{c}_n^{i-1} + \mathbf{t}_i) \|_2 \tag{5.8}$$

Subsequently, we will build a cost matrix $\mathbf{D} \in \mathbb{R}^{M \times N}$, where M and N are the number of clusters in the current frame and the previous frame, respectively. Each element $d_{m,n} = d_{\mathbf{c}}(\mathbf{c}_m^i, \mathbf{c}_n^{i-1})$ in the cost matrix represent the distance cost if the cluster m in the current frame matches with the cluster n in the previous frame. To filter out the unmatched clusters, distances exceeding the threshold d_{max} are assigned a prohibitively high cost to prevent them from being selected in the optimization process:

$$d_{m,n} = \begin{cases} d_{\mathbf{c}}(\mathbf{c}_m^i, \mathbf{c}_n^{i-1}) & \text{if } d_{\mathbf{c}}(\mathbf{c}_m^i, \mathbf{c}_n^{i-1}) \leq d_{\max} \\ \infty & \text{otherwise} \end{cases} \tag{5.9}$$

It is possible that some previously appearing objects do not appear in the current frame, while new objects might appear. To solve this problem, we add dummy targets to the cost matrix. The cost between dummy targets and real objects is set to a high value, which is $2d_{max}$ as penalty costs while the cost between dummy targets is set to 0. This way, the cost matrix can be extended to $\mathbf{D} \in \mathbb{R}^{(M+M')\times(N+N')}$. We can then use the Hungarian algorithm [130] to solve the assignment problem and get the optimal solution. The Hungarian algorithm is a combinatorial optimization algorithm that solves the assignment problem in polynomial time, ensuring that clusters are tracked across frames:

$$\min \sum_{m=1}^{M+M'} \sum_{n=1}^{N+N'} \alpha_{mn} d_{m,n} \tag{5.10}$$

where: M' and N' represent the number of dummy targets added to the current and previous frames respectively. Subject to the constraints:

$$\sum_{m=1}^{M+M'} \alpha_{mn} = 1 \quad \text{for all } n \in \{1, \ldots, M + M'\} \tag{5.11}$$

$$\sum_{n=1}^{N+N'} \alpha_{mn} = 1 \quad \text{for all } m \in \{1, \ldots, N + N'\} \tag{5.12}$$

$$\alpha_{mn} \in \{0, 1\} \tag{5.13}$$

By employing this method, we guarantee that each cluster in the current frame is matched uniquely with a corresponding cluster from the previous frame, thus preserving the continuity of the tracking process.

5.3.3 Dynamic Object Detection

Movement Analysis

Following the process of cluster association, we can determine all matching clusters for all frames across a given time slot. To analyse the movement of the clusters in current frame, we first create a sliding window of a fixed size, k from the initial to the current frame.

We accumulate the transformation matrix $\mathbf{T}_i$ between frame i and the initial frame. From (5.5), we can get the relative rotation $\mathbf{R}_i$ and translation matrix $\mathbf{t}_i$ between current frame and previous frame. To simplify the following expression, we transform them from SO3 to

SE3 and perform a homogeneous transformation on the cluster's centroid.

$$\mathbf{T} = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0}^T & 1 \end{bmatrix} = \begin{bmatrix} rx_x & ry_x & rz_x & x \\ rx_y & ry_y & rz_y & y \\ rx_z & ry_z & rz_z & z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5.14}$$

$$\mathbf{c}_m^i = [x_m^i, y_m^i, z_m^i, 1]^T \tag{5.15}$$

For frames inside the sliding window, we can get a series of transformation matrices $\mathbf{T}_i$, $i \in \{1, 2, \cdots, k\}$. Then, we can calculate the relative transformation matrix between the frame i and the initial frame:

$$\mathbf{T}_i^w = \prod_{j=1}^{i} \mathbf{T}_j \tag{5.16}$$

Where, $\mathbf{T}_i^w$ is the relative transformation matrix between the frame i and the initial frame, and $\mathbf{T}_j$ is the transformation matrix between the frame j and frame $j-1$.

Then, we can transform the cluster's centroid in the current frame to the initial frame:

$$\mathbf{c}_m^{wi} = \mathbf{T}_i^w \mathbf{c}_m^i \tag{5.17}$$

Here, $\mathbf{c}_m^{wi}$ represents the centroid of cluster m in the ith frame within the initial frame's coordinate system. Having already transformed the centroid of the cluster from the current frame to the initial frame, we can now employ the format used prior to the homogeneous transformation. Consequently, the cluster's centroid in the initial frame can be represented as:

$$\mathbf{c}_m^{wi} = [x_m^{wi}, y_m^{wi}, z_m^{wi}]^T \tag{5.18}$$

where x_m^{wi}, y_m^{wi}, and z_m^{wi} represent the x, y, and z coordinates of the centroid position of cluster m of the ith frame in the initial frame's coordinate system.

We can then track pairs of matched clusters to the initial frame if they exist. Figure 5.3 shows the object tracking principles. We can finally calculate the Euclidean distance between the cluster's centroid in the current frame and the initial frame:

$$f(\mathbf{c}_m^{wk}, \mathbf{c}_m^{w1}) = \| \mathbf{c}_m^{wk} - \mathbf{c}_m^{w1} \|_2 \tag{5.19}$$

if the distance is less than a predefined threshold, ϵ_d, we can consider the cluster as a dynamic object.

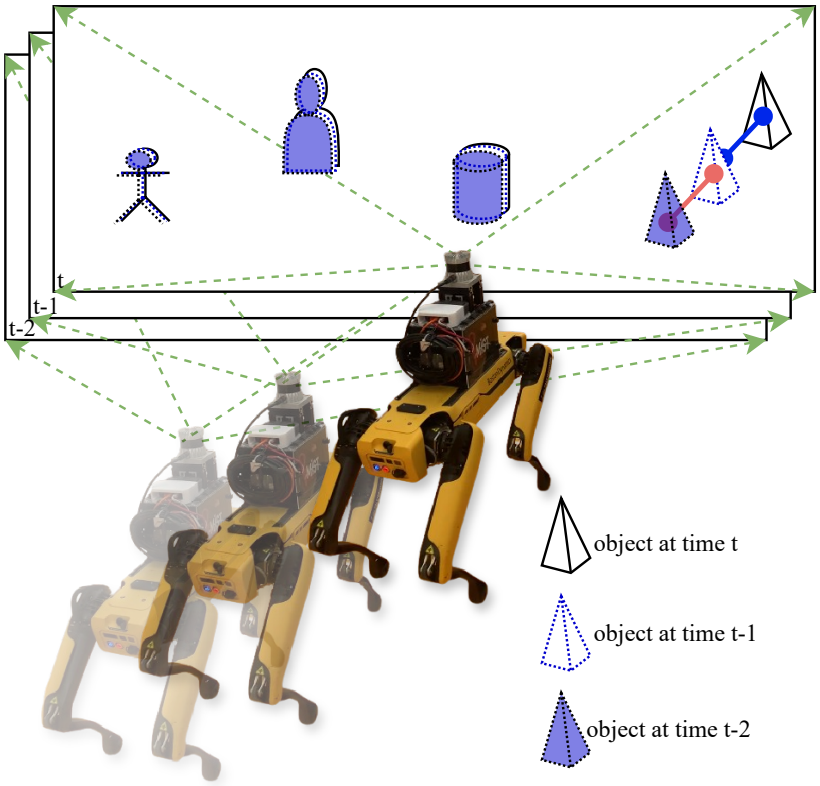


Figure 5.3 Dynamic Object Tracking: to adapt to the drift of the front-end odometry, we accumulate the transformation matrix between the current frame and the first frame within a time window. This approach allows us to ignore historical drift and only account for the drift within the current time window. By transforming all frames into the same coordinate system, we can calculate the distance between the centroids of clusters in consecutive frames and match corresponding clusters between frames. This method enables real-time tracking of all detected foreground objects. By analyzing the movement of clusters within the sliding window, we can effectively detect dynamic objects. In the picture, three frames within the time window are transformed into the initial frame's coordinate system. Although all clusters appear to move due to the odometry drift, the movement of static objects is minimal and usually follows a zero-mean Gaussian distribution. Therefore, we can filter out dynamic objects by accumulating the distance between the cluster's centroid in the current frame and the initial frame.

Outlier Detection

As we only rely on the front-end odometry, its drift may cause some outliers to appear as dynamic objects. To solve this problem, we can calculate the absolute distance of the matched clusters in the current frame:

$$f_a(\mathbf{c}_m^{wk}, \mathbf{c}_m^1) = \sum_{j=2}^{k} \parallel \mathbf{c}_m^{wj} - \mathbf{c}_m^{w(j-1)} \parallel_2 \tag{5.20}$$

if the object is static and the drift noise follows a Gaussian distribution, $f(\mathbf{c}_m^{wk}, \mathbf{c}_m^{w1}) \approx 0$ and $f_a(\mathbf{c}_m^{wk}, \mathbf{c}_m^1) > 0$. So, we can use the ratio of $f(\mathbf{c}_m^{wk}, \mathbf{c}_m^{w1})$ and $f_a(\mathbf{c}_m^{wk}, \mathbf{c}_m^1)$ to filter out the outliers. If the ratio is less than a predefined threshold, ϵ_o, we can consider the cluster as a dynamic object.

In addition to the drifting problem, we are also facing an issue with long objects being mistakenly identified as dynamic objects due to the sparse nature of LiDAR. As the robot moves forward and the LiDAR scans the surface of such objects, the shape of the cluster remains almost the same, causing the center of the object to appear to shift, creating the illusion of movement.

Such clusters tend to grow as the robot moves forward, especially for objects that are long and parallel to the direction of movement. To solve this problem, we propose to calculate the cluster's moving direction and the cluster's main direction and compare the angle between them. If the angle is less than a predefined threshold, ϵ_θ, we can consider the cluster as a dynamic object. For a cluster m in the current frame, we can calculate the moving direction as:

$$\vec{v} = \frac{\mathbf{c}_m^{wk} - \mathbf{c}_m^{w1}}{\parallel \mathbf{c}_m^{wk} - \mathbf{c}_m^{w1} \parallel_2}. \tag{5.21}$$

For the cluster $\mathcal{C}_m$, we can use Principal Component Analysis (PCA) to calculate its main direction $\mathbf{V}$:

$$\mathbf{V} = \text{PCA}(\mathcal{C}_m). \tag{5.22}$$

The angle between the moving direction and the main direction is then

$$\theta = \arccos(\vec{v} \cdot \mathbf{V}). \tag{5.23}$$

If θ is smaller than a predefined threshold, ϵ_θ, we can consider the cluster as an outlier.

Region Growing

After removing the outliers, we have seed points for the dynamic objects in the current frame. After converting the seeds points from the initial coordinate frame to the current, we can get the dynamic objects' center points's pixel position in the raw intensity image (see Figure 5.1, top). A region growing method can be sued on the seed points to segment the whole dynamic object. The basic idea is to start from the seed points and grow the region by adding the neighboring points that have similar properties to the seed points. We use the Euclidean distance between the points to determine whether they are similar. However, finding the neighboring points in the raw point cloud is time-consuming. To solve this problem, we can use the intensity image to find the neighboring points. For each point in the intensity image, we can extract the surrounding pixels directly as neighboring points instead of comparing their Euclidean distance in the raw point cloud. The neighbor points of each seed points' pixel index can be extracted as:

$$N_8(x, y) = \{(x + i, y + j) \mid i, j \in \{-1, 0, 1\}, (i, j) \neq (0, 0)\} \tag{5.24}$$

Where, $N_8(x, y)$ is the 8-neighborhood of the seeds point's pixel (x, y) in the intensity image. The region growing process can be formulated as

$$\mathcal{Q}_{\text{new}} = \mathcal{Q} \cup \{\mathbf{p}_j \mid j \in N_8(i_x, i_y), \exists \mathbf{p}_i \in \mathcal{Q}, \quad \parallel \mathbf{p}_i, \mathbf{p}_j \parallel_2 \leq \epsilon\} \tag{5.25}$$

where, $\mathcal{Q}$ is the set of seed points of the dynamic object, and $\mathbf{p}_j$ is a neighboring point of the seed points. If the distance between the neighboring points and a seed point is less than a predefined threshold, ϵ, we can add the neighboring points to the dynamic object. This process is repeated until all eligible neighboring points are added to the dynamic object.

To prevent regions from absorbing all points we use the ground plane information to filter out neighboring points that do not belong to the dynamic object. Figure 5.1, middle, shows the dynamic object after the region growing process in red. Figure 5.1, bottom, shows the dynamic object points retrieved from the raw point cloud according to the dynamic object's pixel index in the intensity image.

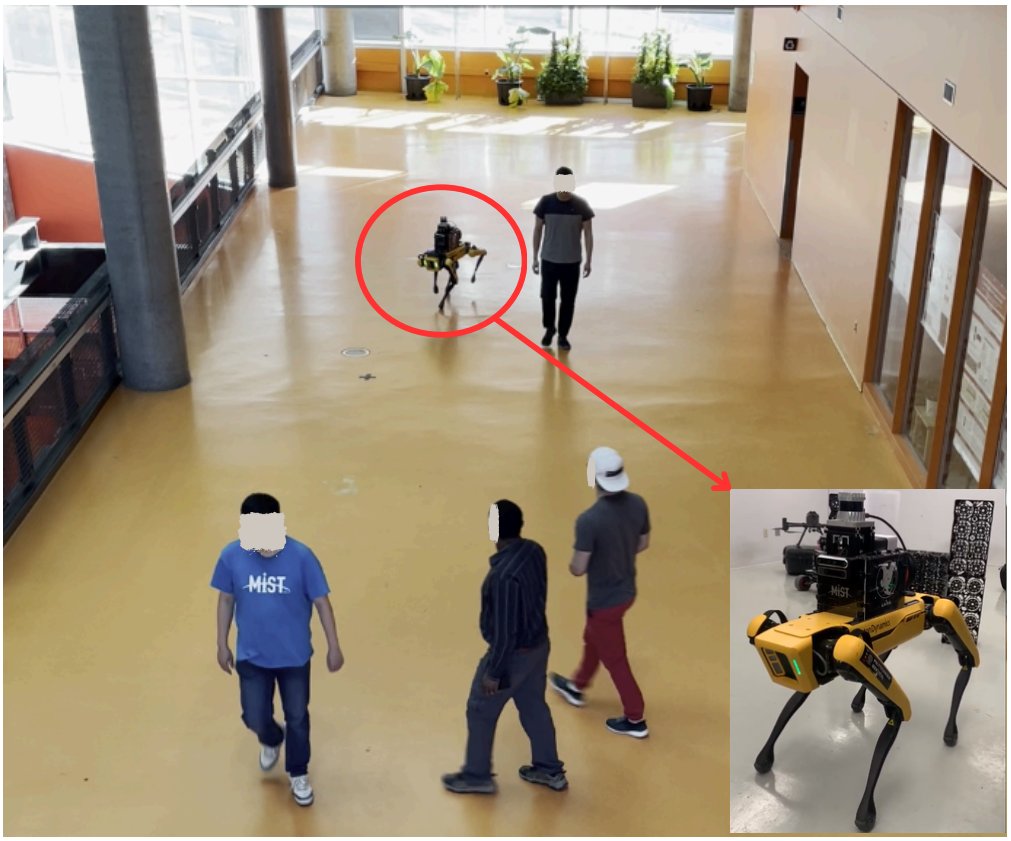


Figure 5.4 Spot robot navigating a dynamic environment. Equipped with an OS0-64 LiDAR sensor, the Spot robot moves through a large area filled with pedestrians as moving objects. Both the pedestrians and the Spot robot exhibit random movements, creating a constantly changing environment that challenges the robot's navigation and object detection capabilities.

5.4 Experimental Evaluation

5.4.1 Dataset

We used a Boston Dynamic Spot robot equipped with an Ouster 64-line LiDAR sensor to collect the dataset. The robot was driven around the floor of a large space with moving objects, shown in Figure 5.4. In normal conditions, the map is blurred by the movement of the pedestrians (Figure 5.5), and the moving objects affect the localization accuracy. We manually labeled the dynamic objects in the dataset to evaluate the performance of the proposed method. The labeled information was stored in the following format: {timestamp, dynamic points index}. The timestamp represents the time when the dynamic objects were detected, and the dynamic points index represents the pixel index of the dynamic objects in the intensity image.

To efficiently label streaming 3D dynamic points, we developed a specialized tool that allows users to label dynamic objects within an intensity image in a streaming style and store the labeled information in binary format.

When labeling dynamic objects, users can check previous and following frames to identify which points are dynamic. Users can then click the center of the dynamic object in the intensity image and apply a region growing method to capture all points belonging to the dynamic object. To aid in this process, the tool provides visual assistance by displaying the extracted foreground points in the intensity image, along with ground information. The labeling tool will be open-sourced[1]

5.4.2 Evaluation Metrics

Due to the time consuming nature of labelling, we only labeled four sequences of the data collected from the Spot robot to assess the performance of the proposed method. The estimated dynamic objects are recorded in the same format as the labeled data. The performance of the proposed method is evaluated by comparing the labeled data with the ground truth provided by labeling. The estimated dynamic objects are showing in figure 5.6 with colorful points.

We used the following metrics to evaluate the performance of the proposed method: precision, recall, IoU, and F1 score [86, 88, 131]. These metrics are commonly used in object detection and tracking tasks and provide a comprehensive evaluation of the method's performance:

- **Precision**: The ratio of correctly identified positive instances to the total instances

[1] `https://github.com/MISTLab/lidar_dynamic_objects_detection.git`

identified as positive.

$$\text{Precision} = \frac{TP}{TP + FP} \tag{5.26}$$

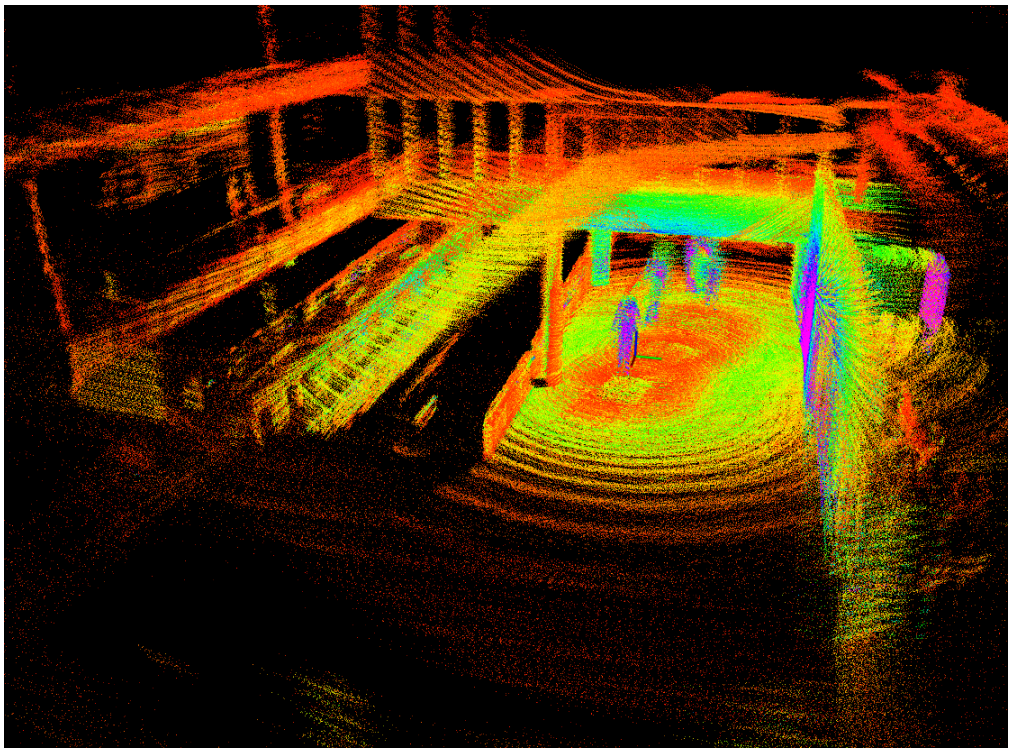


Figure 5.5 Point cloud map of the environment with dynamic objects. The dynamic points create blurring in the map as they move, highlighting the challenges of accurately mapping in dynamic environments.

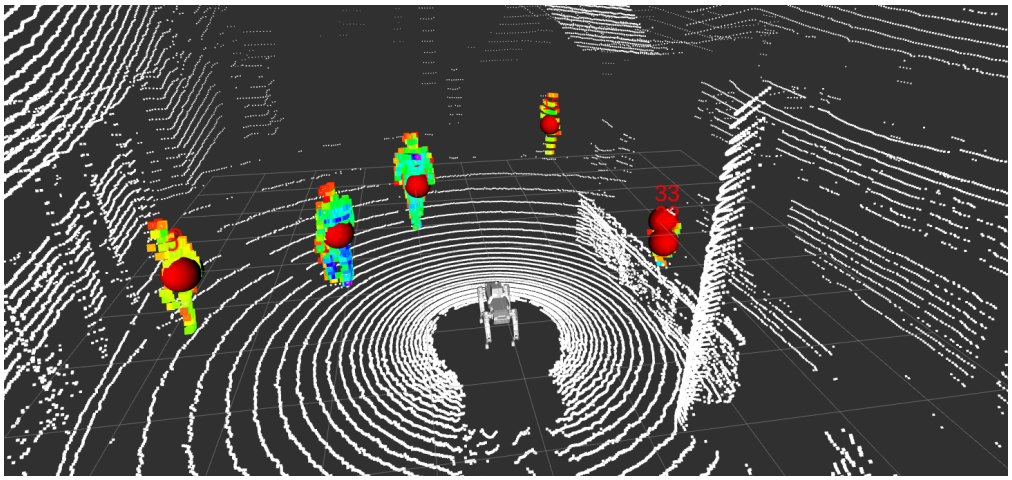


Figure 5.6 Segmentation results of dynamic objects in a single frame. The dynamic objects are segmented from the raw point cloud and displayed with colorful points. The center points of the dynamic objects are marked in red.

- **Recall**: The ratio of correctly identified positive instances to the total actual positive instances.

$$\text{Recall} = \frac{TP}{TP + FN} \tag{5.27}$$

- **IoU**: The ratio of the overlap between the predicted and ground truth bounding boxes to the union of these boxes.

$$\text{IoU} = \frac{\text{Area of Overlap}}{\text{Area of Union}} = \frac{TP}{FN + TP + FP} \tag{5.28}$$

- **F1 Score**: The harmonic mean of precision and recall, providing a single metric that balances both aspects.

$$\text{F1 Score} = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{5.29}$$

Where TP is the number of true positive points, FP is the number of false positive points, and FN is the number of false negative points. The precision, recall, IoU, and F1 score are calculated based on the number of true positive, false positive, and false negative points, providing a comprehensive evaluation of the method's performance.

5.4.3 Experimental Results

We evaluated the performance of the proposed method using the collected dataset and compared it with two state-of-the-art methods: M-detector [86] with FAST-LIO [8] and Removert [93] with a modified A-LOAM [132]. FAST-LIO and A-LOAM provide the robot's movement information to M-detector and Removert, respectively. Additionally, we evaluated the performance of M-detector and Removert using the same front-end-only odometry as our method. The front-end-only odometry drifts significantly compared to FAST-LIO and A-LOAM.

From the data presented in Table 5.1, our method with front-end-only odometry consistently achieves the best performance across all sequences and metrics. The second-best performance is often achieved by the M-detector with FAST-LIO, which shows strong results, particularly in Precision and Recall. Removert with front-end-only odometry consistently underperforms across all metrics and sequences. Taking Sequence 1 in details, M-detector with front-end-only odometry shows a Precision of 0.471 and an IoU of 0.455, with a relatively high Recall of 0.837, but the overall performance is mediocre due to the drift in odometry, which negatively impacts map quality and the ability to examine occlusions.. Removert with front-end-only odometry performs poorly across all metrics. M-detector with FAST-LIO achieves a lower Precision of 0.695 and a very high Recall of 0.915, with the second-best IoU of 0.669, indi-

Table 5.1 Performance metrics comparison across different methods. The best results are highlighted in bold, and the second-best results are underlined. The table compares the precision, IoU (Intersection over Union), recall, and F1 score for M-detector with front-end only odometry, Removert with front-end only odometry, M-detector with FAST-LIO, Removert with A-LOAM, and our proposed method with front-end only odometry across four sequences.

Sequence	**Method**	**Precision**	**IoU**	**Recall**	**F1 Score**
Sequence 1	M-detector w/ front-end only Odom	0.471	0.455	0.837	0.578
	Removert w/ front-end only Odom	0.080	0.063	0.287	0.117
	M-detector w/ FAST-LIO	<u>0.695</u>	<u>0.669</u>	**0.915**	<u>0.775</u>
	Removert w/ A-LOAM	0.651	0.460	0.616	0.610
	Ours w/ front-end only Odom	**0.871**	**0.842**	<u>0.898</u>	**0.875**
Sequence 2	M-detector w/ front-end only Odom	0.486	0.452	0.751	0.547
	Removert w/ front-end only Odom	0.031	0.024	0.141	0.046
	M-detector w/ FAST-LIO	<u>0.775</u>	<u>0.677</u>	<u>0.800</u>	<u>0.759</u>
	Removert w/ A-LOAM	0.766	0.366	0.421	0.524
	Ours w/ front-end only Odom	**0.883**	**0.876**	**0.899**	**0.888**
Sequence 3	M-detector w/ front-end only Odom	0.642	0.588	<u>0.871</u>	0.703
	Removert w/ front-end only Odom	0.149	0.072	0.200	0.134
	M-detector w/ FAST-LIO	<u>0.801</u>	<u>0.728</u>	0.844	<u>0.808</u>
	Removert w/ A-LOAM	0.787	0.376	0.421	0.527
	Ours w/ front-end only Odom	**0.881**	**0.864**	**0.883**	**0.879**
Sequence 4	M-detector w/ front-end only Odom	0.227	0.109	0.243	0.177
	Removert w/ front-end only Odom	0.020	0.017	0.180	0.033
	M-detector w/ FAST-LIO	0.664	<u>0.604</u>	<u>0.814</u>	<u>0.697</u>
	Removert w/ A-LOAM	<u>0.684</u>	0.408	0.525	0.562
	Ours w/ front-end only Odom	**0.911**	**0.853**	**0.883**	**0.887**

cating good performance, but with many false positive points. Removert with A-LOAM has moderate performance with balanced metrics, and its high precision with low recall and IoU means it can detect dynamic objects accurately, but it also misses many dynamic objects. The best performance is from our method with front-end-only odometry, which excels across all metrics with a Precision of 0.871, an IoU of 0.842, a Recall of 0.898, and an F1 Score of 0.875.

Notably, the proposed method relies solely on front-end odometry. The front-end odometry experiences significant drift during robot movement. However, the proposed method still achieves the best performance. This is because the proposed method accumulates the transformation matrix between the current frame and the initial frame within a time window, allowing it to ignore historical drift and account only for the drift within the current time window. Conversely, the other two methods using front-end only odometry perform the worst in all metrics. This is because front-end-only odometry leads to significant drift, affecting the accuracy of mapping. However, M-detector and Removert rely on highly accurate maps to determine if a point is blocked or calculate residuals.

In summary, while all methods suffer from the drift caused by front-end-only odometry, the proposed method's strategy of focusing on a limited time window allows it to mitigate the impact of this drift effectively. This approach significantly enhances its ability to accurately label dynamic objects, as evidenced by its performance across all evaluated metrics and sequences. The reliance of M-detector and Removert on precise maps further exacerbates the performance degradation caused by odometry drift, leading to their lower performance in dynamic detection tasks.

5.4.4 Time Cost

The time cost comparison of the three methods is shown in Figure 5.7. The proposed method integrates front-end odometry and achieves real-time performance. In contrast, the other two methods rely on odometry from external algorithms, and we did not include the external algorithms' time cost in the time cost of M-detector and Removert. Since Removert operates offline, it first uses A-LOAM to generate movement information and scans of the robot's keyframes to files. It then uses these files to build a map and calculate the residuals between scans and the map.

The proposed method maintains a consistently low time cost, well below the real-time threshold throughout the entire time span. This consistent performance highlights the efficiency of the proposed method in integrating front-end odometry and achieving real-time processing.

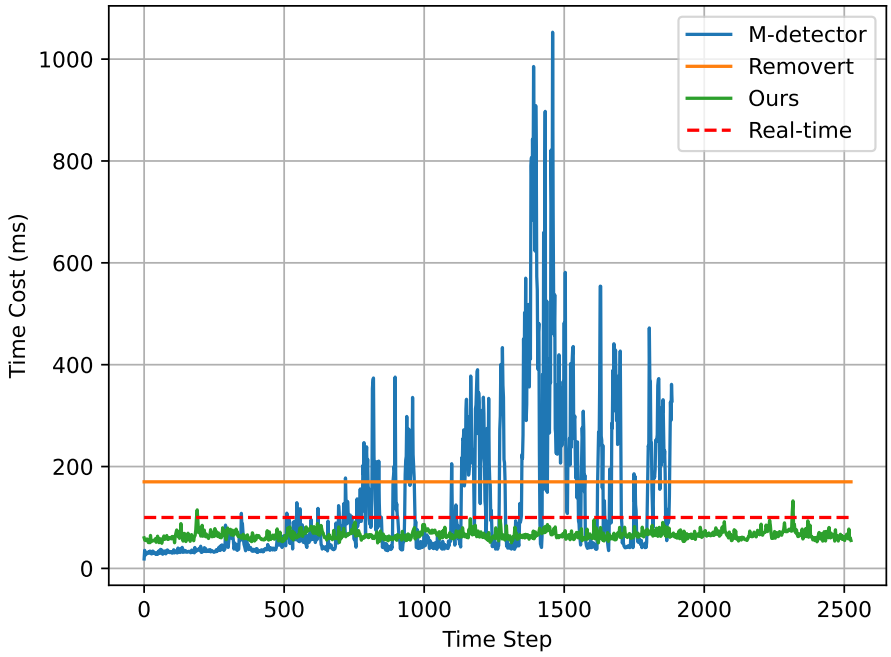


Figure 5.7 Time Cost of Different Methods: The proposed method integrates front-end odometry and achieves real-time performance. In contrast, the other two methods rely on odometry from external algorithms, which is not included in the time cost of M-detector and Removert. Among the three methods, M-detector with FAST-LIO is the fastest for the first 500 frames, but its long-term performance does not match that of the proposed method.

The red dashed line marks the real-time performance threshold. Both M-detector and Removert frequently exceed this threshold, indicating challenges in maintaining real-time performance. In contrast, the proposed method remains consistently below this line, indicating robust real-time performance.

In summary, the proposed method significantly outperforms M-detector and Removert in terms of time cost, maintaining real-time performance throughout the observed period. M-detector shows considerable variability and occasional spikes that disrupt real-time processing, while Removert, despite being more stable than M-detector, consistently exceeds the real-time threshold. These observations underscore the efficiency and stability of the proposed method in dynamic object labeling tasks, leveraging front-end odometry to achieve superior real-time performance.

5.5 Conclusion

In this paper, we propose a real-time dynamic object detection and tracking method for mobile robots equipped with LiDAR sensors. Our approach leverages front-end odometry to enable real-time detection of dynamic objects. By converting unordered LiDAR point cloud data into intensity images and applying a Gaussian Convolution Kernel, our method effectively segments feature points of foreground objects. These segmented points are then clustered into distinct objects and tracked within a sliding time window. This technique accumulates the transformation matrix between the current frame and the initial frame, allowing it to disregard historical drift and focus solely on the drift within the current window.

Our experiments demonstrate that this method achieves superior performance in dynamic object detection across four test sequences, significantly outperforming other state-of-the-art methods. The proposed system shows strong resilience to odometry drift, maintaining high detection accuracy and recall rates. Furthermore, it achieves real-time processing, ensuring reliable operation in dynamic environments over extended periods.

This solution presents a robust and efficient approach to real-time dynamic object detection and tracking, enabling mobile robots to navigate safely and effectively in environments with moving objects. By enhancing the system's capability to manage both static and dynamic features, our method contributes to the development of more accurate and reliable SLAM systems for autonomous robotics.

The proposed method performs well in normal environments but is sensitive to object occlusion. When objects are blocked by other objects, the method loses track of them in the tracking process. Upon reappearance, the method treats them as new objects and tracks them from the initial frame. To address this issue, we can implement a data association method to compare the objects in the current frame with those in the previous 2 or 3 frames. If matches are found, we can maintain continuous tracking of the object from the initial frame. Alternatively, a Kalman filter [133] can be used to predict the object's position when it is occluded. This approach will improve tracking accuracy and reduce tracking loss.

In addition to data association and Kalman filtering, exploring advanced feature matching techniques, such as deep learning-based object re-identification, could further enhance the robustness of the method against occlusion. By combining these strategies, the proposed method can achieve more reliable and precise tracking in dynamic environments, ultimately contributing to the development of more effective SLAM systems for various robotic applications.

Furthermore, integrating the proposed method into a SLAM (Simultaneous Localization and

Mapping) system can enhance localization accuracy. By filtering out dynamic objects in the back-end of the SLAM system, we can eliminate ghost map points and refine the generated map. This integration will leverage the SLAM system's ability to manage static and dynamic features, leading to more robust and accurate environmental mapping and object tracking.

Overall, this section addresses the limitations of our study and proposes future research directions. We suggest advancements in dynamic object tracking with a unstable ego-motion estimation, discuss the broader implications of our findings, and highlight potential applications of our method. This comprehensive overview underscores our contributions to the field of robotics, particularly in autonomous navigation.

Acknowledgment

The authors would like to extend their sincere gratitude to Dong Wang, Konno Genta, and Timothy Lee for their invaluable assistance with the dataset collection. Their expertise and dedication were instrumental in acquiring high-quality data, which was critical to the success of this research. We appreciate their willingness to invest considerable time and effort into this project.

CHAPTER 6 GENERAL DISCUSSION

In this chapter, we assess the impact of our findings on the field. Section 6.1 reviews the general characteristics and advantages of the main body of work presented in previous chapters. Section 6.2 provides a brief overview of the technical challenges encountered during the development of these results, along with the strategies we employed to address these issues.

6.1 General Discussion: Features and Advantages of the Proposed Frameworks

6.1.1 Localization in Degenerate Scenarios

Robust Localization Framework

The first key contribution of this thesis is the development of a robust localization framework (**FW1**) for scenarios with textrue features extracted from LiDAR data, as discussed in Chapters 4. We summarize the core idea as: using intensity information from LiDAR to extract texture features from the environment. These features are then used for ego-motion estimation and loop closure detection. The core benefits of **FW1** include:

- **Performance Guarantees**: Leveraging intensity-based features ensures accurate localization even in challenging environments with insufficient geometric features.
- **Enhanced Loop Closure**: Intensity information improves the robustness of loop closure detection, and further improve the accuracy of the pose graph optimization, reducing drift over time.
- **Adaptability**: The framework is designed to be flexible, making it applicable to a wide range of environments and conditions without extensive reconfiguration.

Handling Geometric Degeneracy

Throughout the development of **FW1**, we focused on addressing the issue of geometric degeneracy, common in environments like long corridors or open spaces. Traditional SLAM systems struggle in such settings due to a lack of distinctive features. By utilizing LiDAR intensity information, we extract reliable texture features that enhance the localization process. This approach fills a critical gap in the existing methods, providing a robust solution where other methods fall short.

6.1.2 Dynamic Object Detection in Real-Time

Real-Time Detection Framework

The second major contribution is the development of a framework (**FW2**) for real-time detection and tracking of dynamic objects, presented in Chapter 5. We summarize the core idea as: using intensity information from LiDAR to identify and track dynamic objects. Key advantages of **FW2** include:

- **Efficient Processing**: By extracting foreground features from intensity data, the framework significantly reduces the computational load, enabling real-time performance.
- **Improved Detection Ability**: The intensity-based features enhance the detection and tracking of dynamic objects, even in complex and dynamic environments.
- **Enhanced Navigation**: The system's ability to detect and track moving objects enhances the robot's navigation capabilities, ensuring safe and efficient movement in dynamic environments.
- **Odometry Drift Compensation**: The framework is designed to maintain the accuracy of detecting dynamic objects despite odometry drift, ensuring reliable performance in real-world scenarios.

Technical Innovations and Applications

In developing **FW2**, we extract foreground features from intensity information to enhance processing efficiency and use intensity-based ego-motion estimation (**FW1**) to provide the necessary odometry information for data association. This approach significantly improves the accuracy and reliability of dynamic object detection, even in challenging scenarios. The techniques developed in this framework have the potential to enhance the performance of various autonomous systems, including self-driving cars, delivery robots, and surveillance drones.

6.2 Technical Challenges and Solutions

6.2.1 Less Structured Environments

One significant challenge in developing a robust SLAM framework for less structured environments is ensuring accurate localization when traditional geometric features are insufficient.

Environments such as long corridors Fig. 6.1, open spaces, or poorly illuminated areas pose considerable difficulties for conventional SLAM systems due to the lack of distinctive geometric features. In the long corridor, there might be enough plane features, but too few edge features to estimate the relative ego-motion movement between two consecutive scans. Unfortunately, aligned plane features (like in the corridor) alone are not sufficient to estimate the robot pose in 6DOF, meaning the robot could lose the sense of forward or backward movement. To solve this problem, one needs an additional reference to constrain the sixth degree of freedom. One possible way is to extract feature points from textural information.

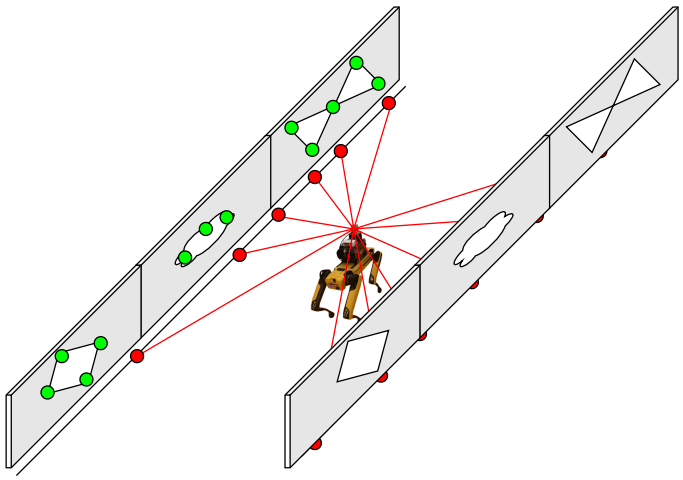


Figure 6.1 In the less structured long corridor environment, the absence of diverse geometric features poses challenges for traditional LiDAR-based SLAM methods. The environment is characterized by enough plane features with few edge features, leading to potential geometric degeneracy. This setting, with its minimal structural complexity, requires advanced techniques to accurately estimate robot poses and maintain reliable navigation. Our novel approach leverages LiDAR intensity images to extract texture feature points (green points) and perform scan registration, ensuring robust localization and mapping even in such unstructured environments.

So we tried to extract ORB feature points from the intensity images, like Fig. 6.3. Beacuse each pixel in the intensity image, which is a 2D image, corresponds to a 3D point in the point cloud, we can use the 2D feature points to get the 3D feature points from the LiDAR data. The 3D feature points can be used to estimate the relative movement between two consecutive scans.

After get the 3D feature points, in the Fig. 6.4, we tried to match them with an existing matching algorithm, TEASER++ [17]. This way we can get the realtive movement between two consecutive scans. However, the time cost of TEASER++ is not stable. It can be very

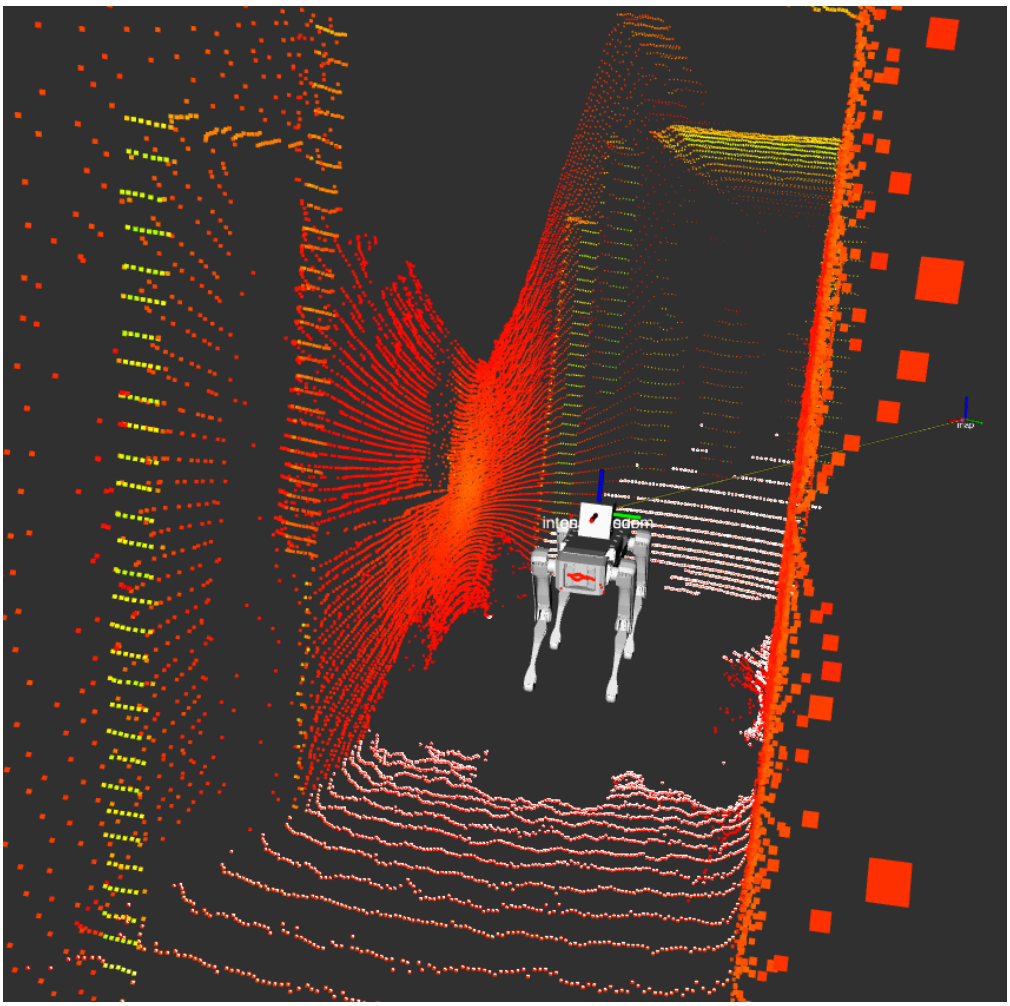


Figure 6.2 Spot in the long corridor environment with point cloud for the environment. The Spot robot navigated through a corridor, surrounded by a detailed point cloud representation of the environment, indicating obstacles and walls in varying colors to signify different intensities.

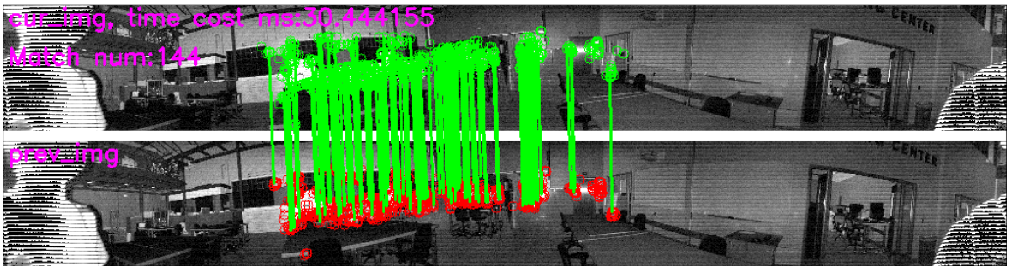


Figure 6.3 ORB feature points extracted from the intensity image for the indoor environment. The red pixels represent the ORB feature points extracted from the first scan, while the green pixels represent the corresponding feature points in the second scan. The green lines represent the matched relationships between the feature points in different frames.

fast (less than 10ms) when the number of points is small enough (around 100 to 300 points in each scan), but very slow (more than 60ms) when the number of points increase from 300 to around 700. This is because the TEASER++ did not take the advantage of the correspondence of matched points. It tries to match all the points in the two scans, which is not necessary. We can use the matched points to estimate the relative movement. That is also the reason why we developed the intensity-odometry to estimate the relative movement between two consecutive scans.

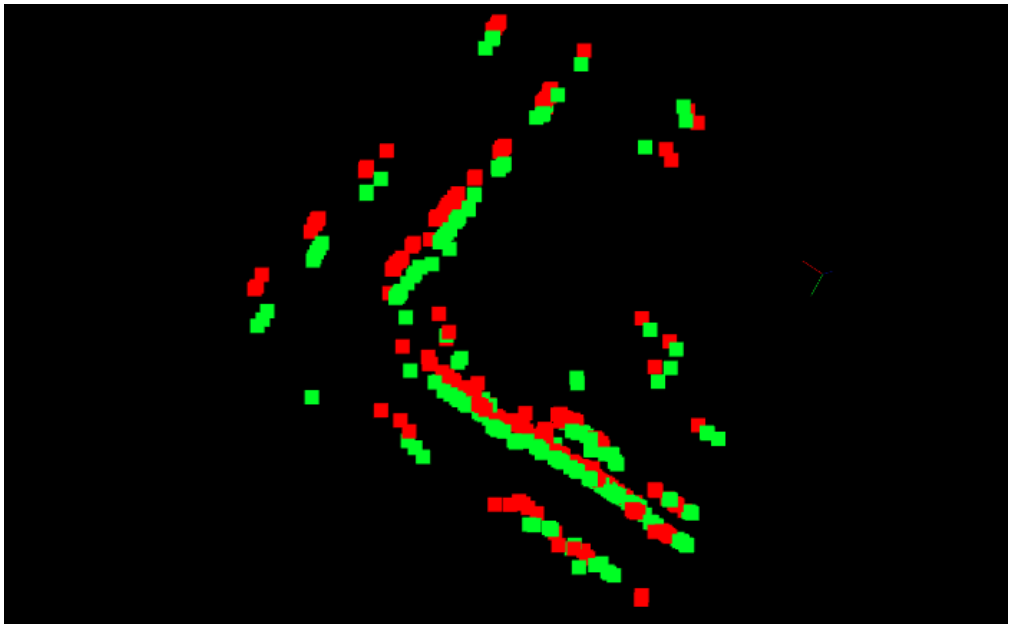


Figure 6.4 3D feature points extracted from the LiDAR data for consecutive scans. The red points represent the feature points in the first scan, while the green points represent the corresponding feature points in the second scan. The green line represent the matched relationships between two points in different frames. The matched points are used to estimate the relative movement between the two scans.

We extract 3D points, $\mathcal{Y}_2 = \{\mathbf{Y}_1, \mathbf{Y}_2, \cdots, \mathbf{Y}_k\}$ directly from the point cloud according to the index of the matched ORB feature points from the intensity image. Each 3D feature point is assigned a score $\mathcal{S} \in \{\mathbf{S}_1, \mathbf{S}_2, \cdots, \mathbf{S}_n\}$ obtained during feature extraction. Similarly, we can extract the corresponding points $\mathcal{X}_2 = \{\mathbf{X}_1, \mathbf{X}_2, \cdots, \mathbf{X}_k\}$ in the following scan, and we scan match as a least squared estimation problem:

$$\bar{\mathbf{X}}_n = \mathbf{R}\mathbf{X}_n + \mathbf{T} \tag{6.1}$$

$$\arg\min_{\mathbf{R},\mathbf{T}} \sum_{n \in \mathcal{N}} \| \left(\mathbf{Y}_n - \bar{\mathbf{X}}_n\right) \|^2 \tag{6.2}$$

where $\mathcal{N} = [1, 2, \cdots, k]$, the $\mathbf{R}$ and $\mathbf{T}$ are the rotation and translation between the two scans. By accumulating the relative movement between two consecutive scans, we can give the robot pose in the global coordinate system.

This approach has been proven to be effective in maintaining accurate localization in not only less structured environments but also normal environment with enough texture informations.

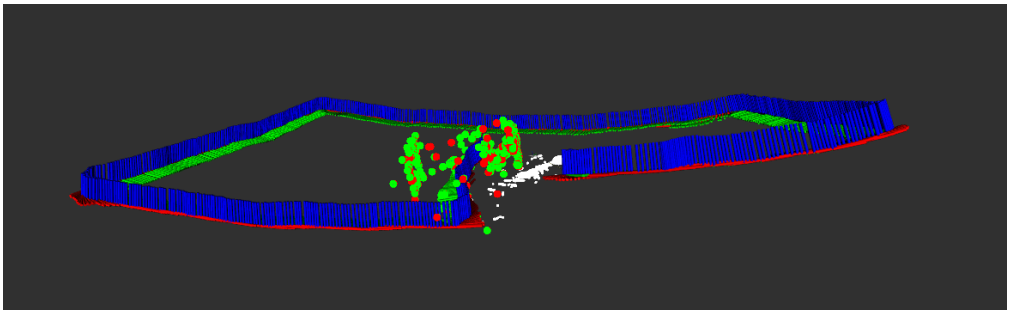


Figure 6.5 Illustration of odometry drift in a font-end odometry. The robot's estimated pose drifts away over time due to accumulated errors in the scan registration process. This drift can lead to inaccurate localization and mapping, affecting the overall performance of the system.

6.2.2 Front-End Odometry Drift

Another significant challenge in SLAM systems is front-end odometry drift, which can lead to inaccurate localization over time. As illustrated in Fig. 6.5, front-end odometry drift results in the end of the trajectory failing to align with the start trajectory, and the trajectory itself appears uneven despite the robot traversing a flat floor. This drift is often caused by errors in the scan registration process, where the robot's movement is estimated based on the LiDAR data.

Once the relative movement between two consecutive scans is determined, this information is used to estimate the robot's pose in the global coordinate system. However, because the relative movement is calculated through an optimization process rather than an analytical solution, it inherently introduces errors. These errors, compounded over time, lead to a gradual drift in the robot's pose estimation. This cumulative error can significantly affect the accuracy of the SLAM system, making it a critical issue to address for maintaining reliable localization.

In our work, we first tried to use the map optimization to correct the drift. The map optimization is a process that do scan to map matching with plane features extracted from the LiDAR data and a Bundle Adjustment to optimize the robot pose after the scan registration

in the back-end. The map optimization can correct the drift a little bit, but the drift still exists. The end of the trajectory is not aligned with the start of the trajectory in the end, as shown in Fig. 6.6.

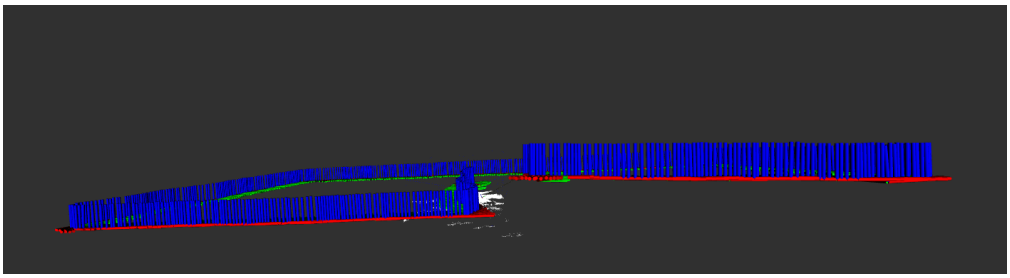


Figure 6.6 Illustration of the improved localization accuracy after map optimization. By scan-to-map matching and Bundle Adjustment, the system corrects odometry drift a little bit. However, the drift still exists. The end of the trajectory is not aligned with the start of the trajectory in the end.

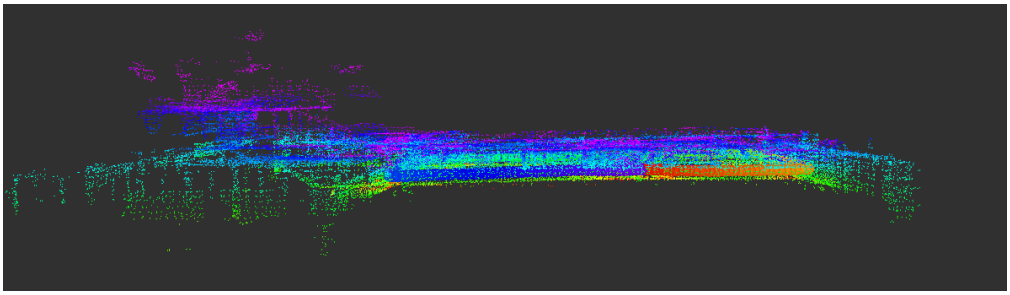


Figure 6.7 Illustration of the improved localization accuracy after pose graph optimization. By incorporating advanced loop closure detection and pose graph optimization techniques, the system corrects odometry drift and maintains accurate localization over time. The mapping result shows that the end of the trajectory is aligned with the start of the trajectory, indicating accurate localization.

To address the issue of the drift after map optimization, we further implemented loop closure detection and pose graph optimization techniques. Loop closure detection involves extracting bag of words (BoW) features for each keyframe and creating a database to store this information. When a new keyframe is added, the system searches the database to find the most similar keyframe. If the similarity exceeds a certain threshold, a loop constraint is added between the new keyframe and the identified similar keyframe.

Pose graph optimization then refines the robot's pose in the global coordinate system by minimizing the errors associated with both loop constraints and odometry constraints. This

optimization process effectively corrects the accumulated drift, ensuring that the trajectory remains accurate and consistent over time. The effectiveness of these techniques is demonstrated in Fig. 6.7, where the corrected trajectory aligns accurately with the start point in the map.

6.2.3 Dynamic Objects Detection

Detecting and tracking dynamic objects in real-time is a challenging task that requires efficient processing and accurate data association. In dynamic environments, objects such as pedestrians, vehicles, or other moving entities can introduce significant complexity to the robot's perception system. In this environment, both robot and object are moving, which makes ego-motion estimation more difficult and makes the data association more complex with an unstable odometry estimation.

For real-time efficiency, we aimed to detect moving objects using only LiDAR data, as cameras may not always be available. LiDAR generates a large 3D point cloud, which is computationally intensive to process and can hinder real-time performance. To address this challenge, we needed a method to downsample the point cloud without losing critical information. Directly downsampling the point cloud was not effective due to the sparse nature of LiDAR data, making it difficult to distinguish objects after downsampling.

Initially, we assumed that all moving objects touched the ground and used Patchwork++ [134] to extract the ground plane, intending to isolate moving objects from the remaining point cloud. However, this method proved unreliable as the ground is not always flat and may include slopes or obstacles, which Patchwork++ struggled to handle accurately.

Next, we explored machine learning methods such as PointNet [29] and RandLA-Net [135] to extract semantic information from the point cloud. Despite their capabilities, these models were unstable for real-time processing, making them unsuitable for our needs.

Finally, we discovered that the low-frequency components of intensity images could accurately represent foreground objects. By extracting these components, we could identify and cluster the foreground points in the LiDAR data. This approach was both fast and accurate, allowing us to track objects across consecutive scans. Since the foreground points constitute a small fraction of the total points in the LiDAR data, the processing time was significantly reduced, enabling real-time performance.

Next, we needed to associate the newly detected objects with those detected in previous frames. Data association is a critical step in tracking dynamic objects, as it ensures continuity across frames. Since moving objects can affect ego-motion estimation, we cannot rely solely

on accurate odometry information. However, without movement information, associating detected objects with their previous counterparts becomes challenging.

To address this, we used front-end intensity odometry to provide the necessary movement information for data association. As discussed earlier, front-end odometry tends to drift over time, but within a short time window, this drift remains tolerable. Therefore, we created a short time window and transformed all object points into the same coordinate system within this window. We then used the Global Nearest Neighbor (GNN) algorithm to associate the newly detected objects with those from previous frames, ensuring reliable and continuous tracking.

However, during experiments, we discovered that static objects, particularly long ones, were sometimes misclassified as moving objects. This issue arises because LiDAR data is sparse, capturing only a few lines of points on long objects. As the robot moves forward, these lines appear to shift as the LiDAR scans the rest of the long objects. Consequently, when these points are transformed into the same coordinate system, the center of these objects seems to move forward, even though they are static.

To address this problem, we incorporated shape information to filter out long static objects. By treating the low-frequency components of the intensity image as representations of foreground objects, the edges of long objects are highlighted. We then calculated the main direction of the feature points of these objects. If the main direction of an object's features is parallel to its perceived movement direction, we classify it as a static object. This approach allows us to accurately filter out long static objects and focus on tracking only the truly moving objects.

CHAPTER 7 CONCLUSION

7.1 Summary of Works

In this thesis, we addressed the critical challenges of achieving accurate and reliable simultaneous localization and mapping (SLAM) in dynamic and unstructured environments. The two core contributions of this research are the development of an innovative LiDAR-based SLAM method that leverages intensity images and the formulation of a robust dynamic object detection and tracking framework.

The first part of the thesis introduced a novel real-time SLAM system that integrates LiDAR intensity images to overcome the limitations of traditional geometric feature-based SLAM. By extracting feature points from LiDAR-generated point clouds and matching them with intensity images, this method significantly improves the robustness and accuracy of SLAM in environments with minimal geometric features. Experimental results demonstrated the system's high accuracy and real-time performance in various challenging scenarios, including environments with low texture and changing illumination.

The second part of the thesis focused on dynamic object detection and tracking using LiDAR data. We proposed a system that emphasizes the extraction of low-frequency components from LiDAR data as feature points for foreground objects. This approach enables real-time tracking and detection of dynamic objects, significantly enhancing the system's resilience to odometry drift. The experimental evaluation showed that the proposed method accurately detects and tracks dynamic objects in real-time, making it suitable for use in urban areas, warehouses, and other dynamic environments.

Together, these contributions provide substantial advancements in the field of autonomous navigation, with potential applications in robotics and autonomous vehicles.

7.2 Limitations

Despite the significant contributions, this research has some limitations that must be acknowledged.

Intensity based LiDAR SLAM system

Firstly, the SLAM system's reliance on LiDAR intensity images, while beneficial in many scenarios, may still face challenges in environments with extremely low or high reflectivity

surfaces, where intensity data may not provide sufficient contrast. This limitation can hinder the system's ability to accurately extract and match feature points, leading to reduced localization and mapping accuracy. For example, highly reflective surfaces, such as metallic objects, can produce saturated intensity values, while very low reflectivity surfaces, like some types of asphalt, can result in insufficient intensity information. These scenarios can complicate the feature extraction process, making it difficult for the SLAM system to maintain robust performance.

Moreover, the current approach might struggle with intensity variations due to changes in environmental lighting conditions or sensor aging. Such variations can affect the consistency of intensity images over time, potentially degrading the SLAM system's accuracy. Developing methods to normalize or adapt to these intensity variations could be an essential area for future research to improve the system's robustness.

In addition to intensity-related challenges, the computational requirements for real-time processing present another limitation. The proposed SLAM system, although optimized for performance, demands significant computational resources to process LiDAR data, extract features, perform scan registration in real-time, and optimize the map and pose graph in the back-end. These requirements can limit the deployment of the system on platforms with constrained computational capabilities, such as small drones or low-cost autonomous vehicles. As the system processes large volumes of data at high frequencies, it may necessitate the use of high-performance processors or dedicated hardware accelerators, which might not be feasible for all applications.

Furthermore, the integration of additional sensor data, such as IMU or visual information, can exacerbate the computational load, requiring even more advanced hardware solutions. Balancing the need for high accuracy with computational efficiency is a significant challenge that must be addressed to make the system more accessible for a broader range of applications. Exploring more efficient algorithms and leveraging advancements in hardware, such as parallel processing on GPUs or specialized AI accelerators, could help mitigate these computational challenges. Additionally, investigating lightweight and adaptive SLAM algorithms that can dynamically adjust their computational requirements based on the available resources could enhance the system's flexibility and applicability.

While the SLAM system leveraging LiDAR intensity images offers substantial benefits, it faces challenges related to intensity variations and computational demands. Addressing these limitations through further research and technological advancements will be crucial for improving the system's robustness and expanding its deployment across various real-world applications.

Dynamic Object Detection and Tracking

The proposed dynamic object detection and tracking method performs well in normal environments but is sensitive to object occlusion. When objects are blocked by other objects, the method loses track of them in the tracking process. Upon reappearance, the method treats them as new objects and tracks them from the initial frame. To address this issue, implementing a data association method to compare the objects in the current frame with those in the previous 2 or 3 frames could help. If matches are found, continuous tracking of the object from the initial frame can be maintained.

Alternatively, a Kalman filter can be used to predict the object's position when it is occluded. This approach will improve tracking accuracy and reduce tracking loss. In addition to data association and Kalman filtering, exploring advanced feature matching techniques, such as deep learning-based object re-identification, could further enhance the robustness of the method against occlusion. By combining these strategies, the proposed method can achieve more reliable and precise tracking in dynamic environments, ultimately contributing to the development of more effective SLAM systems for various robotic applications.

Lastly, the experimental evaluations were conducted in controlled environments. Real-world deployments may present unforeseen challenges that were not encountered in the test scenarios. Hence, further field testing is necessary to validate the robustness and adaptability of the proposed methods in diverse real-world settings.

7.3 Future Research

Building on the findings of this thesis, several avenues for future research can be pursued to further enhance the capabilities of LiDAR-based SLAM and dynamic object detection systems.

Firstly, integrating additional sensor modalities, such as cameras or radar, could complement LiDAR data, providing richer environmental information and improving system robustness. Multi-sensor fusion techniques could leverage the strengths of each sensor type, enabling more reliable operation in diverse conditions.

Secondly, advancements in machine learning and deep learning techniques offer promising opportunities to enhance feature extraction, segmentation, and classification processes within the SLAM system. Developing adaptive algorithms that can learn from the environment and improve over time could significantly enhance the system's performance and reliability.

Thirdly, extending the dynamic object detection framework to handle more complex scenar-

ios, such as occlusions, overlapping objects, and varying object velocities, would be beneficial. Incorporating advanced tracking algorithms and leveraging temporal information across multiple frames could improve detection accuracy in highly dynamic environments.

A crucial next step is the fusion of **FW1** and **FW2** to filter out dynamic objects, thereby further enhancing the SLAM localization accuracy proposed in **FW1**. We have already integrated the intensity odometry component of **FW1** into **FW2**. The next phase involves feeding the dynamic object detection results back into the map optimization component of **FW1**, which will optimize the SLAM system by excluding the influence of dynamic objects.

Lastly, conducting extensive field tests in various real-world settings, such as urban areas, industrial sites, and natural environments, would provide valuable insights into the system's performance and identify potential areas for improvement. Collaboration with industry partners could facilitate the practical deployment of the proposed methods, contributing to the advancement of autonomous navigation technologies.

REFERENCES

[1] C. Campos *et al.*, "ORB-SLAM3: An Accurate Open-Source Library for Visual, Visual–Inertial, and Multimap SLAM," *IEEE Transactions on Robotics*, 2021.

[2] T. Shan *et al.*, "LVI-SAM: Tightly-coupled lidar-visual-inertial odometry via smoothing and mapping," in *2021 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2021, pp. 5692–5698.

[3] H. Taheri and Z. C. Xia, "Slam; definition and evolution," *Engineering Applications of Artificial Intelligence*, vol. 97, p. 104032, 2021.

[4] T. Taketomi, H. Uchiyama, and S. Ikeda, "Visual slam algorithms: A survey from 2010 to 2016," *IPSJ transactions on computer vision and applications*, vol. 9, pp. 1–11, 2017.

[5] H. Matsuki *et al.*, "Gaussian splatting slam," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2024, pp. 18 039–18 048.

[6] A. Rosinol, J. J. Leonard, and L. Carlone, "Nerf-slam: Real-time dense monocular slam with neural radiance fields," in *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2023, pp. 3437–3444.

[7] H. Oleynikova *et al.*, "Voxblox: Incremental 3d euclidean signed distance fields for on-board mav planning," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017, pp. 1366–1373.

[8] W. Xu *et al.*, "FAST-LIO2: Fast direct lidar-inertial odometry," *IEEE Transactions on Robotics*, 2022.

[9] R. Smith, M. Self, and P. Cheeseman, "Estimating uncertain spatial relationships in robotics," in *Autonomous robot vehicles*. Springer, 1990, pp. 167–193.

[10] B. Huang, J. Zhao, and J. Liu, "A survey of simultaneous localization and mapping with an envision in 6g wireless networks," *arXiv preprint arXiv:1909.05214*, 2019.

[11] J. Zhang and S. Singh, "LOAM: Lidar odometry and mapping in real-time." in *Robotics: Science and Systems*, vol. 2, no. 9, 2014.

[12] T. Shan and B. Englot, "LeGO-LOAM: Lightweight and ground-optimized lidar odometry and mapping on variable terrain," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 4758–4765.

[13] T. Shan *et al.*, "LIO-SAM: Tightly-coupled lidar inertial odometry via smoothing and mapping," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2020, pp. 5135–5142.

[14] W. Hess *et al.*, "Real-time loop closure in 2D LIDAR SLAM," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2016, pp. 1271–1278.

[15] K. Chen *et al.*, "Direct LiDAR Odometry: Fast Localization with Dense Point Clouds," *arXiv preprint arXiv:2110.00605*, 2021.

[16] I. Vizzo *et al.*, "KISS-ICP: In defense of point-to-point icp–simple, accurate, and robust registration if done the right way," *IEEE Robotics and Automation Letters*, vol. 8, no. 2, pp. 1029–1036, 2023.

[17] H. Yang, J. Shi, and L. Carlone, "TEASER: Fast and Certifiable Point Cloud Registration," *IEEE Trans. Robotics*, 2020.

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

[19] W. Xu and F. Zhang, "FAST-LIO: A fast, robust lidar-inertial odometry package by tightly-coupled iterated kalman filter," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 3317–3324, 2021.

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

[21] C. Qin *et al.*, "Lins: A lidar-inertial state estimator for robust and efficient navigation," in *2020 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2020, pp. 8899–8906.

[22] B. M. Bell and F. W. Cathey, "The iterated kalman filter update as a gauss-newton method," *IEEE Transactions on Automatic Control*, vol. 38, no. 2, pp. 294–297, 1993.

[23] H. Tang *et al.*, "Ff-lins: A consistent frame-to-frame solid-state-lidar-inertial state estimator," *IEEE Robotics and Automation Letters*, 2023.

[24] Z. Chen *et al.*, "Relead: Resilient localization with enhanced lidar odometry in adverse environments," *arXiv preprint arXiv:2402.18934*, 2024.

[25] F. L. Markley, "Attitude error representations for kalman filtering," *Journal of guidance, control, and dynamics*, vol. 26, no. 2, pp. 311–317, 2003.

[26] H. Yang *et al.*, "Graduated non-convexity for robust spatial perception: From non-minimal solvers to global outlier rejection," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 1127–1134, 2020.

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

[28] S. Arshad and G.-W. Kim, "Role of deep learning in loop closure detection for visual and lidar slam: A survey," *Sensors*, vol. 21, no. 4, 2021. [Online]. Available: https://www.mdpi.com/1424-8220/21/4/1243

[29] C. R. Qi *et al.*, "PointNet: A 3D Convolutional Neural Network for real-time object class recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 652–660.

[30] ——, "PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space," *Advances in neural information processing systems*, vol. 30, 2017.

[31] R. Dubé *et al.*, "SegMatch: Segment based place recognition in 3D point clouds," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2017, pp. 5266–5272.

[32] M. A. Uy and G. H. Lee, "PointNetVLAD: Deep Point Cloud Based Retrieval for Large-Scale Place Recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 4470–4479.

[33] R. Arandjelovic *et al.*, "NetVLAD: CNN Architecture for Weakly Supervised Place Recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 5297–5307.

[34] G. Kim and A. Kim, "Scan context: Egocentric spatial descriptor for place recognition within 3d point cloud map," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 4802–4809.

[35] D. Cattaneo, M. Vaghi, and A. Valada, "LCDNet: Deep Loop Closure Detection and Point Cloud Registration for LiDAR SLAM," *IEEE Transactions on Robotics*, vol. 38, no. 4, pp. 2074–2093, 2022.

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

[37] K. Vidanapathirana *et al.*, "LoGG3D-Net: Locally guided global descriptor learning for 3d place recognition," in *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, 2022, pp. 2215–2221.

[38] X. Chen *et al.*, "Suma++: Efficient lidar-based semantic slam," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2019, pp. 4530–4537.

[39] Q. Li *et al.*, "LO-Net: Deep real-time lidar odometry," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 8473–8482.

[40] X. Zhong *et al.*, "SHINE-Mapping: Large-scale 3d mapping using sparse hierarchical implicit neural representations," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2023, pp. 8371–8377.

[41] T. D. Barfoot *et al.*, "Into Darkness: Visual Navigation Based on a Lidar-Intensity-Image Pipeline," in *Robotics Research: The 16th International Symposium ISRR*. Springer, 2016, pp. 487–504.

[42] T. Shan *et al.*, "Robust Place Recognition using an Imaging Lidar," *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 5469–5475, 2021.

[43] T. Guadagnino *et al.*, "Fast sparse lidar odometry using self-supervised feature selection on intensity images," *IEEE Robotics and Automation Letters*, vol. 7, no. 3, pp. 7597–7604, 2022.

[44] R. A. Hewitt and J. A. Marshall, "Towards intensity-augmented slam with lidar and tof sensors," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2015, pp. 1956–1961.

[45] H. Wang, C. Wang, and L. Xie, "Intensity-SLAM: Intensity assisted localization and mapping for large scale environment," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 1715–1721, 2021.

[46] H. Li *et al.*, "An intensity-augmented lidar-inertial slam for solid-state lidars in degenerated environments," *IEEE Transactions on Instrumentation and Measurement*, vol. 71, pp. 1–10, 2022.

[47] M. Jung *et al.*, "Intensity and ambient enhanced lidar-inertial slam for unstructured construction environment," *The Journal of Korea Robotics Society*, vol. 16, no. 3, pp. 179–188, 2021.

[48] P. Pfreundschuh *et al.*, "Coin-lio: Complementary intensity-augmented lidar inertial odometry," *arXiv preprint arXiv:2310.01235*, 2023.

[49] Z. Dai *et al.*, "An intensity-enhanced lidar slam for unstructured environments," *Measurement Science and Technology*, vol. 34, no. 12, p. 125120, 2023.

[50] H. Wang, C. Wang, and L. Xie, "Intensity scan context: Coding intensity and geometry relations for loop closure detection," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 2095–2101.

[51] L. Di Giammarino *et al.*, "Visual place recognition using lidar intensity information," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2021, pp. 4382–4389.

[52] X. Zhang *et al.*, "A lidar-intensity slam and loop closure detection method using an intensity cylindrical-projection shape context descriptor," *International Journal of Applied Earth Observation and Geoinformation*, vol. 122, p. 103419, 2023.

[53] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardos, "ORB-SLAM: A Versatile and Accurate Monocular SLAM System," *IEEE transactions on robotics*, vol. 31, no. 5, pp. 1147–1163, 2015.

[54] R. Mur-Artal and J. D. Tardós, "ORB-SLAM2: An Open-Source SLAM System for Monocular, Stereo, and RGB-D Cameras," *IEEE Transactions on Robotics*, vol. 33, no. 5, pp. 1255–1262, 2017.

[55] E. Rublee *et al.*, "Orb: An efficient alternative to sift or surf," in *2011 International conference on computer vision*. Ieee, 2011, pp. 2564–2571.

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

[57] T. Qin, P. Li, and S. Shen, "VINS-Mono: A robust and versatile monocular visual-inertial state estimator," *IEEE Transactions on Robotics*, vol. 34, no. 4, pp. 1004–1020, 2018.

[58] T. Qin *et al.*, "A general optimization-based framework for global pose estimation with multiple sensors," *arXiv preprint arXiv:1901.03642*, 2019.

[59] Z. Xiao *et al.*, "Effloc: Lightweight vision transformer for efficient 6-dof camera relocalization," *arXiv preprint arXiv:2402.13537*, 2024.

[60] N. Parmar *et al.*, "Image transformer," in *International conference on machine learning*. PMLR, 2018, pp. 4055–4064.

[61] J. Li *et al.*, "Next-vit: Next generation vision transformer for efficient deployment in realistic industrial scenarios," *arXiv preprint arXiv:2207.05501*, 2022.

[62] B. Wang *et al.*, "Atloc: Attention guided camera localization," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 06, 2020, pp. 10 393–10 401.

[63] S. Brahmbhatt *et al.*, "Geometry-aware learning of maps for camera localization," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 2616–2625.

[64] H. Rebecq *et al.*, "Evo: A geometric approach to event-based 6-dof parallel tracking and mapping in real time," *IEEE Robotics and Automation Letters*, vol. 2, no. 2, pp. 593–600, 2016.

[65] C. Forster *et al.*, "Svo: Semidirect visual odometry for monocular and multicamera systems," *IEEE Transactions on Robotics*, vol. 33, no. 2, pp. 249–265, 2016.

[66] S. Y. Loo *et al.*, "Cnn-svo: Improving the mapping in semi-direct visual odometry using single-image depth prediction," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 5218–5223.

[67] J. Engel, V. Koltun, and D. Cremers, "Direct sparse odometry," *IEEE transactions on pattern analysis and machine intelligence*, vol. 40, no. 3, pp. 611–625, 2017.

[68] R. A. Newcombe, S. J. Lovegrove, and A. J. Davison, "Dtam: Dense tracking and mapping in real-time," in *2011 international conference on computer vision*. IEEE, 2011, pp. 2320–2327.

[69] C. Kerl, J. Sturm, and D. Cremers, "Dense visual slam for rgb-d cameras," in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2013, pp. 2100–2106.

[70] R. A. Newcombe, D. Fox, and S. M. Seitz, "Dynamicfusion: Reconstruction and tracking of non-rigid scenes in real-time," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 343–352.

[71] W. N. Greene *et al.*, "Multi-level mapping: Real-time dense monocular slam," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2016, pp. 833–840.

[72] J. Czarnowski *et al.*, "Deepfactors: Real-time probabilistic dense monocular slam," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 721–728, 2020.

[73] K. M. Jatavallabhula, G. Iyer, and L. Paull, "∇slam: Dense slam meets automatic differentiation," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 2130–2137.

[74] D. Xu, D. Anguelov, and A. Jain, "Pointfusion: Deep sensor fusion for 3d bounding box estimation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 244–253.

[75] X. Zuo *et al.*, "Lic-fusion: Lidar-inertial-camera odometry," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2019, pp. 5848–5854.

[76] J. Zhang and S. Singh, "Visual-lidar odometry and mapping: low-drift, robust, and fast," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, 2015, pp. 2174–2181.

[77] J. Lin *et al.*, "R2LIVE: A robust, real-time, lidar-inertial-visual tightly-coupled state estimator and mapping," *arXiv preprint arXiv:2102.12400*, 2021.

[78] J. Lin and F. Zhang, "R3LIVE: A robust, real-time, rgb-colored, lidar-inertial-visual tightly-coupled state estimation and mapping package," *arXiv preprint arXiv:2109.07982*, 2021.

[79] B. Li *et al.*, "Self-supervised visual-lidar odometry with flip consistency," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 2021, pp. 3844–3852.

[80] X. Wu *et al.*, "LHMap-loc: Cross-modal monocular localization using lidar point cloud heat map," *arXiv preprint arXiv:2403.05002*, 2024.

[81] A. Azim and O. Aycard, "Detection, classification and tracking of moving objects in a 3d environment," in *2012 IEEE Intelligent Vehicles Symposium*. IEEE, 2012, pp. 802–807.

[82] J. Schauer and A. Nüchter, "The peopleremover—removing dynamic objects from 3-d point cloud data by traversing a voxel occupancy grid," *IEEE robotics and automation letters*, vol. 3, no. 3, pp. 1679–1686, 2018.

[83] S. Thrun, "Learning occupancy grid maps with forward sensor models," *Autonomous robots*, vol. 15, pp. 111–127, 2003.

[84] A. Hornung *et al.*, "Octomap: An efficient probabilistic 3d mapping framework based on octrees," *Autonomous robots*, vol. 34, pp. 189–206, 2013.

[85] L. Schmid *et al.*, "Khronos: A unified approach for spatio-temporal metric-semantic slam in dynamic environments," *arXiv preprint arXiv:2402.13817*, 2024.

[86] H. Wu *et al.*, "Moving event detection from lidar point streams," *Nature Communications*, vol. 15, no. 1, p. 345, 2024.

[87] H. Lim, S. Hwang, and H. Myung, "ERASOR: Egocentric ratio of pseudo occupancy-based dynamic object removal for static 3d point cloud map building," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 2272–2279, 2021.

[88] H. Lim *et al.*, "Erasor2: Instance-aware robust 3d mapping of the static world in dynamic scenes," in *Robotics: Science and Systems (RSS 2023)*. IEEE, 2023.

[89] F. Pomerleau *et al.*, "Long-term 3d map maintenance in dynamic environments," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2014, pp. 3712–3719.

[90] N. Banerjee *et al.*, "Lifelong mapping using adaptive local maps," in *2019 European Conference on Mobile Robots (ECMR)*. IEEE, 2019, pp. 1–8.

[91] R. Ambruş *et al.*, "Meta-rooms: Building and maintaining long term spatial models in a dynamic world," in *2014 IEEE/RSJ international conference on intelligent robots and systems*. IEEE, 2014, pp. 1854–1861.

[92] C. Jiang *et al.*, "Static-map and dynamic object reconstruction in outdoor scenes using 3-d motion segmentation," *IEEE Robotics and Automation Letters*, vol. 1, no. 1, pp. 324–331, 2016.

[93] G. Kim and A. Kim, "Remove, then revert: Static point cloud map construction using multiresolution range images," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2020, pp. 10 758–10 765.

[94] E. Palazzolo *et al.*, "Refusion: 3d reconstruction in dynamic environments for rgb-d cameras exploiting residuals," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2019, pp. 7855–7862.

[95] L. Schmid *et al.*, "Dynablox: Real-time detection of diverse dynamic objects in complex environments," *IEEE Robotics and Automation Letters*, 2023.

[96] B. Curless and M. Levoy, "A volumetric method for building complex models from range images," in *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, 1996, pp. 303–312.

[97] L. Nunes *et al.*, "Unsupervised class-agnostic instance segmentation of 3d lidar data for autonomous vehicles," *IEEE Robotics and Automation Letters*, vol. 7, no. 4, pp. 8713–8720, 2022.

[98] H. Thomas *et al.*, "Learning spatiotemporal occupancy grid maps for lifelong navigation in dynamic scenes," in *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, 2022, pp. 484–490.

[99] ——, "Kpconv: Flexible and deformable convolution for point clouds," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2019, pp. 6411–6420.

[100] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *Medical image computing and computer-assisted intervention–MICCAI 2015: 18th international conference, Munich, Germany, October 5-9, 2015, proceedings, part III 18*. Springer, 2015, pp. 234–241.

[101] X. Chen *et al.*, "Moving object segmentation in 3d lidar data: A learning-based approach exploiting sequential data," *IEEE Robotics and Automation Letters*, vol. 6, no. 4, pp. 6529–6536, 2021.

[102] A. Milioto *et al.*, "Rangenet++: Fast and accurate lidar semantic segmentation," in *2019 IEEE/RSJ international conference on intelligent robots and systems (IROS)*. IEEE, 2019, pp. 4213–4220.

[103] S. Li *et al.*, "Multi-scale interaction for real-time lidar data segmentation on an embedded platform," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 738–745, 2021.

[104] T. Cortinhal, G. Tzelepis, and E. Erdal Aksoy, "Salsanext: Fast, uncertainty-aware semantic segmentation of lidar point clouds," in *Advances in Visual Computing: 15th International Symposium, ISVC 2020, San Diego, CA, USA, October 5–7, 2020, Proceedings, Part II 15*. Springer, 2020, pp. 207–222.

[105] J. Sun *et al.*, "Efficient spatial-temporal information fusion for lidar-based 3d moving object segmentation," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2022, pp. 11 456–11 463.

[106] B. Mersch *et al.*, "Receding moving object segmentation in 3d lidar data using sparse 4d convolutions," *IEEE Robotics and Automation Letters*, vol. 7, no. 3, pp. 7503–7510, 2022.

[107] C. Choy, J. Gwak, and S. Savarese, "4d spatio-temporal convnets: Minkowski convolutional neural networks," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 3075–3084.

[108] Y. Cao and G. Beltrame, "VIR-SLAM: Visual, inertial, and ranging slam for single and multi-robot systems," *Autonomous Robots*, vol. 45, no. 6, pp. 905–917, 2021.

[109] Q. Tong and C. Shaozu, "A-LOAM: Advanced implementation of loam," https://github.com/HKUST-Aerial-Robotics/A-LOAM, 2019.

[110] K. Ebadi *et al.*, "DARE-SLAM: Degeneracy-aware and resilient loop closing in perceptually-degraded environments," *Journal of Intelligent & Robotic Systems*, vol. 102, pp. 1–25, 2021.

[111] K. Chen *et al.*, "Direct LiDAR Odometry: Fast Localization with Dense Point Clouds," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 2000–2007, 2022.

[112] J. Lin and F. Zhang, "A fast, complete, point cloud based loop closure for lidar odometry and mapping," *arXiv preprint arXiv:1909.11811*, 2019.

[113] X. Chen *et al.*, "SuMa++: Efficient LiDAR-based Semantic SLAM," in *Proceedings of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2019.

[114] ——, "OverlapNet: A siamese network for computing LiDAR scan similarity with applications to loop closing and localization," *Autonomous Robots*, pp. 1–21, 2022.

[115] Z. Liu and F. Zhang, "BALM: Bundle Adjustment for Lidar Mapping," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 3184–3191, 2021.

[116] S. Rusinkiewicz and M. Levoy, "Efficient variants of the icp algorithm," in *Proceedings third international conference on 3-D digital imaging and modeling*. IEEE, 2001, pp. 145–152.

[117] Y. Cai, W. Xu, and F. Zhang, "ikd-Tree: An incremental KD tree for robotic applications," *arXiv preprint arXiv:2102.10808*, 2021.

[118] R. Kümmerle *et al.*, "G2o: A general framework for graph optimization," in *2011 IEEE International Conference on Robotics and Automation*. IEEE, 2011, pp. 3607–3613.

[119] T. K. Kim, "T test as a parametric statistic," *Korean journal of anesthesiology*, vol. 68, no. 6, pp. 540–546, 2015.

[120] P. Furgale *et al.*, "Toward automated driving in cities using close-to-market sensors: An overview of the v-charge project," in *2013 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2013, pp. 809–816.

[121] H. Peng, Z. Zhao, and L. Wang, "A Review of Dynamic Object Filtering in SLAM Based on 3D LiDAR," *Sensors*, vol. 24, no. 2, p. 645, 2024.

[122] J. Redmon *et al.*, "You Only Look Once: Unified, Real-Time Object Detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 779–788.

[123] J. Redmon and A. Farhadi, "YOLO9000: better, faster, stronger," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 7263–7271.

[124] A. Farhadi and J. Redmon, "YOLOv3: An incremental improvement," in *Computer vision and pattern recognition*, vol. 1804. Springer Berlin/Heidelberg, Germany, 2018, pp. 1–6.

[125] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, "YOLOv4: Optimal speed and accuracy of object detection," *arXiv preprint arXiv:2004.10934*, 2020.

[126] Z. Ge *et al.*, "YOLOX: Exceeding YOLO Series in 2021," *arXiv preprint arXiv:2107.08430*, 2021.

[127] E. Arani *et al.*, "A comprehensive study of real-time object detection networks across multiple domains: A survey," *arXiv preprint arXiv:2208.10895*, 2022.

[128] W. Du and G. Beltrame, "Real-time simultaneous localization and mapping with lidar intensity," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2023, pp. 4164–4170.

[129] Y. Bar-Shalom and X.-R. Li, *Multitarget-multisensor tracking: principles and techniques*. YBS publishing Storrs, CT, 1995, vol. 19.

[130] H. W. Kuhn, "The hungarian method for the assignment problem," *Naval research logistics quarterly*, vol. 2, no. 1-2, pp. 83–97, 1955.

[131] C. Goutte and E. Gaussier, "A probabilistic interpretation of precision, recall and f-score, with implication for evaluation," in *European conference on information retrieval*. Springer, 2005, pp. 345–359.

[132] G. Kim, "SC-A-LOAM," https://github.com/gisbi-kim/SC-A-LOAM, 2021, accessed: 2021-07-16.

[133] R. E. Kalman, "A new approach to linear filtering and prediction problems," 1960.

[134] S. Lee, H. Lim, and H. Myung, "Patchwork++: Fast and robust ground segmentation solving partial under-segmentation using 3d point cloud," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2022, pp. 13 276–13 283.

[135] Q. Hu *et al.*, "RandLA-Net: Efficient semantic segmentation of large-scale point clouds," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 11 108–11 117.

[136] F. Wilcoxon, "Individual comparisons by ranking methods," in *Breakthroughs in statistics: Methodology and distribution*. Springer, 1992, pp. 196–202.

APPENDIX A ANNOTATION TOOL

To efficiently label dynamic objects in streaming 3D point clouds, we developed an advanced annotation tool that directly labels dynamic objects in intensity images. This tool allows users to annotate dynamic objects within intensity images in a streaming style, storing the labeled data in a predefined format. During the dataset collection process, the point cloud is indexed by the pixel positions in the intensity image, facilitating the retrieval of points based on their pixel indices.

Users can click around the centers of a dynamic objects in the intensity image and apply a region growing method to capture all points belonging to the dynamic object. The dynamic object is displayed in green, see figure A.2. The labeled information is stored in the specified format by clicking "Save Current Frame". As figure A.3 presents, the first column is the timestamp, and the rest columns are the pixel position of dynamic points in the intensity image.

When labeling dynamic objects, users can check previous and subsequent frames by clicking the "Previous" and "Next" buttons to identify dynamic points.

Sometimes, if the labeled result is not accurate, users can click the "Delete Current Frame" button to delete the labeled information and label it again. By clicking "Next," users can label the next frame.

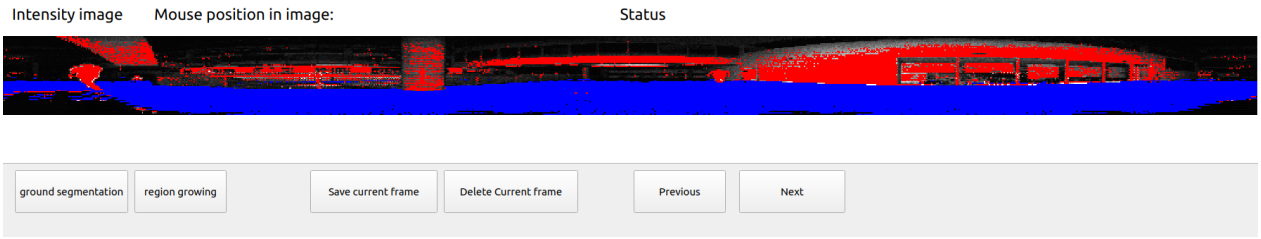


Figure A.1 Annotation tool for labeling dynamic objects in the intensity image. This tool allows users to label dynamic objects within an intensity image in a streaming style and store the labeled information in a specified format.

Graphical User Interface (GUI) Overview

The GUI of the annotation tool is illustrated in Figure A.1. Upon loading a frame of the point cloud from the dataset, the tool first converts the point cloud into an intensity image, as

detailed in Equation (5.1). To assist users in visual tracking, the tool segments feature points of foreground objects and highlights them in red within the intensity image, following the procedures outlined in Equations (5.2) through (5.4). Ground plane information is extracted from the point cloud and displayed in blue, providing a clear distinction between the ground and other objects. Figure A.1 exemplifies how this visual assistance enhances user experience by making dynamic objects easily identifiable, even when they are far from the robot.

Labeling Process

Users can annotate dynamic objects by clicking near their centers in the intensity image. The tool employs a region-growing algorithm to capture all points associated with the dynamic object, which is then displayed in green, as shown in Figure A.2. This labeled information is saved in a specified format by clicking the `Save Current Frame` button. As depicted in Figure A.3, the first column of the saved file represents the timestamp, while the subsequent columns contain the pixel positions of the dynamic points in the intensity image.

Navigation and Correction

To ensure accuracy, users can navigate through previous and subsequent frames using the `Previous` and `Next` buttons, allowing for a thorough review and identification of dynamic points. If the labeling is inaccurate, users can delete the labeled information by clicking the `Delete Current Frame` button and re-label the frame. The `Next` button facilitates the transition to the next frame for continuous labeling.

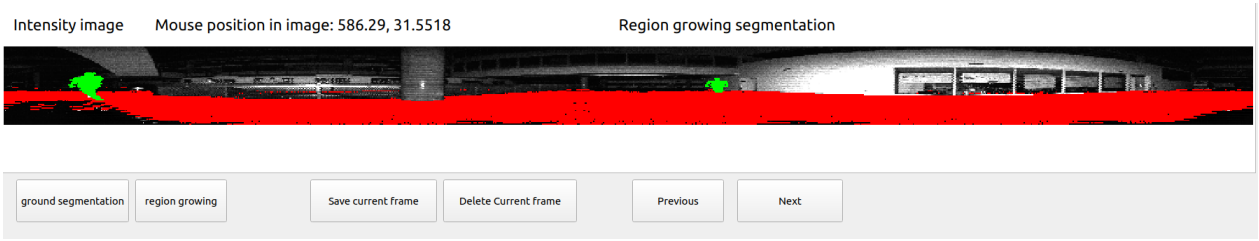


Figure A.2 The users can identify dynamic objects by browsing through the previous and next few frames of the intensity image. By clicking the center of the dynamic objects, seed points will be assigned to those objects. Clicking the "Region Growing" button will gather all points belonging to this dynamic object. The labeled objects will be displayed in green.

```
891.100434 27282 27283 27284 27285 27286 28307 28308 28309 28310 29332 29333 29334 29335 29336 29337 29338 30353
891.201017 27282 27283 27284 27285 27286 28306 28307 28308 28309 29332 29333 29334 29335 29336 29337 29338 30353
891.301514 27281 27282 27283 27284 27285 28305 28306 28307 28308 28309 29331 29332 29333 29334 29335 29336 29337
891.401844 27281 27282 27283 27284 28304 28305 28306 28307 28308 29331 29332 29333 29334 29335 29336 29337 30353
891.501877 29332 29333 29334 29335 29336 30353 30354 30355 30356 30357 30358 30359 30360 31377 31378 31379 31380
891.601909 27279 27280 27281 27282 27283 28302 28303 28304 28305 28306 28307 29330 29331 29332 29333 29334 29335
891.701779 29329 29330 29331 29332 29333 29334 30351 30352 30353 30354 30355 30356 30357 30358 31376 31377 31378
891.801584 27275 27276 27277 27278 27279 28300 28301 28302 28303 28304 29326 29327 29328 29329 29330 29331 29332
891.901336 27275 27276 27277 27278 27279 28299 28300 28301 28302 28303 29326 29327 29328 29329 29330 29331 30348
892.000954 27275 27276 27277 27278 28299 28300 28301 28302 29325 29326 29327 29328 29329 29330 29331 30347 30348
892.200386 29323 29324 29325 29326 29327 29328 30345 30346 30347 30348 30349 30350 30351 30352 31369 31370 31371
892.300186 29321 29322 29323 29324 29325 29326 29327 30344 30345 30346 30347 30348 30349 30350 30351 31369 31370
892.400063 29319 29320 29321 29322 29323 29324 29325 29326 30342 30343 30344 30345 30346 30347 30348 30349 30350
892.500020 29318 29319 29320 29321 29322 29323 29324 29325 30341 30342 30343 30344 30345 30346 30347 30348 30349
892.600035 29317 29318 29319 29320 29321 29322 29323 30339 30340 30341 30342 30343 30344 30345 30346 30347 30348
892.700097 29316 29317 29318 29319 29320 29321 30338 30339 30340 30341 30342 30343 30344 30345 30346 31361 31362
892.800245 29315 29316 29317 29318 30335 30336 30337 30338 30339 30340 30341 30342 30343 30344 31359 31360 31361
```

Figure A.3 Parts of the labeled information in the file. The first column represents the timestamp of the current frame, while the remaining columns contain the pixel indices of the dynamic objects in the intensity images.

Enhanced Functionality

Our annotation tool is designed to streamline the labeling process and improve the efficiency and accuracy of dynamic object identification in intensity images. By providing intuitive visual aids and robust navigation options, the tool ensures that users can label dynamic objects precisely, thereby enhancing the quality of the dataset.

Example Workflow

1. **Loading and preprocessing**: Load the dataset through the bag file path specified in the configuration file, and convert the point cloud to an intensity image.

2. **Feature Segmentation**: Segment and highlight foreground feature points in red and ground plane points in blue.

3. **Dynamic Object Annotation**: Click around dynamic objects to label them using the region-growing method, with the dynamic object displayed in green.

4. **Saving and Navigation**: Save the labeled data and navigate through frames using the provided buttons, correcting any inaccuracies as needed.

By integrating these features, our annotation tool provides a comprehensive solution for efficiently labeling dynamic objects in 3D point clouds, supporting both research and practical applications in autonomous navigation and robotics.

APPENDIX B DETAILED ANALYSIS OF EXPERIMENTAL RESULTS

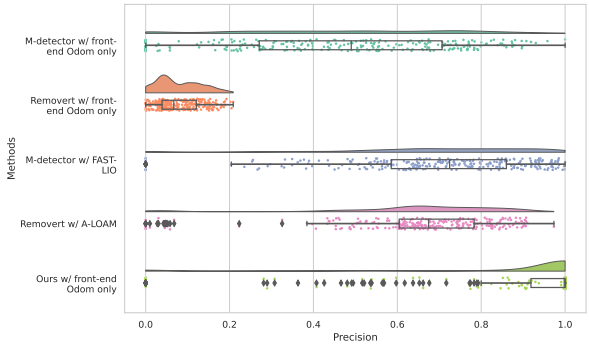
In this section, we will conduct a detailed analysis of the experimental results from paper in chapter 5. The experimental results shown in Figure B.1 provide a detailed comparison of the precision performance of different dynamic object detection methods across four sequences. The precision results across four sequences demonstrate the superiority and consistency of our method with front-end odometry. In Sequence 1, our method achieves the highest and most stable precision, significantly outperforming other methods such as M-detector with FAST-LIO and Removert with A-LOAM, which, while performing well, show greater variability. Removert and M-detector with front-end odometry exhibit lower precision and higher variability, indicating less reliability in dynamic object detection.

For Sequence 2, the precision results reinforce the findings from Sequence 1. Our method maintains the highest precision with minimal variability, demonstrating its robustness across different sequences. The performance of M-detector with FAST-LIO, although high in precision, shows notable fluctuations. The performance of Removert with A-LOAM has similar precision with M-detector with FAST-LIO, but lower variability. Meanwhile, Removert with front-end odometry and M-detector with front-end odometry continue to perform poorly, suggesting a consistent trend of lower precision and higher variability.

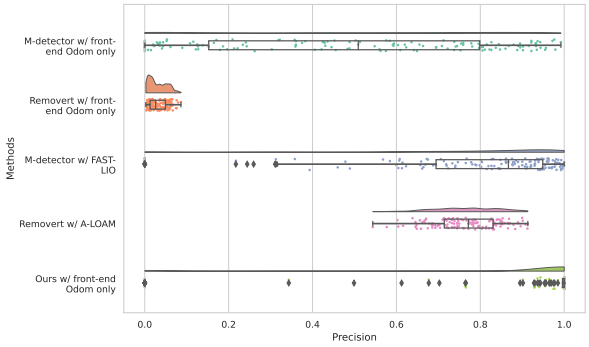
Sequence 3 precision results highlight the consistency and robustness of our method, which again achieves the highest precision throughout the sequence. M-detector with FAST-LIO remains a strong performer but with less stability compared to our method. The lower precision and higher variability observed in M-detector with front-end odometry, mirror the trends seen in previous sequences, underlining its limitations in dynamic object detection. Removert with A-LOAM in this sequence shows better performance compared to the earlier sequences, but still lags behind our method. Removert with front-end odometry continues to show the least precision, consistent with the earlier sequences, highlighting the challenges in maintaining accuracy with drift odometry.

The precision results in Sequence 4 further validate the effectiveness of our method, which leads with the highest precision and the least variability. M-detector with FAST-LIO shows competitive performance but with occasional dips in precision. Removert and M-detector with front-end odometry continue to show the least precision, consistent with the earlier sequences, highlighting the robustness and reliability of our approach.

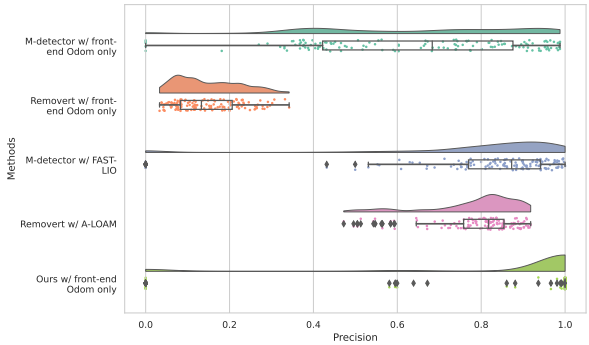
The IoU results in Figure B.2 across the sequences reinforce the trends observed in the



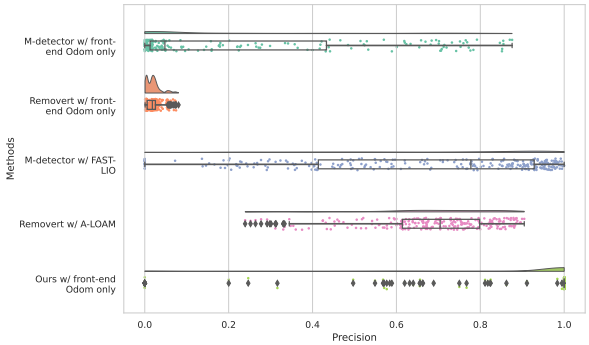
(a) *Precision results in Sequence 1*: This subfigure shows the precision results for the first sequence. The precision of each method is plotted, demonstrating the variability and performance across the sequence. Our method with front-end odometry achieves the highest and most stable precision throughout the sequence, significantly outperforming the other methods. M-detector with FAST-LIO also shows good performance but with more variability.



(b) *Precision results in Sequence 2*: This subfigure displays the precision results for the second sequence. Similar to Sequence 1, our method with front-end odometry maintains the highest precision with minimal variability. M-detector with FAST-LIO again shows high precision but with notable fluctuations. Removert and M-detector with front-end odometry continue to perform poorly, indicating a consistent trend across different sequences.

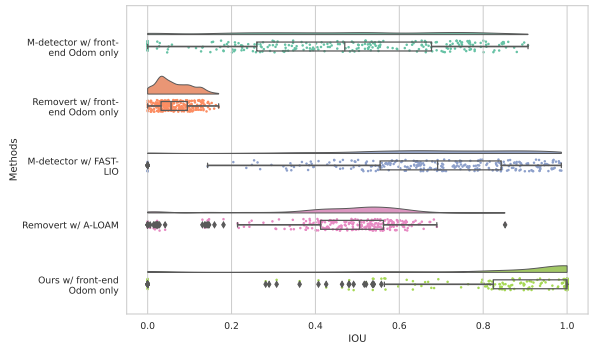


(c) *Precision results in Sequence 3*: This subfigure presents the precision results for the third sequence. It highlights how each method performed in terms of precision throughout the sequence. Our method consistently achieves the highest precision, reinforcing its robustness and reliability. The performance of M-detector with FAST-LIO remains high but less stable compared to our method. Removert with A-LOAM has slightly lower precision to M-detector with FAST-LIO, but has more stable performance. Removert with front-end odometry exhibit similar lower precision trends as observed in previous sequences.

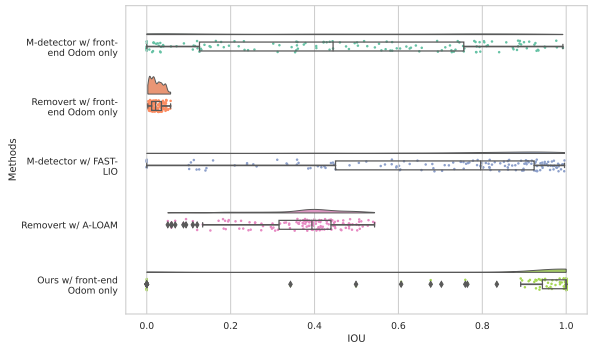


(d) *Precision results in Sequence 4*: This subfigure illustrates the precision results for the fourth sequence. It offers insights into the precision performance of each method in this sequence. Our method with front-end odometry once again leads with the highest precision and lowest variability. M-detector with FAST-LIO shows competitive precision but with occasional dips. Removert with A-LOAM still has similar performance with M-detector with FAST-LIO, but has more stable performance. Removert and M-detector with front-end odometry show the least precision, consistent with the patterns seen in the earlier sequences.

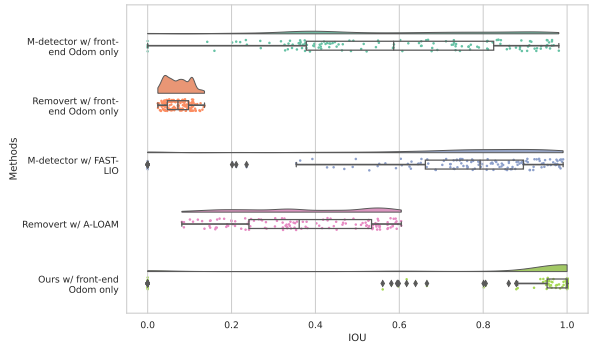
Figure B.1 **Precision results across four sequences.** This figure presents the precision results for different dynamic object detection methods across four test sequences. The methods compared are: M-detector with front-end Odom, Removert with front-end Odom, M-detector with FAST-LIO, Removert with A-LOAM, and our method with front-end Odom.



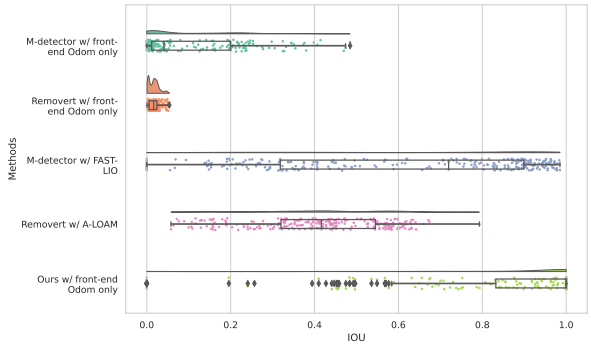
(a) *IoU results in Sequence 1*: This subfigure presents the IoU results for the first sequence. Our method with front-end odometry achieves the highest IoU, indicating its effectiveness in accurately detecting dynamic objects. M-detector with FAST-LIO shows good IoU but with more variability. Removert with A-LOAM and M-detector with front-end odometry display lower IoU and higher variability, suggesting challenges in maintaining consistency.



(b) *IoU results in Sequence 2*: This subfigure shows the IoU results for the second sequence. Similar to the precision results, our method with front-end odometry achieves the highest IoU with minimal variability. M-detector with FAST-LIO also performs well but shows notable fluctuations. M-detector with front-end odometry continue to show lower IoU and higher variability. Removert has lower variability, but also has lower IoU.



(c) *IoU results in Sequence 3*: This subfigure presents the IoU results for the third sequence. Our method with front-end odometry consistently achieves the highest IoU, reinforcing its robustness. M-detector with FAST-LIO remains competitive but less stable. Removert with A-LOAM and M-detector with front-end odometry show similar lower IoU trends as observed previously.



(d) *IoU results in Sequence 4*: This subfigure illustrates the IoU results for the fourth sequence. Our method with front-end odometry once again leads with the highest IoU and lowest variability. M-detector with FAST-LIO shows competitive IoU but with occasional dips. Removert and M-detector with front-end odometry show the lowest IoU, consistent with the patterns seen in earlier sequences.

Figure B.2 **IoU results across four sequences**. This figure presents the IoU results for different dynamic object detection methods across four test sequences. The methods compared are: M-detector with front-end odometry, Removert with front-end odometry, M-detector with FAST-LIO, Removert with A-LOAM, and our method with front-end odometry.

precision analysis. In Sequence 1, our method with front-end odometry achieves the highest IoU, demonstrating its effectiveness in accurately detecting dynamic objects. M-detector with FAST-LIO shows good IoU but with greater variability. Removert with A-LOAM and M-detector with front-end odometry display lower IoU and higher variability, indicating challenges in maintaining consistency. The Removert with front-end odometry shows the least IoU among all methods.

In Sequence 2, our method maintains the highest IoU with minimal variability, similar to the precision results. M-detector with FAST-LIO performs well but with notable fluctuations. The lower IoU and higher variability with M-detector with front-end odometry and the least IoU in Removert with front-end odometry, further confirm their limitations in dynamic object detection when they equipped lower precision odometry.

The IoU results for Sequence 3 continue to show our method's robustness, consistently achieving the highest IoU. M-detector with FAST-LIO remains competitive but less stable compared to our method. The trends of lower IoU and higher variability in Removert with A-LOAM and M-detector with front-end odometry, are consistent with previous observations.

For Sequence 4, our method once again leads with the highest IoU and the least variability. M-detector with FAST-LIO shows competitive IoU but with occasional dips. Removert and M-detector with front-end odometry continue to display the lowest IoU, aligning with the patterns seen in earlier sequences.

The recall results in Figure B.3 similarly highlight the robustness and effectiveness of our method. In Sequence 1, the recall performance of our method is notably similar to the M-detector with front-end odometry method. Among the methods, M-detector with FAST-LIO achieves the best recall results, demonstrating the highest recall values and the least variability. Despite this, our method remains highly competitive, exhibiting substantial recall performance that outperforms the other methods. Both our method and the M-detector with front-end odometry demonstrate superior recall results compared to the Removert methods, specifically Removert with front-end odometry and Removert with A-LOAM, which consistently show lower recall values and higher variability.

Sequence 2 recall results demonstrate that our method has the highest recall with minimal variability. M-detector with FAST-LIO shows high recall but notable fluctuations. M-detector with front-end odometry also exhibits competitive recall, although with more variability. The lower recall and higher variability in Removert with A-LOAM and Removert with front-end odometry, indicate consistent poor performance across different sequences.

In Sequence 3, our method consistently achieves the highest recall, reinforcing its robustness.

M-detector remains high in recall but less stable. However, in this sequence, the M-detector with front-end odometry has better performance than M-detector with FAST-LIO. Removert with A-LOAM and M-detector with front-end odometry show similar lower recall trends as previously observed.

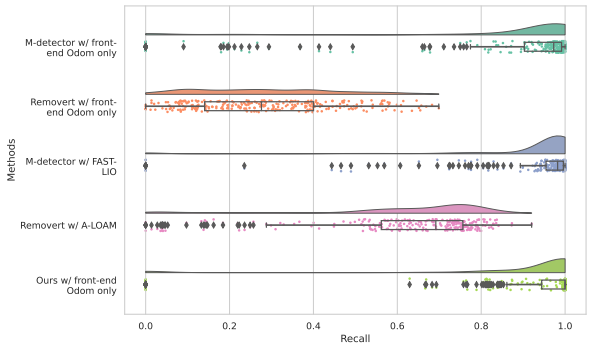
The recall results in Sequence 4 further illustrate the effectiveness of our method, leading with the highest recall and lowest variability. M-detector with FAST-LIO shows competitive recall but with occasional dips. However, in this sequence M-detector with front-end odometry's recall performance is not as good as previous sequences. It has similir lower recall as Removert with front-end odometry . Removert with A-LOAM preformes better than before but still has lower recall compared to M-detector with FAST-LIO and our method.

The F1 Score results in Figure B.4 across the sequences provide a comprehensive view of our method's performance. In Sequence 1, our method with front-end odometry achieves the highest and most stable F1 Score, significantly outperforming other methods. M-detector with FAST-LIO shows good F1 Score but with more variability, while Removert with A-LOAM and M-detector with front-end odometry show lower F1 Score and M-detector with front-end odometry shows higher variability. Removert with A-LOAM shows better F1 score compared with the performance in other sequences. The Removert with front-end odometry has the lowest F1 Score among all methods.

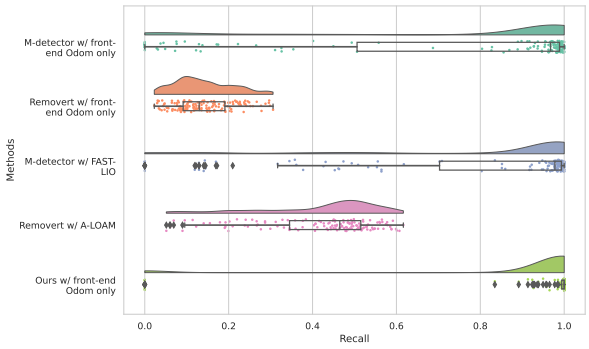
For Sequence 2, our method maintains the highest F1 Score with minimal variability, similar to the precision and recall results. M-detector with FAST-LIO shows high F1 Score but notable fluctuations. The lower F1 Score in Removert with front-end odometry and the higher variability of M-detector with front-end odometry, further confirm their consistent poor performance during noisy odometry.

Sequence 3 F1 Score results reinforce our method's robustness, consistently achieving the highest F1 Score. M-detector with FAST-LIO remains competitive but less stable. The M-detector with front-end odometry has better F1 score than Removert with A-LOAM, but they all have higher variability. The lowest F1 Score in Removert with front-end odometry is consistent with previous observations.

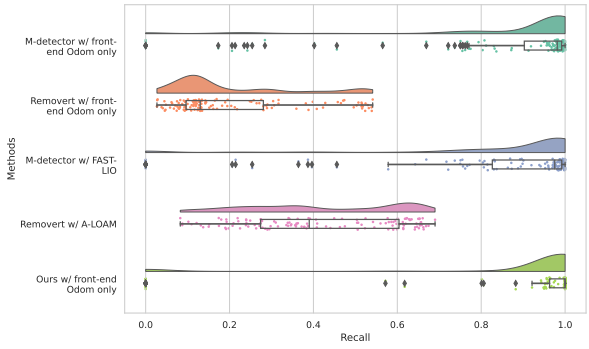
In Sequence 4, our method once again leads with the highest F1 Score and lowest variability. M-detector with FAST-LIO shows competitive F1 Score but with even worse stability compared the performance in the previous sequences. Removert with front-end odometry continue to display the lowest F1 Score, aligning with the patterns seen in earlier sequences. M-detector with front-end odometry has similar F1 Score as Removert with front-end odometry, but with higher variability.



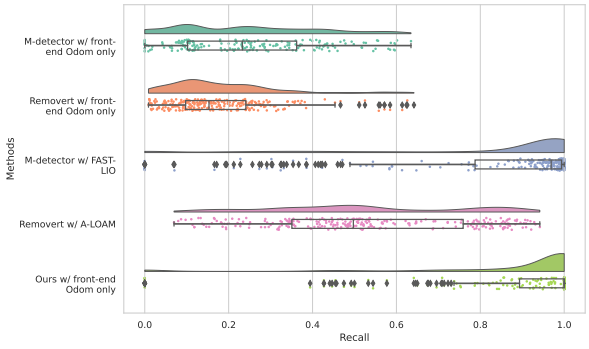
(a) *Recall results in Sequence 1*: This subfigure shows the recall results for the first sequence. M-detector with FAST-LIO achieves the best recall with the least variability. Our method with front-end odometry remains competitive, outperforming Removert methods, which show lower recall and higher variability.



(b) *Recall results in Sequence 2*: This subfigure displays the recall results for the second sequence. Our method achieves the highest recall with minimal variability. M-detector with FAST-LIO shows high recall with fluctuations. Removert methods continue to perform poorly with lower recall.

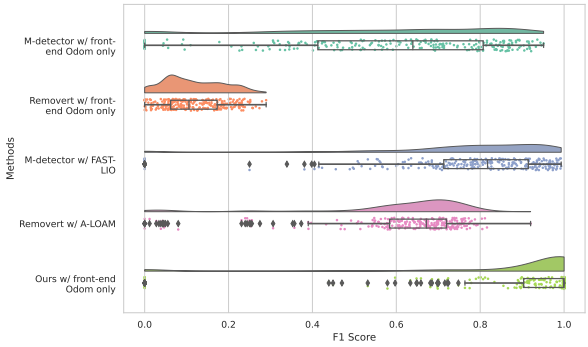


(c) *Recall results in Sequence 3*: This subfigure presents the recall results for the third sequence. Our method consistently achieves the highest recall. M-detector with front-end odometry performs better than M-detector with FAST-LIO. Removert methods show similar lower recall trends.

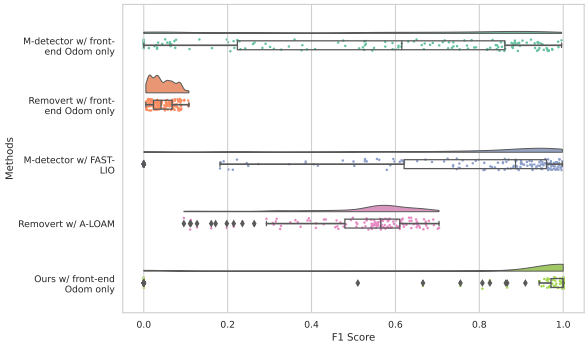


(d) *Recall results in Sequence 4*: This subfigure illustrates the recall results for the fourth sequence. Our method leads with the highest recall and lowest variability. M-detector with FAST-LIO shows competitive recall but with occasional dips. M-detector with front-end odometry and Removert methods perform worse compared to previous sequences.

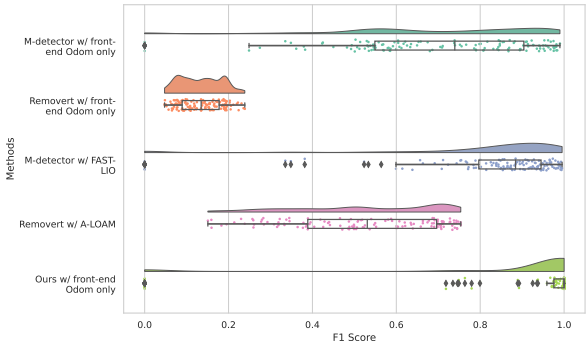
Figure B.3 **Recall results across four sequences**. This figure presents the recall results for different dynamic object detection methods across four test sequences. The methods compared are: M-detector with front-end odometry, Removert with front-end odometry, M-detector with FAST-LIO, Removert with A-LOAM, and our method with front-end odometry.



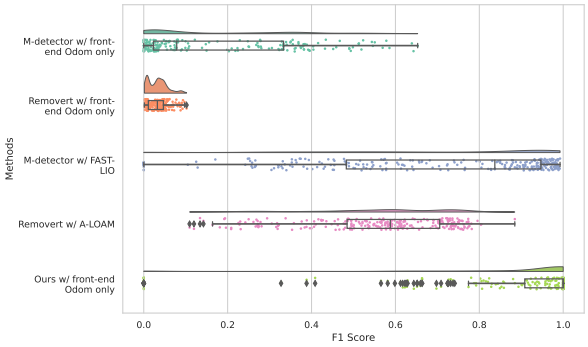
(a) *F1 Score results in Sequence 1*: This subfigure demonstrates our method with front-end odometry achieving the highest and most consistent F1 Score, significantly outperforming the other methods. M-detector with FAST-LIO shows competitive scores but with notable variability. Removert with A-LOAM and M-detector with front-end odometry exhibit lower F1 Scores and increased variability. Removert with A-LOAM shows slight better performance compared with other sequences.



(b) *F1 Score results in Sequence 2*: Our method continues to maintain the highest F1 Score with minimal variability, echoing the precision and recall outcomes. M-detector with FAST-LIO, although high, exhibits significant fluctuations. Both Removert and M-detector with front-end odometry confirm their weaker performance under noisy conditions by showing lower scores or higher variability.



(c) *F1 Score results in Sequence 3*: This subfigure highlights the robustness of our method, consistently achieving the highest F1 Score. M-detector with FAST-LIO remains competitive but but less stable. M-detector with front-end odometry outperforms Removert with A-LOAM in this sequence, though both continue to demonstrate high variability and lower scores.



(d) *F1 Score results in Sequence 4*: Our method leads with the highest F1 Score and lowest variability, confirming its consistent performance across sequences. M-detector with FAST-LIO remains competitive but with decreased stability. Removert with front-end odometry consistently shows the lowest F1 Scores, consistent with earlier patterns, while M-detector with front-end odometry displays similar scores but with increased variability.

Figure B.4 F1 Score results across four sequences. This figure presents the F1 Score results for different dynamic object detection methods across four test sequences. The methods compared are: M-detector with front-end odometry, Removert with front-end odometry, M-detector with FAST-LIO, Removert with A-LOAM, and our method with front-end odometry.

Overall, the detailed analysis of experimental results across precision, IoU, recall, and F1 score consistently highlights the superiority and robustness of our method with front-end odometry. The consistent high performance and low variability of our method across different sequences underscore its reliability and effectiveness in dynamic object detection.

Additionally, we performed statistical analysis using the Wilcoxon Signed-Rank Test [136] to evaluate the significance of the differences between the methods. We selected the Wilcoxon Signed-Rank Test because some of the data for precision, IoU, recall, and F1 score do not follow the normal distribution, and some data present outliers or extreme values, such as 0 or 1. Therefore, the Wilcoxon Signed-Rank Test is well-suited for this analysis. The results demonstrate that our method with front-end odometry significantly outperforms the other methods in terms of precision, IoU, recall, and F1 score, with all p-values < 0.05. This statistical analysis further confirms the superior performance of our method in dynamic object detection.