UNIVERSITÄT BONN

Masterarbeit
zur Erlangung des Grades
Master of Science (M. Sc.)
der Landwirtschaftlichen Fakultät
der Rheinischen Friedrich-Wilhelms-Universität Bonn
Institut für Geodäsie und Geoinformation

3D LIDAR Place Recognition for Autonomous Driving Exploiting Local Density Maps

von

Saurabh Gupta

aus
Mumbai, India

Bonn 2023

Supervisor:
Prof. Dr. Cyrill Stachniss, University of Bonn, Germany

Second Supervisor:
Dr. Tiziano Guadagnino, University of Bonn, Germany

Statement of Authorship

I hereby certify that this master thesis has been composed by myself. I have not made use of the work of others or presented it here unless it is otherwise acknowledged in the text. All references and verbatim extracts have been quoted, and all sources of information have been specifically acknowledged.

Place, Date	(Signature)

Acknowledgments

I would like to show my gratitude towards all the people who have helped me directly and indirectly throughout the master's and this thesis. Firstly, I would like to thank Prof. Dr. Cyrill Stachniss for giving me the opportunity to work on this topic. I thank my family for supporting me remotely and, most importantly, for facilitating my stay abroad for studying.

I would like to dedicate a special thanks to my mentors, Ignacio Vizzo and Tiziano Guadagnino. Their presence made this journey more valuable to me, and I learned a lot of technical as well as non-technical things from them. The efforts they have taken to support me and my work and the confidence they have instilled in me are beyond words. I wish every student gets rewarded with such mentors. I would also like to thank Benedikt Mersch for his valuable input throughout the thesis and Inkscape lessons, as well as for translating the abstract of this thesis to German. This thesis would not have been possible without these mentors. I would also like to extend my warm thanks to Lasse Peters, Federico Magistri, Nicky Zimmerman, Thomas Laebe, and Jens Behley for their valuable guidance throughout the masters. I express my gratitude towards Meher and Meltem for proofreading this thesis. Also, I thank the janitor and the housemaster for maintaining the working spaces in the lab.

I also acknowledge the love and support I have received from my friends Amit, Awadhut, Ashish, Jay, Jheel, Rushikesh, Pranav, and Sachin. They have been my pillars of strength throughout the last eight years, especially during the years spent in isolation during the pandemic, and no words would be enough to acknowledge them enough. The new friends I earned in Bonn have also been very important in this journey, helping me to overcome the pandemic blues and enjoy life in Bonn. They all have always stood by me, especially in testing times, and cheered me up. So, a big thanks to Richa, Sai, Preetham, Sumanth, Alonso, Amanda, Kim, and Saurav. A special thanks to Dhagash for partnering with me in several assignments and projects throughout the masters, for the numerous days and nights spent doing robotics stuff, and, more importantly, for being a great friend.

Last but not least, I would like to thank my therapist Dipl.-Psych. Afua Adusei-Poku-Erken for helping me out in my most difficult times during the last six months. This thesis would have been impossible without her weekly sessions.

Abstract

An autonomous robot should always have an accurate estimate of its location within the environment and a descriptive map of the surroundings, enabling safe and accurate navigation, and facilitating interactions with the environment when necessary. Furthermore, these robots are often required to be deployed in previously unknown environments, as in the case of autonomous vehicles. Such a task is called simultaneous localization and mapping (SLAM) in robotics parlance. Such a system, though, has to deal with a variety of noise present in the sensed environment, the sensor suite, and the algorithms used to estimate these quantities, resulting in inaccurate estimates of the robot's location and, subsequently, of the map generated using the sensed information from these locations.

The ability to detect revisited places or loop closures in SLAM is crucial to overcoming this problem, as the geometrical information obtained from a loop closure can override the inaccurate pose estimates. Mainly, it allows for correcting the drifting pose estimates from a sensor odometry pipeline. Researchers have relied on camera images to achieve this task, using features and semantic information extracted from these images. However, the lighting conditions affect the camera images, limiting their use in adverse weather, or even at night. As a result, the research on place recognition shifted focus on 3D LiDAR sensors due to their superior performance in adverse lighting conditions, with the added benefit of having 3D information of the surroundings.

In this research work, we address the problem of effectively detecting loop closures in LiDAR SLAM systems in various environments with longer lengths of sequences and agnostic of the scanning pattern of the sensor. While many approaches for loop closures using 3D LiDAR sensors rely on individual scans, we propose the usage of local maps generated from locally consistent odometry estimates. Several recent approaches compute the maximum elevation map on a bird eye view projection of point clouds to compute feature descriptors. In contrast, we use the density image computed from a bird eye view representation, which is robust to viewpoint changes.

The utilization of dense local maps allows us to reduce the complexity of

features describing these maps, as well as the size of the database required to store these features over a long sequence. This yields a real-time application of our approach for a typical robotic 3D LiDAR sensor. We perform extensive experiments to evaluate our approach against other state-of-the-art approaches and show the benefits of our proposed approach. Finally, we also provide an open-source implementation of our approach, with an efficient implementation of the core library written in C++, and easy-to-use Python bindings API.

Zusammenfassung

Autonome Roboter benötigen für eine sichere und genaue Navigation sowie Interaktion mit ihrer Umgebung detailliertes Wissen über ihre Pose und eine Karte ihrer Umgebung. Ein Beispiel sind autonome Autos, die normalerweise in einer unbekannten Umgebung eingesetzt werden. Diese Aufgabe wird in der Robotik allgemein als simultane Lokalisierung und Kartierung (SLAM) bezeichnet. Ein solches System muss jedoch mit einer Vielzahl von Ungenauigkeiten in der erfassten Umgebung, den Sensoren und den Algorithmen umgehen, was zu ungenauen Schätzungen der Roboterpose und anschließend der Karte führt, die anhand der ermittelten Posen erstellt wird.

Einen bereits besuchten Ort wiederzuerkennen und einen Schleifenschluss zu bestimmen ist ein entscheidender Schritt in SLAM Algorithmen, um zusätzliche Bedingungen für die potenziell driftenden Posenschätzungen zu finden und diese Ungenauigkeiten zu korrigieren. In der Vergangenheit haben Forscher für diese Aufgabe Kamerabilder benutzt und aus diesen zum Beispiel semantische Merkmale extrahiert. Die Kamerabilder werden jedoch von den Lichtverhältnissen negativ beeinflusst, was ihre Verwendung bei ungünstigen Wetterbedingungen oder bei Nacht einschränkt. Infolgedessen verlagerte sich der Schwerpunkt der Forschung im Bereich der Schleifenschlusserkennung auf 3D LiDAR Sensoren, da diese auch bei ungünstigen Lichtverhältnissen zuverlässig Daten aufnehmen und den zusätzlichen Vorteil bieten, dass sie 3D Informationen über die Umgebung liefern.

In dieser Forschungsarbeit befassen wir uns mit dem Problem der effektiven Erkennung von Schleifenschlüssen in LiDAR SLAM Systemen in diversen Umgebungen und mit unterschiedlichen Aufnahmedauern unabhängig vom Scanmuster des Sensors. Während viele Ansätze zur Erkennung von Schleifenschlüssen mit 3D LiDAR Sensoren einzelne Scans nutzen, fokussieren wir uns auf die Verwendung lokaler Karten, die mithilfe der geschätzten Posen generiert werden. Neuere Ansätze projizieren die Punktwolke aus der Vogelperspektive auf ein 2D Gitter und nutzen die maximale Höhe der Punkte in jedem Voxel als Deskriptoren zur Beschreibung der lokalen Umgebung. Im Gegensatz dazu bestimmen wir die Dichte der Punkte in jedem Voxel, was robuster gegenüber Änderungen des

Blickwinkels ist.

Mit der Verwendung dieser 2D Repräsentation können wir die Komplexität der Deskriptoren, sowie die Größe der Datenbank, die zur Speicherung dieser Merkmale über eine lange Sequenz erforderlich ist, reduzieren. Dies ermöglicht die Echtzeitanwendung unseres Ansatzes auf Robotern mit 3D LiDAR Sensor. Wir führen außerdem umfangreiche Experimente durch, um unseren Ansatz im Vergleich zu existieren Methoden zu evaluieren und zeigen damit die Vorteile unserer Arbeit auf. Schließlich stellen wir eine effiziente Implementierung mit einer C++ Bibliothek als auch einer einfach zu verwendenden Python API als Open-Source Projekt öffentlich zur Verfügung.

Task Description

Place recognition plays a critical role in simultaneous localization and mapping (SLAM) systems by facilitating the detection of revisited locations within the map. This capability allows for effective drift correction through the integration of constraints into the underlying factor graph, thereby enhancing the accuracy of odometry estimates and the resulting map. Robust place recognition becomes furthermore relevant in outdoor SLAM due to the accumulated drift over longer trajectories of sequential 3D pose estimation. Despite the well-established body of research on place recognition employing camera images [31], the challenges presented by outdoor environments, including variations in illumination and seasonal changes, render image-based recognition algorithms susceptible to failure. Conversely, LiDAR sensors demonstrate resilience to such perceptual variations, motivating a closer examination of place recognition techniques utilizing 3D LiDAR data – a domain that currently lacks an abundance of literature compared to its image-based counterparts. Furthermore, contemporary odometry and SLAM systems [61] [64] [9] commonly rely on LiDAR sensors as their primary means of pose estimation.

The existing methods for 3D LiDAR-based place recognition typically operate on individual laser scans, either computing the overlap between two non-consecutive scans, or extracting and matching features from them. The overlap-based methods [7] are limited by the extent of drift in the odometry as they make use of scan matching algorithms. The feature-based approaches involve computing local point features and generating descriptors based on their local neighborhood [21]. This enables obtaining an initial estimation of loop closure by matching and aligning these point features. Another approach is to assign an egocentric global descriptor to each scan [19]. This speeds up the matching process, but does not provide an initial guess on the relative pose. Such descriptors are dependent on the density of the scans around the local features or the sensor's frame of reference, and given the projective model of LiDAR, this density is influenced by the location of the sensor in the environment. Hence, these methods perform best when the revisit is within a certain threshold on the true 3D pose. Additionally, some recent works heavily rely on the scanning pattern of conventional scanners [26].

The student should develop an approach orthogonal to the scan-based approaches, wherein he aggregates multiple scans within fixed displacement intervals and create local maps for place recognition. The main assumption is that a reliable odometry pipeline [61] maintains local consistency, with significant drift occurring only over longer sequences. This aggregation should provide more information to compute robust features and make our approach independent of the range-sensor scan pattern and density. Consequently, it should be able to effectively handle traditional LiDARs, depth cameras, as well as the unconventional scan pattern of Livox LiDARs [29]. Moreover, the approach should yield a 2D relative pose between the matched local maps which reduces the computational cost associated with 3D pose estimation.

The plan is to utilize the local density maps obtained from the recorded data sequence. These density maps are a 2D histogram of the orthographic projection of the 3D local maps (point clouds) onto the ground plane. By treating these density maps as images, one can take advantage of image-based features, descriptors, and matching algorithms to identify similar density maps. However, there is a distinction in our approach compared to pinhole camera-based image alignment, as one can estimate a 2D rigid body transform instead of a homography since our density map is an orthographic projection. This estimated 2D transform can then be utilized as initial guess for a 3D map-to-map alignment process.

The objective of the thesis is to develop such a system and implement a proficient code base in C++ capable of handling a series of 3D scans from outdoor environments, regardless of the type of range-sensor used. The code should identify which sequence indices correspond to the same physical locations within the scanned environment and calculate the relative 3D pose for these associations. Additionally, a comparative analysis should be conducted to assess the effectiveness of our approach compared to existing state-of-the-art methods for place recognition and loop closure. A suitable evaluation metric aligned with the established standards for this topic has to be employed for a fair assessment.

Point Of Departure:

- Literature review to gain insight over the past contributions to the field of place recognition in the context of SLAM [19] [21] [26] [31].
- Study feature detection and descriptor generation for images, as well as algorithms to efficiently store and match these feature descriptors.
- Brainstorm over a fair evaluation paradigm to compare map-based approaches with existing scan-based approaches.

Contents

Chapter 1

Introduction

The modern world has seen a rapid rise in the deployment of robots to automate tasks such as driving vehicles, household chores such as cleaning, as shown in Figure 1.1, as well as smart logistics and inventory management. All these tasks require a system capable of autonomous mobility. In order to achieve such autonomy, a mobile robot must have an awareness of its surroundings as well as its own location and orientation within it. These two tasks, i.e., mapping, and localization, are interdependent, with the availability of accurate pose estimates enabling an accurate map representation and vice versa. This is traditionally referred to as the simultaneous localization and mapping (SLAM) problem. This interdependence between localization and mapping makes it non-trivial to achieve high accuracy on either of the tasks, mainly due to the various sources of noise present in such a system.

In this thesis, we focus on detecting revisited places (loop closures), which, as discussed later, play a crucial role in SLAM systems to compensate for the uncertainties during estimation. Furthermore, we particularly consider range sensors such as light detection and ranging (LiDAR) sensors as our primary sensing modality, mainly because of their superior performance in a variety of environmental conditions and the relatively accurate and dense 3D range measurements of modern LiDARs.

1.1 Motivation

LiDAR SLAM systems primarily rely on the iterative closest point (ICP) [2, 3, 8, 48, 49] point cloud alignment strategy between consecutive 3D laser scans to keep a track of the ego-motion of the robot. More recent approaches also perform 3D registration of individual scans against a map of the environment generated using previously aligned scans [10, 36, 61]. However, the sequential odometry estimates often suffer from drift, which is even more emphasized over longer traversals in

Figure 1.1: Robots capable of performing automated tasks: On the left is an autonomous vehicle equipped with sensors such as LiDAR, RADAR, and cameras. Courtesy: *Waymo*. On the right is a robotic vacuum cleaner that can autonomously navigate household spaces using a camera mounted at the front. Courtesy: *iRobot*.

unstructured environments. This drift can occur due to the inherent noise in the robot motion and sensor data, the presence of dynamics in the environment, as well as non-trivial data association problems in ICP. As a result of such drift, when the robot revisits a known place, its own belief may indicate a completely different location.

Figure 1.2 (top) shows the drift in the KISS-ICP [61] odometry estimates over a 6.1 km long MulRan [25] KAIST03 sequence, wherein the recording platform loops over the same location multiple times as shown by the reference trajectory in red. The drift is also reflected in the map generated using these pose estimates as seen in Figure 1.2 (bottom). This affects downstream tasks that might require interaction with the environment. To overcome this problem, the detection of re-visited locations, also called loop closures, is of utmost importance for any SLAM pipeline. It works as a feedback loop in the SLAM pipeline, verifying whether the current pose estimate conforms with the observation of the surroundings obtained from the sensor. Typically, a pose-graph [17] backend is used in SLAM pipelines to incorporate the information obtained from the detected loop closures.

Detecting previously seen places requires crafting an as-unique-as-possible description of the sensed environment. At the same time, such a description should be invariant to changes in the 3D viewpoint, scan pattern of LiDARs, and dynamic objects in the sensed environment. Furthermore, to incorporate loop closures in a SLAM system, this description should preferably capture the geometry of the scene to enable relative pose estimation. Traditional methods [15, 19, 21, 26, 28] rely on computing such a description for each LiDAR scan. Subsequently, a matching algorithm is used to compare these descriptions and detect loop closures. The bird eye view (BEV) projection of point clouds is one popular approach [26, 28, 28] towards detecting loop closures, providing a compact 2D representation for faster feature detection and feature matching.

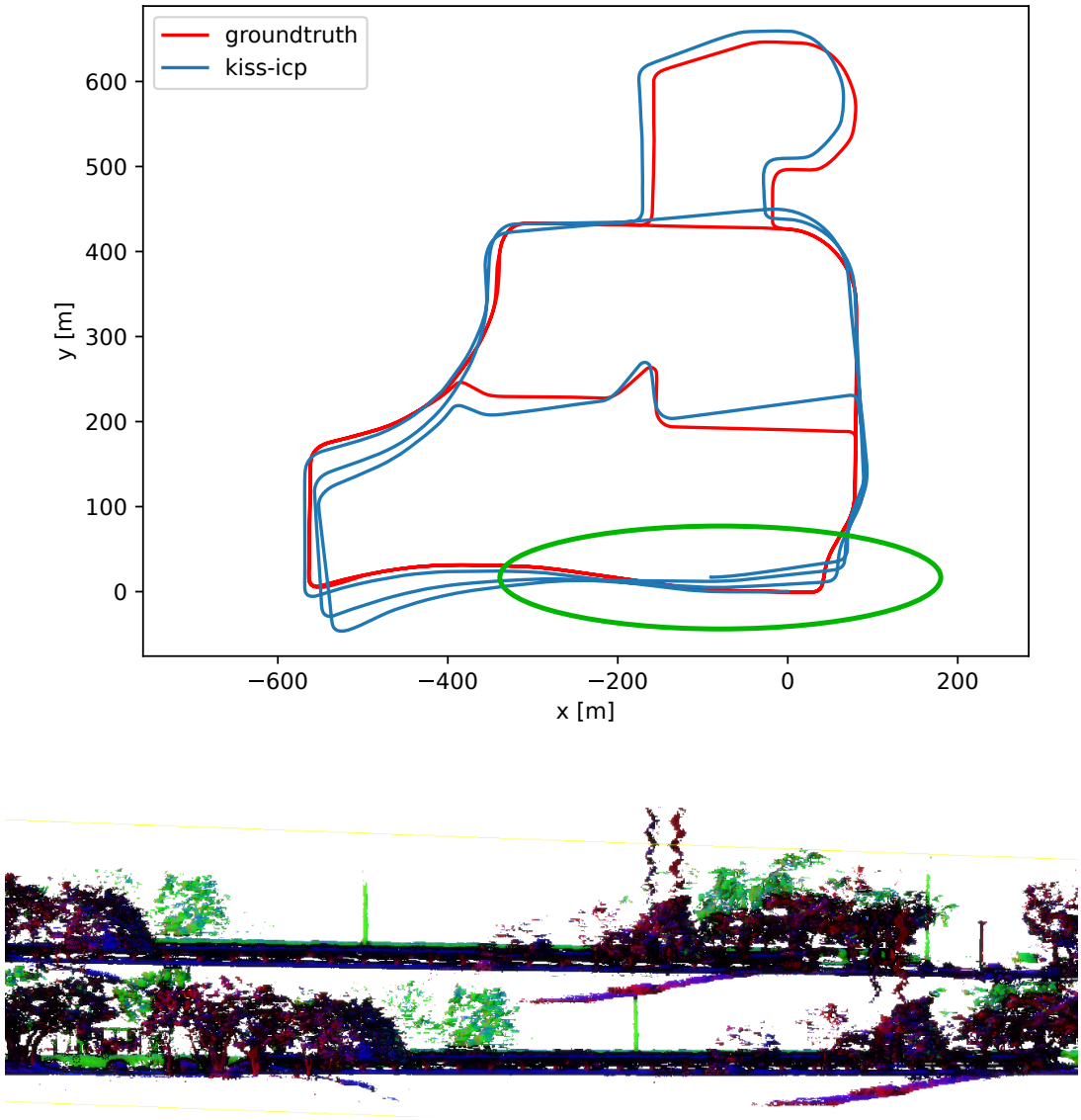


Figure 1.2: The drift in LiDAR odometry [61] estimates for long traversals can be clearly seen in the plot on the top of the figure. On the bottom, a map of the scene generated from these odometry estimates, using the VDBFusion [60] mapping pipeline, is shown. It can clearly be seen that the same physical location (highlighted by a green ellipse) has been mapped twice at different estimated locations, resulting in a distorted map.

1.2 Goal and Main Contributions

The main contribution of this thesis is a simple yet effective approach for detecting revisited places (loop closures) in online LiDAR SLAM systems. Local maps generated using locally consistent sensor odometry estimates form the core element of our approach. This is one of the first methods to make use of local maps for the purpose of place recognition. A majority of methods for place recognition in LiDAR SLAM systems detect closures at the level of individual LiDAR scans. However, the aggregation of information in the form of a map allows us to detect

loop closures more effectively, independent of the LiDAR sensor's density or scan pattern.

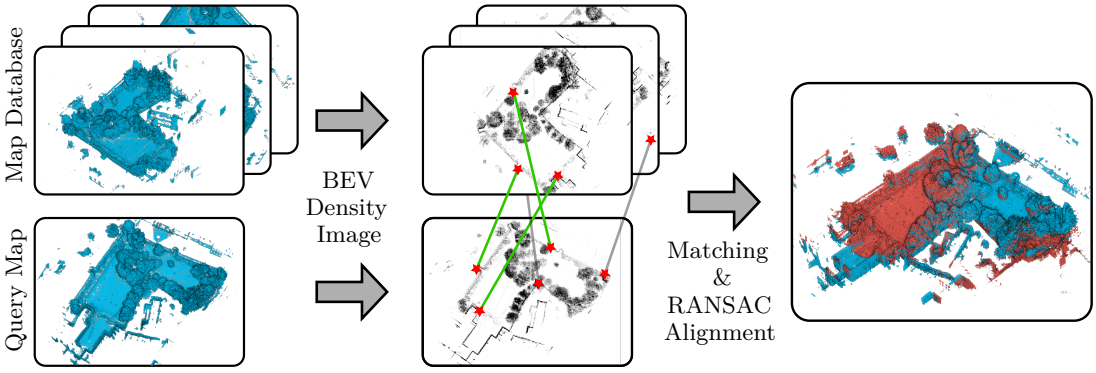


Figure 1.3: A database over previously seen local maps is queried for matching a new local map (left). We perform feature matching on features computed on the bird eye view (BEV) density images of these maps to detect closures (center). Finally, a RANSAC-based 2D rigid body alignment is used to obtain the 2D pose estimate between loop closed local maps (right).

Furthermore, we map the three-dimensional local map to a two-dimensional density image through an orthographic bird eye view (BEV) projection. The choice of preserving the density of points from the local map, instead of the widely adopted approach of preserving the maximum elevation, makes our method more robust to larger variations in the viewpoint during a potential revisit. These two design choices subsequently allow us to use a simple feature detection and matching pipeline on the density images to effectively detect loop closures. Finally, we also provide a validation step to perform a geometric verification of detected loop closures and provide a complete 2D rigid body transform between the loop-closed local maps. This initial estimate aids the pose-graph optimizer in the backend of a SLAM system. An overview of our approach can be seen in Figure 1.3. In summary:

- Our approach can effectively detect loop closures between two temporally separated local maps in a variety of environments.
- It can provide a 2D rigid body transform between the detected loop closures.
- It is agnostic of the sensor scan pattern by exploiting local maps.

1.3 Overview of the Thesis

This thesis aims to provide an efficient approach for detecting revisited places in the context of 3D LiDAR SLAM. We provide in Chapter 2 a brief summary of the existing techniques and state-of-the-art methods for place recognition with data from a 3D LiDAR sensor. Chapter 3 provides a background on the basic

techniques from robotics used in the development of this thesis. A detailed explanation of our approach and the implementation details can be found in Chapter 4. A comprehensive quantitative as well as qualitative analysis of our pipeline on a variety of datasets and comparisons against other state-of-the-art approaches can be found in Chapter 5, thereby supporting all the aforementioned claims.

Furthermore, we open-source our pipeline, which has been implemented efficiently in C++. We also support the project with Python bindings of the C++ library, thereby making it easy to use for other researchers and allowing further improvement on the proposed idea. This work is under review at the International Conference on Robotics and Automation (2024).

Chapter 2

Related Work

The task of loop closing in the domain of 3D laser ranging sensors has been a topic of discussion in many recent works. These works have mainly been driven by the significant improvements in 3D LiDAR sensor technology over the past few years, with increasing density and accuracy of the scanners. Zhang et al. [65] provides a detailed analysis of the existing literature on place recognition with 3D LiDARs in outdoor environments. In this section, we provide a brief analysis of some key approaches for place recognition in the context of SLAM, as well as highlight the key differences of our approach as compared to them.

The simplest approaches for place recognition in SLAM rely solely on the individual scans and their pose estimates. The pose estimates are directly used to compute a pose-similarity [42, 50] between two non-consecutive scans within a pre-defined search radius to detect loop closures. S4-SLAM [66] and Mendes et al. [35] compute an overlap ratio between the point clouds to check if they are recorded from the same location. OverlapNet [7] computes the overlap between the range-image representation of two scans using a deep neural network. To validate the loop closures thereby obtained, a point cloud registration step is often performed [11, 16, 51]. These approaches, however, are limited by the amount of drift present in the odometry estimates, requiring a search radius proportional to the magnitude of the drift.

Place recognition using 3D LiDARs has been significantly influenced by the existing methods for place recognition using camera images. Many approaches [4,21, 45, 46, 62] propose computing 3D point features from the individual point clouds. They discretize the neighborhood around these feature points into a geometrical grid and compute a local descriptor based on the height [4], density [14, 21, 57], or distance and angle [44, 45] of the points contained within such a neighborhood. PointNet [37] and PointNet++ [38] pioneered a learning-based method using multi-layer perceptron networks to detect features and shapes from a point

cloud. The main benefit of such methods is that they preserve the 3D information, allowing for the estimation of a complete 3D relative pose between the detected revisits. However, these methods are partially affected by the sparsity of a typical LiDAR point cloud, are not viewpoint-invariant, and are significantly affected by occlusions in the data.

The normal distribution transform (NDT) [34] proposed a single global descriptor for a point cloud to overcome the requirement of storing and matching multiple local features per scan, often done by training a bag-of-words model or creating a tree-like database. They superimpose a voxel grid on the scans and approximate a normal distribution over the set of points in each cell of this voxel grid. A histogram over these normal distributions becomes the global description of the point cloud. Matching such a global descriptor is straightforward, requiring less computational effort. However, it is challenging to recover relative pose estimates between loop-closed scans using global descriptors. ContourContext [20], too, proposes using normal distributions computed over point clouds to define a global descriptor. However, instead of using a voxel grid discretization, they compute clusters (contours) of points at discrete elevation intervals, see Figure 2.1. Then, a normal distribution is approximated over each such cluster. A constellation of these clusters is used to define a global descriptor for matching, also encoding the relative pose between the clusters, which in turn allows us to estimate the relative pose between matched scans. The use of discretized height intervals, though, makes this approach sensitive to larger viewpoint changes during revisits. This is because the projective model of traditional LiDAR sensors does not preserve the height of the points scanned on a surface when viewed from a different location.

In contrast to the above approaches which operate directly on the 3D data, a lot of recent approaches [6, 19, 24, 26, 28, 32, 33, 40, 54, 63, 67] propose a low dimensional mapping of the 3D data in order to reduce the computational cost

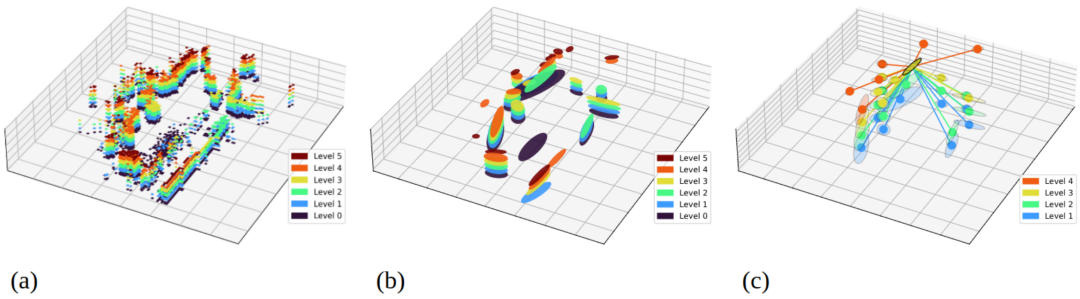
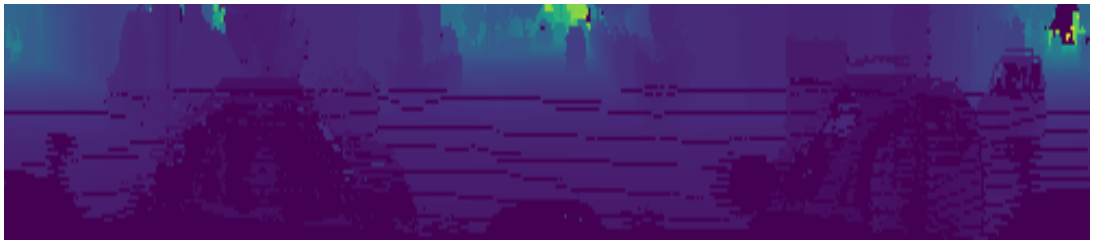


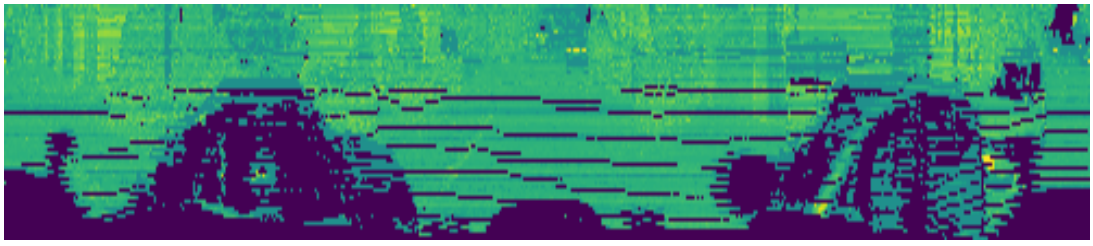
Figure 2.1: An overview of the global descriptor computation in ContourContext. a) discretization of the point cloud at different elevation levels, b) clustering the points at each level and defining a 2D Gaussian distribution over these clusters, and c) constellation of clusters. Courtesy: *ContourContext* [20].

involved in processing 3D information. Even though such a dimensionality reduction discards crucial geometric information about the scene, such methods often perform comparably or even better than their 3D counterparts, especially in outdoor scenes. The dimensionality reduction is usually done by projecting the point cloud onto a two-dimensional manifold while preserving point-wise information such as range, color, intensity, height, or density. Such a two-dimensional projection can be treated as an image, allowing for the use of the widely available literature on place recognition with camera images. Roehling et al. [40] propose a 1-dimensional histogram over the elevation of points in the scan as the description of the scene and use a simple Wasserstein distance between two histograms as the metric to detect revisited places. This descriptor, although very fast to compute and compare, suffers significantly from the loss of information along two dimensions and also does not provide any initial guess on the relative alignment between detected loop closures.

Among the two-dimensional projection-based methods, the spherical projection of point clouds into a range-image or an intensity-image is widely used, an example of which can be seen in Figure 2.2a and Figure 2.2b respectively. Steder et al. [54] pioneered the range-image spherical projection, computing 2D local feature descriptors on these images for place recognition. They extend this method [55] to use NARF features [56] along with a bag-of-words model for place recognition. Zhuang et al. [67] and Cao et al. [6] respectively use SURF [1] and



(a) Range Image



(b) Intensity Image

Figure 2.2: An example of range and intensity projections of a 3D point cloud. The horizontal axis corresponds to the azimuthal angle, and the vertical axis corresponds to the elevation angle of a scan from a typical LiDAR sensor with a spherical scan pattern. a) The spherical projection of a point cloud preserving the range of each scanned point. b) The spherical projection of a point cloud preserving the intensity of each scanned point.

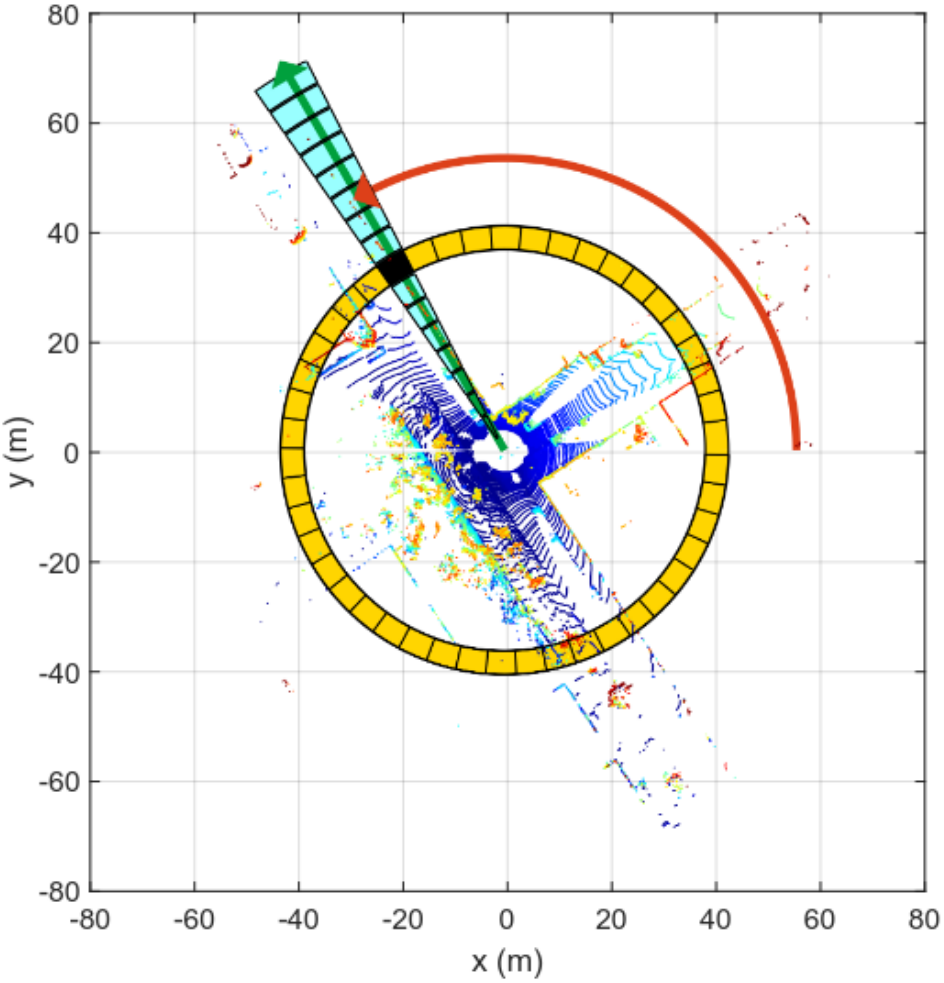
ORB [43] features computed from the range-images

Bird eye view (BEV) projection of point clouds onto a 2D plane is another common paradigm employed by several recent approaches [19,24,26,28,32,33,63]. BVMatch [32] proposes using a density-preserving BEV projection of point clouds on the local ground plane. They compute the maximum index map on a Log-Gabor filtered representation of these density images. Furthermore, it requires training a bag-of-words model for place recognition, making it unsuitable for use in online SLAM within unknown environments.

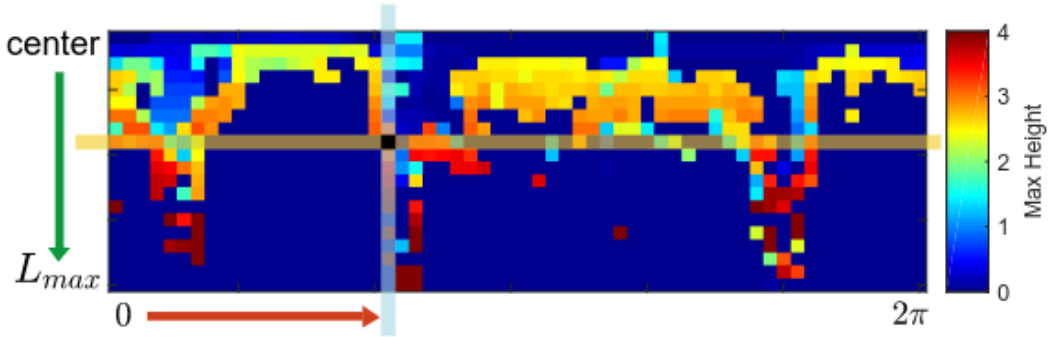
ScanContext [26] proposes a global descriptor over a polar elevation map (Scan/Polar Context) for each point cloud, with the polar grid centered at the local reference frame of the LiDAR, see Figure 2.3. The polar coordinates make the descriptor invariant to in-plane rotations, with a rotation corresponding to column shifts of the ScanContext descriptor. However, there is no such relation between translational shifts and the descriptor. They try to overcome this issue by manually shifting the reference frame of scans by the expected translation during revisits and storing the descriptors computed from these root-shifted scans as well. Even though the global descriptor allows for faster place recognition, it provides only a yaw angle estimate between the detected closures. Furthermore, the polar grid works best only for typical LiDAR sensors with a 360° azimuthal field-of-view.

ScanContext++ [24] tries to improve upon the drawbacks of ScanContext [26] by computing Cartesian elevation maps (Cart Context) along with the polar elevation maps. They further suggest augmenting both Cart Context and Polar Context by manually rotating and shifting the laser scans, respectively, the magnitude of these augmentations being decided by the expected variations in viewpoint during revisits. The choice of preserving the maximum elevation of points during the BEV projection makes both these approaches sensitive to viewpoint changes during revisit, and also, the implicit assumption of having a traditional projective pattern of LiDAR scans makes them unsuitable for solid-state LiDARs like the Livox Horizon scanner.

Another recent work by Yuan et al. [63] proposes using a set of geometric primitives, i.e., triangles with unique side lengths, as the description of a point cloud. In particular, they propose identifying large planar regions in a local map generated using a fixed number of consecutive scans and their odometry estimates. Then, for the voxels along the boundary of each such planar region, they project all the points in these voxels to the local plane they surround, preserving the maximum height of the points. Eventually, they identify local maxima from these projections to build the triangle descriptor. See Figure 2.4. This approach is also suitable for non-traditional LiDAR scanners due to the accumulation of scans into a local map.



(a) Bin division along azimuthal and radial directions



(b) Scan context

Figure 2.3: An overview of the computation of a ScanContext descriptor of a point cloud. They define a polar grid over the bird eye view projection of individual LiDAR scans, preserving the maximum height of the points contained in each cell of the polar grid. Courtesy: *ScanContext* [26].

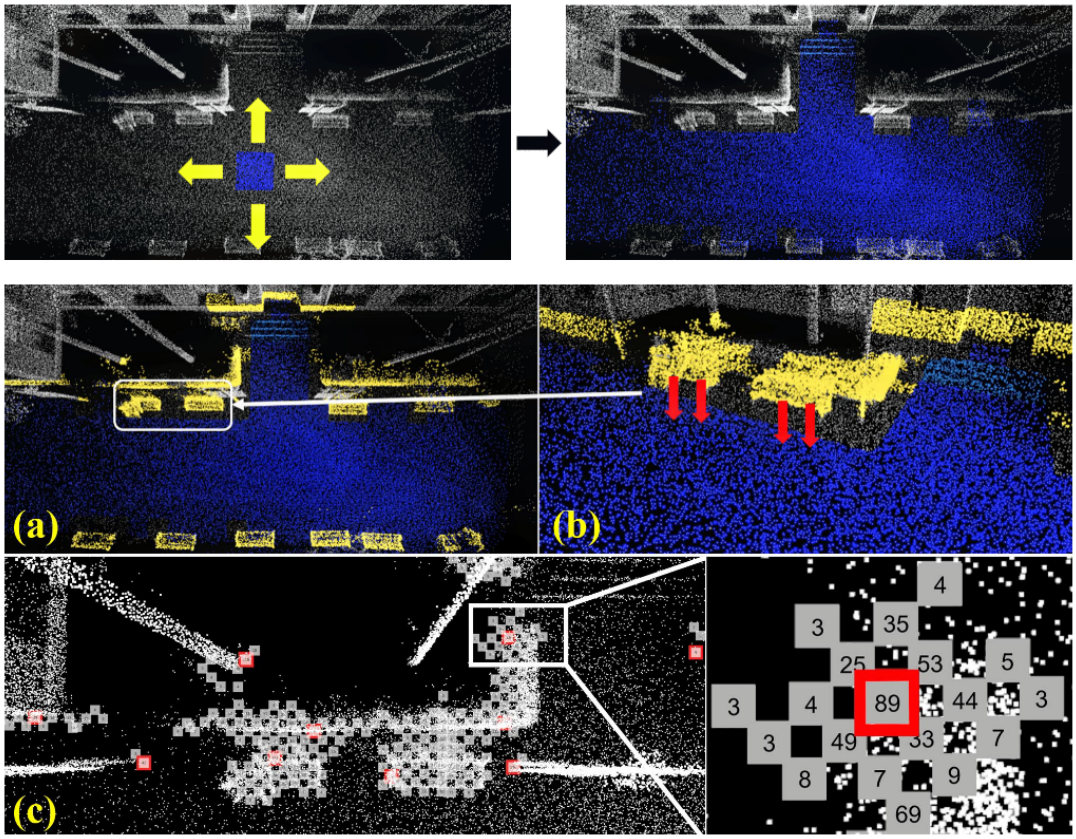


Figure 2.4: An overview of point cloud processing to compute stable triangle descriptors (STD). They use a plane-expanding process to detect planar regions in the point clouds (top). a) Then, voxels on the boundary of each such planar region are identified (in yellow). b) The points in these neighboring voxels are projected to the plane they neighbor, preserving the maximum elevation of points. c) Unique triangle-like descriptors are computed from the local maxima of these projections. Courtesy: *Stable triangle descriptors (STD)* [63].

Among the 2D projection-based approaches, BEV projections [24, 26, 28, 32, 33] seem to be preferred over the range image-based approaches [6, 54, 55, 67]. A key reason is that the range images lose depth information while projecting spherically. On the other hand, BEV projections preserve 2D geometry along the local XY plane, which is crucial in the case of most autonomous ground robots/vehicles due to their motion profile.

Our proposed method makes use of local maps as the primary representation, where we accumulate scans based on travel distance criterion rather than naively accumulating a fixed number of consecutive scans [10,28,63]. Such maps allow us the benefit of having a dense collection of points representing the environment, which show a stronger invariance towards viewpoint changes. Aggregation of multiple scans also makes our method robust to different sensor scan patterns and field-of-views. We further compute a BEV projection of these local maps, preserving the density of points in each vertical column, similar to BVMatch [32],

instead of the widely used elevation images. This also helps to increase robustness against viewpoint changes and noise in the sensing process. Utilizing existing feature descriptors from the computer vision community [43], we compute local features from these density images and store them in an efficient binary search tree [47] for place recognition. A final geometric validation step allows us to provide a complete 2D relative pose estimate between detected map-level closures, which aids the final 3D registration step involved in basically any SLAM pipeline.

Chapter 3

Basic Techniques

This chapter provides an introduction to the task of place recognition, loop closure evaluation, LiDAR odometry, image-based feature detection, and geometric alignment of 2D point-sets. Section 3.1 introduces the task of place recognition and its application in the context of SLAM. The following Section 3.2 briefly discusses the evaluation of loop closures in SLAM. Section 3.3 explains the iterative closest point (ICP) algorithm, which forms a core element of our pipeline to generate locally consistent maps. In Section 3.4, we provide the reader with an overview of the process of local feature detection and description in 2D images. Finally, we introduce the binary tree database used to store and match the features in Section 3.5 and explain the geometric verification step in Section 3.6.

3.1 Place Recognition in SLAM

The task of place recognition in robotics refers to a robot's awareness of its surroundings at any given location and the ability to differentiate between locations based on the observations it makes. It is the capability to identify previously visited places, as the name suggests. In the most basic form, place recognition can be achieved by having a precise localization of the robotic platform at all times. This way, the pose estimates themselves can suggest if a place is being revisited or not. However, such an accurate localization is non-trivial to achieve, requiring alternate approaches for place recognition. The most common approach towards place recognition is to utilize exteroceptive sensors mounted on the robotic platforms to gather distinctive information about the surroundings. RGB cameras, depth cameras, stereo-pair cameras, light detection and ranging sensors (LiDAR), and radio detection and ranging (RADAR) are a few examples of exteroceptive sensors.

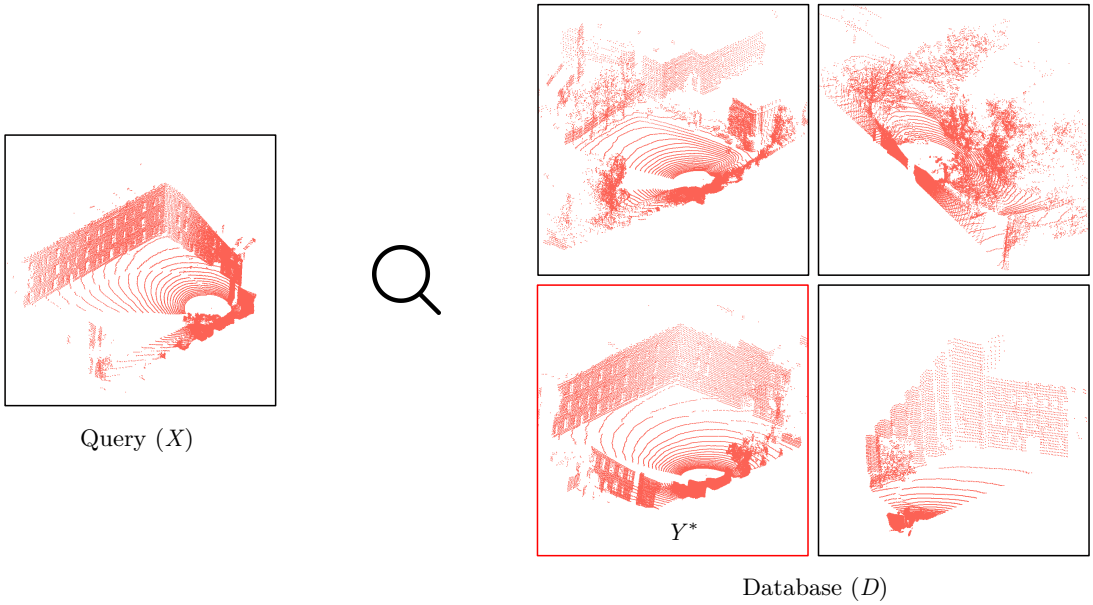


Figure 3.1: Place recognition is a task that involves creating a database (D) over the information collected from sensors (LiDAR in this case), and finding the best match (Y^*) for the information obtained from a new frame of sensor data (X).

The focus of this thesis is on LiDAR sensors as the primary sensing modality for place recognition. In its most basic form, place recognition with 3D LiDARs can be viewed as an optimization task to find the best matching LiDAR frame Y^* to a query frame X from within a database D of previously recorded frames, as formulated in Equation (3.1) and depicted in Figure 3.1.

$$Y^* = \underset{Y}{\operatorname{argmax}} \; p(X,\; Y \,|\, Y \in D), \tag{3.1}$$

where $p(X,\; Y \,|\, Y \in D)$ defines a similarity metric between two LiDAR frames.

Generally, the LiDAR data is processed a priori in order to remove noise and extract meaningful information from them. Additionally, in the context of loop closing for SLAM applications, the place recognition pipeline should be able to execute with real-time constraints, as well as preferably provide a relative pose estimate between the detected revisits to aid the pose-graph optimization. In our approach, we use the same fundamental concept as in Equation (3.1), but instead of operating on individual LiDAR point clouds, we operate on local maps generated from multiple such LiDAR scans.

3.2 Evaluation of Loop Closures

A fair evaluation of the proposed approach forms an essential part of scientific research. This section provides a summary of the evaluation criteria we use to compare and validate our work against other state-of-the-art approaches. A

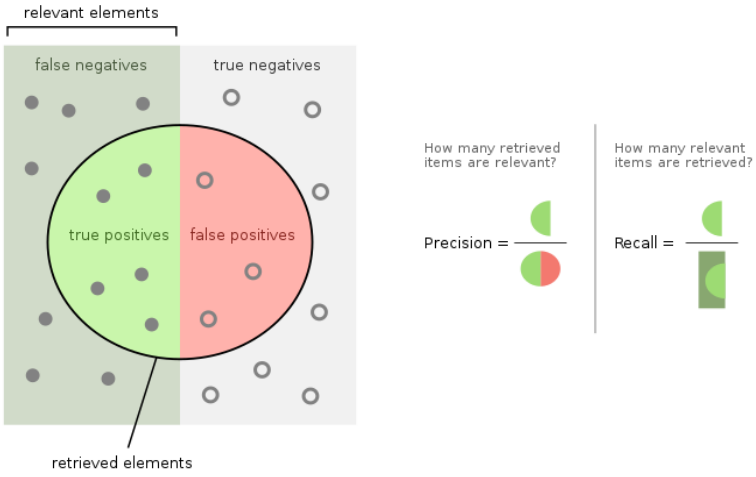


Figure 3.2: An illustration of precision-recall estimation. The small circles (filled/unfilled) correspond to all possible pairs of scans. The filled small circles correspond to the pairs from the same physical locations in reality. The bigger circle encompasses all the pairs of scans that an algorithm predicts to be from the same location. Courtesy: *Wikepedia*.

precision-recall curve is the most fundamental metric used to evaluate the results from a place recognition pipeline. This is also a widely used metric in a variety of classification tasks, especially in the domain of machine learning. Figure 3.2 shows an illustration explaining what precision and recall values quantify. The relevant elements could be seen as all the ground-truth pairs of sensor scans belonging to the same location, whereas the retrieved elements are all the pairs of sensor scans that our algorithm predicts as belonging to the same location. Given these sets of detected and predicted loop closures, we can easily deduce with the help of Figure 3.2 that the precision value quantifies the ratio of retrieved elements that are relevant to the task, and the recall value quantifies the percentage of relevant elements that were successfully retrieved. In other words, the precision value informs us how precise a certain algorithm is at the task of place recognition, penalizing any wrong predictions, whereas the recall value informs us how much of the ground-truth loop closures our algorithm can successfully predict, penalizing any failure to recognize similar places.

However, the precision and recall scores are not sufficient independently and always need to be evaluated together. This is because any given algorithm can achieve 100% recall by simply classifying every scan pair as a loop closure or achieve very high precision by providing only a single highly confident prediction with close to zero recall. As a result, the precision-recall curve is generally used to highlight the range of performance an algorithm offers through its primary configuration parameters. An example of a precision-recall curve comparison between two algorithms is shown in Figure 3.3. A better algorithm will always have the precision-recall curve reaching for the upper-right corner of the graph. The F1

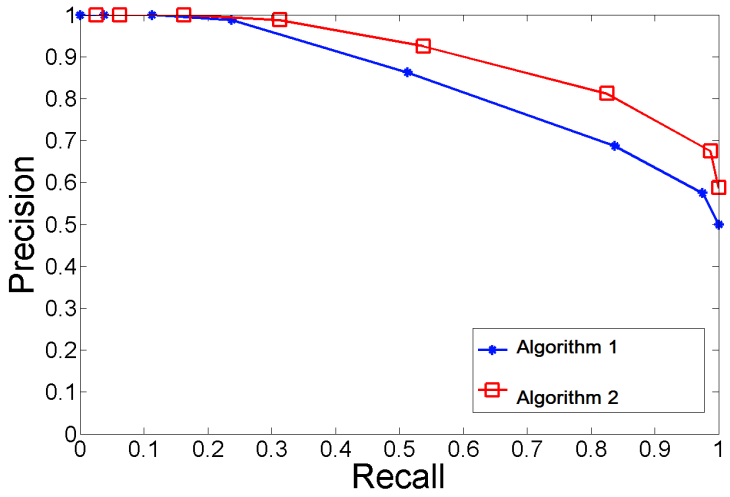


Figure 3.3: A comparison of precision-recall curves from two algorithms. These curves show that algorithm 2 (red) has an overall better performance as compared to algorithm 1 (blue). Algorithm 2 (red) has better or at least an at par precision value for the entire range of recall values. Courtesy: *https://www.datacamp.com/tutorial/precision-recall-curve-tutorial*.

score is another metric that is commonly used to evaluate place recognition and loop closure detection approaches. It is the harmonic mean of precision and recall values corresponding to a given configuration of the pipeline; see Equation (3.2). A high F1 score also corresponds to being closer to the upper-right corner of the precision-recall curve.

$$\mathrm{F1} = 2\,\frac{\mathrm{precision} \times \mathrm{recall}}{\mathrm{precision} + \mathrm{recall}} \tag{3.2}$$

An ideal algorithm for place recognition and loop closure should have both 100 % precision and 100 % recall, resulting in an F1 score of 1.0. However, in the context of SLAM, higher precision is preferred at the cost of low recall because a few but confident loop closure detections are sufficient to reduce drift significantly in the pose-graph optimization. In this thesis, we compare the highest achievable F1 scores by each of the approaches as well as their corresponding precision and recall values. We also use precision-recall curves to exhibit the range of performance offered by each of these approaches.

3.3 3D LiDAR Odometry

The focus of this thesis is to perform place recognition using 3D LiDAR data, with the aim of detecting loop closures for SLAM applications. As a result, we safely assume the presence of a LiDAR odometry pipeline computing pose estimates of the sequential LiDAR sensor frames. Such an odometry pipeline traditionally works on the principle of point cloud alignment using the iterative

closest point (ICP) [3, 8] algorithm. In this section, we discuss in brief the basic working principles of a generic ICP point cloud alignment algorithm. We also mention some details about KISS-ICP [61], the odometry pipeline we use to generate the local maps required for our approach.

Figure 3.4 provides an illustration of point cloud alignment in its most basic form, with known data associations between two point clouds. Given two point clouds $\{\boldsymbol{x}_n\} \in \mathbb{R}^3$ and $\{\boldsymbol{y}_n\} \in \mathbb{R}^3$, and a set $\mathcal{C}$ of point-wise associations between the two point clouds, we need to find a rigid body transform $R \in \mathbb{SO}(3)$ and $\boldsymbol{t} \in \mathbb{R}^3$ satisfying the optimization task as formulated in Equation (3.3). This rigid body transform is estimated using the Kabsch-Umeyama [23, 58] algorithm or with a non-linear least-squares alignment approach.

$$\begin{aligned} &\min \sum_{n \in \mathcal{C}} \|\boldsymbol{y}_n - \bar{\boldsymbol{x}}_n\|^2 \\ &\text{where, } \bar{\boldsymbol{x}}_n = R\boldsymbol{x}_n + \boldsymbol{t} \end{aligned} \tag{3.3}$$

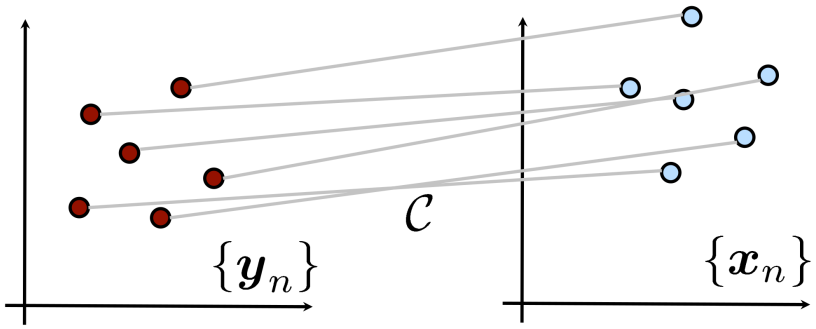


Figure 3.4: Point cloud alignment with known data association. Given two point clouds $\{\boldsymbol{x}_n\}$ and $\{\boldsymbol{y}_n\}$, and a point-wise association $\mathcal{C}$ between the two point clouds, we need to estimate a relative rigid body transform such that the associated points are completely aligned. Courtesy: *Lecture slides by Prof. Dr. Cyrill Stachniss.*

Even though the point cloud alignment task, as explained above, looks straightforward, the assumption about the availability of the correspondences set $\mathcal{C}$ does not hold true in reality. This data association task forms a core element of any ICP algorithm. An ICP algorithm can be basically divided into an iterative process consisting of the following two steps: i) performing data association between two point clouds, and ii) computing the rigid body alignment (Equation (3.3)) using the correspondence set so obtained. This iterative process is terminated upon convergence decided by a variety of criteria. The data association step has a huge impact on the convergence and speed of the iterative process. A myriad of different data association strategies have been suggested by researchers to improve the accuracy and efficiency of point cloud alignment for LiDAR odometry. These strategies range from simply computing the closest point using a kd-tree to extracting features from the point clouds and aligning matching features. Moreover,

due to the increasing size of point clouds obtained from modern-day LiDARs, a lot of methods employ a pre-processing step, such as uniform sampling, feature-based sampling, and random sampling, to reduce the computational cost of their algorithms.

Among the closest point-based data association strategies, the point-to-point and point-to-plane metrics are widely adopted, along with a least-squares-error minimization formulation solved using the Gauss-Newton algorithm. The point-to-point metric simply computes the nearest point in the target point cloud for each point in the source point cloud and minimizes the distances between these associations to compute the rigid body alignment. On the other hand, the point-to-plane metric takes it one step further, computing the distance projection of the point-to-point distance along the unit-normal at the target point, as illustrated in Figure 3.5.

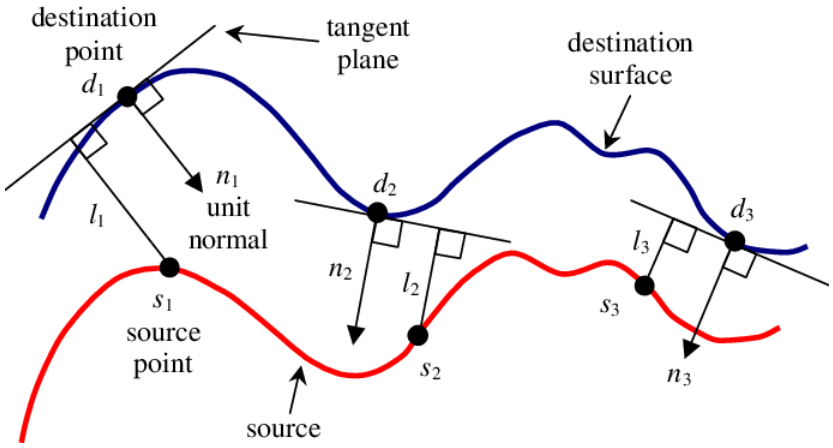


Figure 3.5: An illustration of the point-to-plane data association strategy between two point clouds (red and blue). Courtesy: *Lecture slides by Prof. Dr. Cyrill Stachniss.*

In this thesis, we use the KISS-ICP pipeline proposed by Vizzo et al. [61]. It is a LiDAR odometry pipeline based on point-to-point ICP that can provide accurate pose estimates in a variety of environments and robot motion profiles. The basic assumption of KISS-ICP is that the sensing platform moves at a constant velocity between two consecutive scans. This assumption usually holds as, in the typical LiDAR sensor frame rate (10Hz or 20Hz), the change in velocity is minimal even when the robot is subject to high acceleration. Thanks to this assumption, the odometry pipeline modules can be heavily simplified, leading to a system that requires little to no parameter tuning in most scenarios. Furthermore, KISS-ICP maintains a moving local map representation of the environment in the form of a voxel hash map, which provides a scan-to-map alignment that is more robust to sensor noise. This helps to improve the local consistency of the odometry estimates, which is crucial to our approach to detecting revisited places with local maps. Additionally, the authors of KISS-ICP provide an easy-to-install

and easy-to-use open-source implementation of their research, making it suitable for hassle-free use in our own pipeline.

3.4 Local Feature Descriptors in 2D Images

Our approach uses a 2D image-like representation of the local maps to compute locally distinctive features for place recognition. We make use of one of the existing techniques for extracting such features from natural images with certain modifications specific to our use case. This section provides an overview of the process of extracting features from images and assigning a unique descriptor to each such feature.

This task can be divided into two core components: i) keypoint extraction and ii) computing the descriptor for each keypoint. Several classical feature descriptors rely on the assumption that the keypoints are repeatable and distinctive. Also, for them to be suitable for matching across different frames, they have to be invariant to minor viewpoint changes, in-plane rotations, scale, and illumination changes. Corners in the 2D projection of 3D structures are often highly distinct points in images offering the aforementioned invariant properties. The search for such corners usually involves looking for points with intensity changes along two orthogonal directions in its local neighborhood. The structure matrix, as seen in Equation (3.4), is typically used to compute the variation in the intensity of the pixels surrounding a potential keypoint.

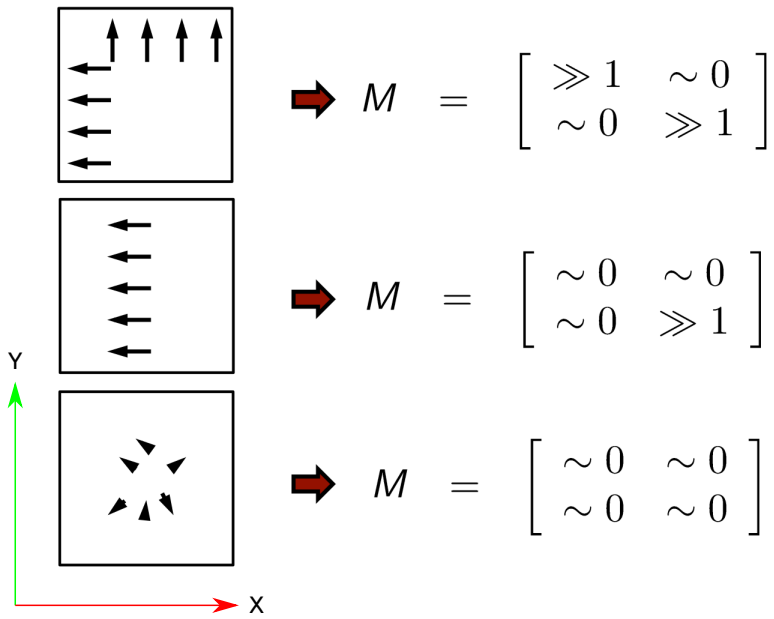


Figure 3.6: This figure shows typical structure matrices (M) for different patterns of intensity variation around a given pixel in an image. Courtesy: *Lecture slides by Prof. Dr. Cyrill Stachniss.*

$$M = \begin{bmatrix} \sum_W J_x^2 & \sum_W J_x J_y \\ \sum_W J_y J_x & \sum_W J_y^2 \end{bmatrix}, \tag{3.4}$$

where W represents the pixels in a local neighborhood, and J_x and J_y are the Jacobian of image intensities along the x and y directions respectively. Figure 3.6 shows approximate templates of a structure matrix for different patterns of intensity variation around a pixel in an image. We can detect corner-like points looking at the eigenvalues of the structure matrix, with a higher eigenvalue representing a larger variation in intensity along the direction of the corresponding eigenvector. The Foerstner corner detector [13], the Harris corner detector [18], and the Shi-Tomasi corner detector [52] are some famous corner detection algorithms based on the structure matrix formulation. Lowe [30] proposed using a difference of Gaussian representation to compute keypoints across different scales, accounting for the scale ambiguity inherent in perspective projection cameras. However, this is not relevant to our pipeline as the density images we use are generated by an orthographic projection of 3D point clouds and, therefore, have no scale variance of the keypoints.

These keypoints, although locally distinct, do not capture enough information by themselves for comparison and matching with other keypoints. As a result, we need to define descriptors summarizing the local structure around them, as seen in Figure 3.7. SIFT [30], SURF [1], BRIEF [5], and ORB [43] are some famous feature descriptors. In this thesis, we use the oriented FAST rotated BRIEF (ORB) [43] features for the density images generated by the bird-eye-view projection of the local maps. The main reason for making this choice is the binary domain of the ORB descriptor, which is easy to compute and compare. Algorithm 1 shows the algorithm used to compute the binary BRIEF descriptors from an image patch. The authors suggest five different sampling strategies for the pixel pairs from a given patch as seen in Figure 3.8.

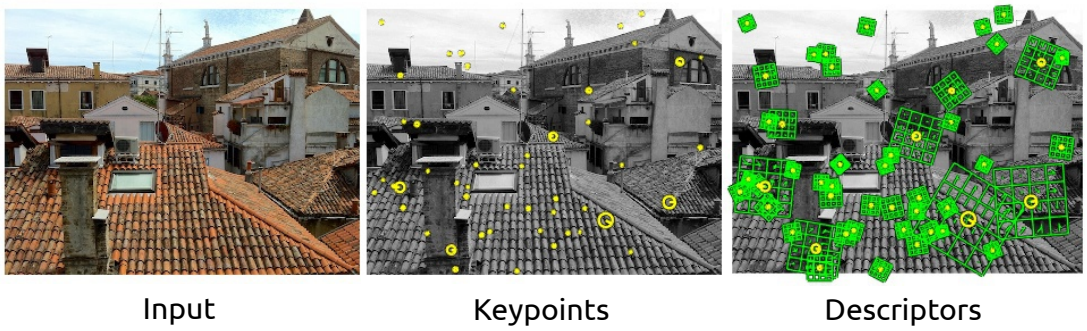


Figure 3.7: Feature extraction in images: The input image is used to first extract distinct corner-like keypoints, followed by a feature descriptor computation using the pixels in a local neighborhood of the keypoints. Courtesy: *Vedaldi and Fulkerson.*

Algorithm 1 Algorithm to compute binary descriptor for a keypoint.

Select a patch (W) around a keypoint in an image (I).
Select a pre-defined set ($\mathcal{S}$) of pixel pairs from within this patch.
For each such pixel pair $(s_1, s_2) \in \mathcal{S}$:
if $I(s_1) \leq I(s_2)$ **then**
    $b \leftarrow 1$
else
    $b \leftarrow 0$
Concatenate b's into a binary string ($\boldsymbol{B}$)

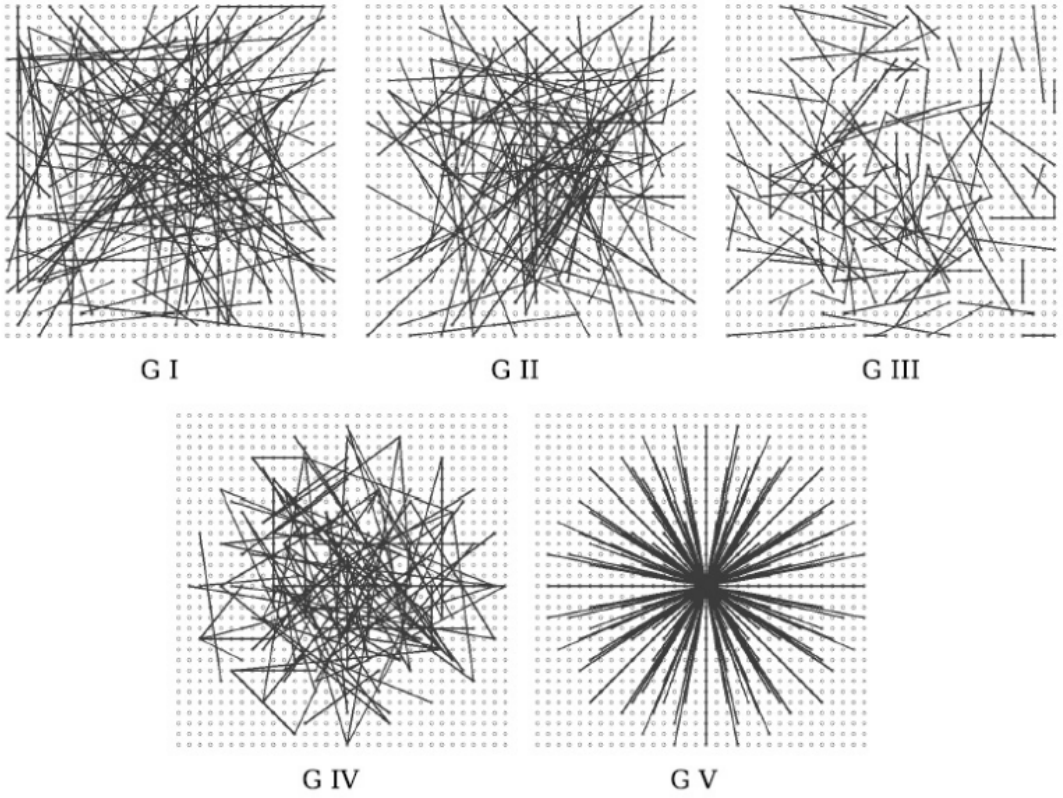


Figure 3.8: Sampling strategies for pixel pairs in BRIEF and ORB descriptors. G I is uniform sampling, G II and G III are Gaussian sampling, and G IV and G V are sampled from a coarse polar grid. Courtesy: *Calonder et al. [5].*

ORB [43] uses the FAST [41] features along with the Harris corner measure [18] to generate the keypoints. They use the BRIEF [5] binary descriptor with an additional processing step to make it rotation invariant. They use the moments of a patch computed as in Equation (3.5) and compute the intensity centroid and orientation of the patch as shown in Equation (3.6) and Equation (3.7) respectively.

$$m_{pq} = \sum_{x,\,y} x^p y^q I(x, y) \tag{3.5}$$

$$C = \left(\frac{m_{10}}{m_{00}}, \frac{m_{01}}{m_{00}} \right) \tag{3.6}$$

$$\theta = \text{atan2}(m_{01}, m_{10}) \tag{3.7}$$

The binary descriptor allows the use of a simple Hamming distance metric (see Equation (3.8)) to compare two descriptors efficiently. Note that we tune the ORB feature computation to not perform the extra steps necessary for scale invariance.

$$\begin{aligned} \boldsymbol{u} &= (u_1, \ \ldots, u_n) \in \{0,1\}^n \\ \boldsymbol{v} &= (v_1, \ \ldots, v_n) \in \{0,1\}^n \\ d(\boldsymbol{u}, \boldsymbol{v}) &= \#\{i : u_i \neq v_i, \ i = 1, \ldots, n\} \end{aligned} \tag{3.8}$$

3.5 Feature Database

Place recognition using local features relies on a database to store and retrieve all the features from the laser scans recorded at previously visited locations. An efficient data structure for such a database allows for a faster matching of the features. A simple database would be a list of all the feature descriptors, but this would require an expensive brute-force search to match new features with the database. Another commonly used database for image-based place recognition is the bag-of-words (BoW) by Sivic and Zisserman [53], where a clustering algorithm is used to group together similar features into clusters (words). The features in each frame are collectively represented as a histogram over these bag-of-words, and frame-to-frame matching is performed by comparing the distances between these histograms using a suitable metric. This method, however, requires a pre-training step over the feature set to compute the clusters, making it unsuitable for real-time applications in unknown environments. In this work, we make use of the Hamming distance embedding binary search tree (HBST) by Schlegel et al. [47], which does not require any pre-trained information. It is also available as an open-source C++ header-only library on GitLab.

HBST is a binary search tree data structure for storing bit-string data, as is the case for the ORB [43] features we use. It allows for efficient image retrieval through binary descriptor matching. For a given set $\{\boldsymbol{d}_j\}$ of binary descriptors extracted from a density image I_j, the binary tree stores a subset $\{\boldsymbol{d}_{i,j}\}$ of these features in each leaf node $\mathcal{L}_i$. Each non-leaf node $\mathcal{N}_i$ has exactly two child nodes, and it stores an index k_i which corresponds to the bit index for splitting the set of input descriptors. In order to construct a balanced tree, i.e., for both the children nodes of any given node to have an equal number of descriptors, HBST suggests

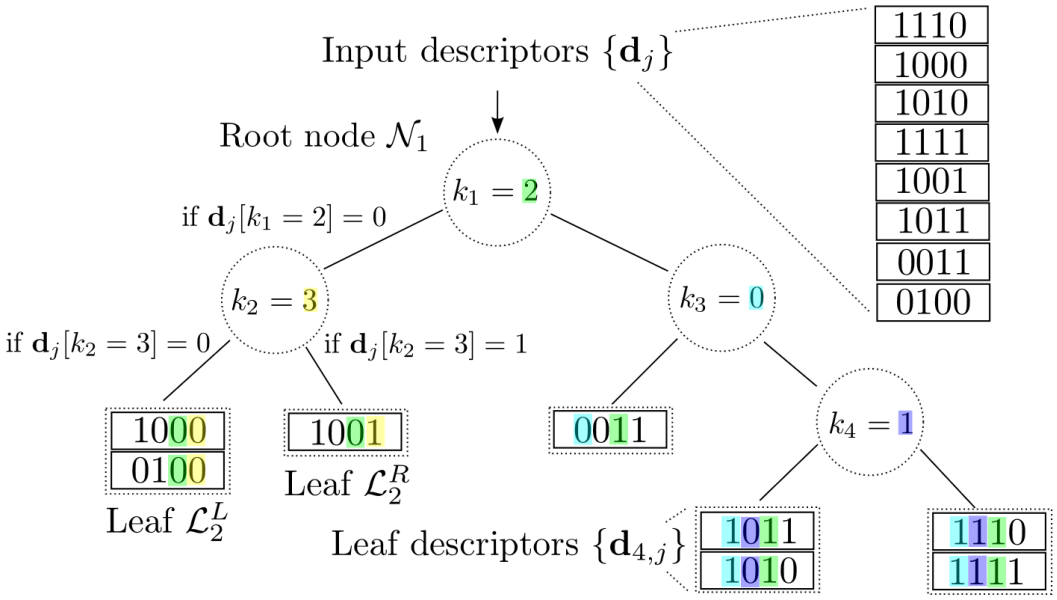


Figure 3.9: An example of HBST tree construction for 8 descriptors of length 4 bits each. The tree has four nodes $\{\mathcal{N}_i\}$ (circles) with bit indices $\{k_i\}$, five leaf nodes $\{\mathcal{L}_i\}$ (rectangles) and a maximum depth of 3. Courtesy: *Schlegel et al. [47].*

a splitting criterion corresponding to the bit which is in the active (on) state for roughly half of the N_j descriptors $\{\boldsymbol{d}_j\}$ at each node $\mathcal{N}_i$. A mathematical formulation for this splitting criterion can be seen in Equation (3.9).

$$k_i^* = \underset{k_i}{\operatorname{argmin}} \left| 0.5 - \frac{1}{N_j} \sum_j \boldsymbol{d}_j[k_i] \right|. \tag{3.9}$$

Furthermore, the HBST tree requires that any bit index k_i should appear only once during a traversal from the root node to any of the leaves. This implies that the maximum depth of the HBST binary tree is restricted by the size of the descriptors, imposing an upper bound on the number of bit comparisons required before reaching one of the leaf nodes. An illustration of an HBST tree construction can be seen in Figure 3.9. Additionally, HBST implementation also allows to store an index along with each descriptor, which is helpful for tasks such as place recognition, for which we can choose this index to be the index of the density image each descriptor belongs to.

3.6 Geometric Verification and 2D Alignment

In this section, we discuss the basic techniques we use in the geometric verification step of our approach. This geometric validation is necessary due to the incompleteness of the HBST [47] binary tree as well as the noise inherent in the

entire process, ranging from the LiDAR scans to the odometry-based local map generation and the feature computation from the density images. Since we have a set of 2D feature correspondences between two density images, we use a 2D rigid body alignment of these 2D point sets, along with a RANSAC [12] strategy for outlier rejection.

The random sampling consensus (RANSAC), as proposed by Fischler and Bolles [12], is a widely adopted paradigm for outlier rejection in model fitting problems in machine learning. The task of 2D rigid body alignment can be seen as a model fitting problem as well, where we have to estimate a transform $\mathsf{T} \in \mathbb{SE}(2)$ that aligns a set of corresponding 2D points with minimum error. As the name suggests, RANSAC involves sampling a subset of the available data such that the solution computed using this subset establishes a consensus over a majority of the dataset. The size of this subset is usually governed by the minimum number of data points required to compute a solution to the problem at hand. In our case, for 2D rigid body alignment, this corresponds to 2 matching point sets, as two points are sufficient to define a unique line segment in 2 dimensions. An overview of the RANSAC process can be seen below:

- Sample N_{min} data points from the dataset $\mathcal{S}$.
- Use these N_{min} data points to compute a solution $\mathcal{T}$ for the model parameters.
- Assign a score to this model by computing the fraction of inlier data points under this model, decided by a preset threshold.
- Repeat the above three steps until the best model is found based on the model scores, or terminate after a preset number of iterations.
- Reject all the outliers for the best model chosen and compute an approximate solution using all the inlier data points.

To compute a solution for our 2D rigid body alignment model, we make use of the Kabsch-Umeyama [23, 58] algorithm for 2D point set registration. Equation (3.10) and Equation (3.11) show the exact process of computing the 2D alignment between two pairs of matching keypoints in two dimensions.

$$\begin{aligned} \bar{\boldsymbol{p}} &= 0.5 * (\boldsymbol{p}_1 + \boldsymbol{p}_2), \\ \bar{\boldsymbol{q}} &= 0.5 * (\boldsymbol{q}_1 + \boldsymbol{q}_2), \\ \Sigma &= \sum_{i=1,2} (\boldsymbol{p}_i - \bar{\boldsymbol{p}})(\boldsymbol{q}_i - \bar{\boldsymbol{q}})^\mathsf{T}, \end{aligned} \tag{3.10}$$

where, $\boldsymbol{p}_1, \boldsymbol{p}_2 \in \mathbb{R}^2$ and $\boldsymbol{q}_1, \boldsymbol{q}_2 \in \mathbb{R}^2$ correspond to the two sampled feature matches, and

$$
\begin{aligned}
\left[U, \Lambda, V^{\mathsf{T}}\right] &= svd(\Sigma), \\
R &= V \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & \text{sign}(\det(VU^{\mathsf{T}})) \end{bmatrix} U^{\mathsf{T}}, \\
\boldsymbol{t} &= \bar{\boldsymbol{q}} - R\bar{\boldsymbol{p}},
\end{aligned} \tag{3.11}
$$

where, $R \in \mathbb{SO}(2)$ and $\boldsymbol{t} \in \mathbb{R}^2$ are the relative 2D alignment between them. Using this relative alignment from the reference features' frame to the query features' frame, we transform all the remaining reference features to the query features' frame and compute the model score for RANSAC iterations. Upon termination, the number of inliers obtained from this RANSAC alignment strategy is used to conclude if two local maps belong to the same location or not, and the corresponding pose estimates are also used as the initial guess to the pose-graph alignment pipeline within a SLAM framework.

Chapter 4

Approach

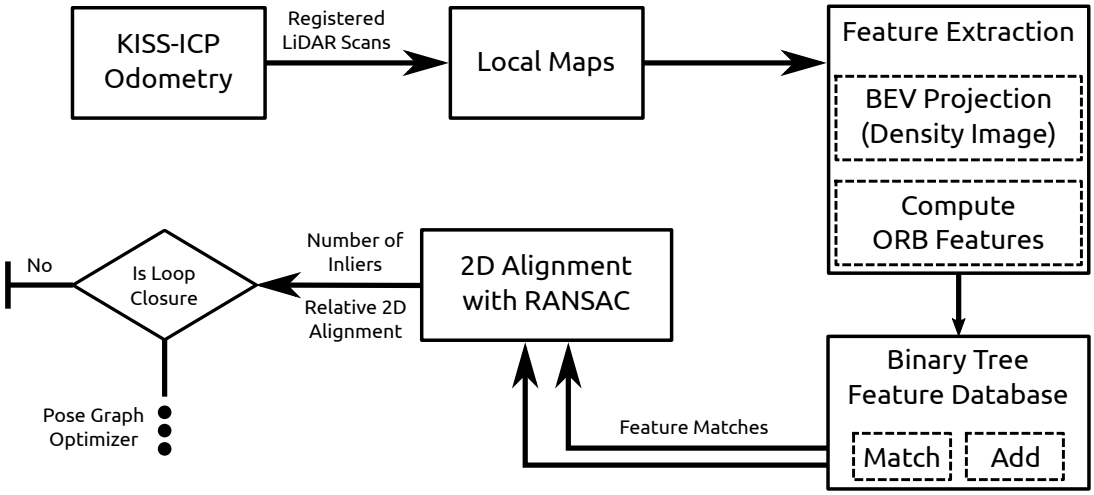


Figure 4.1: A block diagram of our approach towards place recognition with 3D LiDARs.

Our proposed approach performs place recognition with 3D LiDARs at a local map level and provides a complete 2D initial guess on the relative pose between matched maps. Figure 4.1 depicts a block diagram of the sub-tasks in our pipeline. First, we generate the local maps using pose estimates from a LiDAR odometry pipeline, as explained in Section 4.1. In Section 4.2, we show how the birds eye view projection density images are obtained from these local maps. Then, in Section 4.3, we discuss the subsequent steps for computing local features and descriptors from these density images and creating a database for storing them. This database is also used to perform feature matching of the previously seen local maps with the incoming query local map. Finally, Section 4.4 elaborates on the geometrical verification step performed to validate the set of matches obtained by the naive feature matching done in Section 4.3.

4.1 Local Maps

Our approach for place recognition and loop closure detection with 3D LiDARs uses local maps as the primary representation of the environment. We build a local map by accumulating consecutive LiDAR scans with their relative pose estimates obtained from an ICP odometry pipeline. In particular, we use the KISS-ICP [61] odometry pipeline, a state-of-the-art approach for LiDAR odometry with strong open-source community support. An implicit assumption made by our approach is that a good LiDAR odometry pipeline will provide us with locally consistent pose estimates. This can be verified from the trajectory plot in Figure 4.2.

To generate the local maps, we consider a set of n consecutive 3D point clouds $\{P_i, P_{i+1}, \ldots, P_{i+n-1}\}$ recorded from a LiDAR in its local reference frame, and transform them into the reference frame of the i^{th} scan $\{{}^{i}P_i, {}^{i}P_{i+1}, \ldots, {}^{i}P_{i+n-1}\}$ as follows:

$$ {}^{i}P_{i+n-1} = \mathsf{T}_i^{-1}\mathsf{T}_{i+n-1}P_{i+n-1}, \tag{4.1} $$

where $\mathsf{T}_i \in \mathbb{SE}(3)$ is the odometry estimate of the pose connected to the i^{th} scan. We aggregate all these scans into a local map $\mathcal{M}_i$ using a voxel grid with a resolution of $\nu_{map} \times \nu_{map} \times \nu_{map}$ cubic meters per voxel. We store the voxels in a hash table instead of a naive 3D array. For this, we use the implementation provided in the KISS-ICP [61] repository on GitHub. This allows for a memory-efficient representation of the voxel grid local map. To ensure a uniform density of points in the voxel grid, we only retain a maximum of 20 points per voxel. The local map $\mathcal{M}_i$ is centered at the frame T_i, the first scan of the map. The size of this map $\mathcal{M}_i$ is decided based on the distance traveled by the sensing platform. We accumulate n consecutive scans until the travel distance $\|\boldsymbol{t}_{i+n-1} - \boldsymbol{t}_i\|_2$ exceeds a threshold of τ_m meters, where $\boldsymbol{t}_i \in \mathbb{R}^3$ is the translational component of the odometry pose estimate T_i.

Some samples of such local maps can be seen in Figure 4.3. These local maps allow us to aggregate more information about the environment even when working with small field-of-view LiDARs like the Livox Avia, as it can be seen from the local map in Figure 4.3b. Although these maps contain sufficient distinctive features for place recognition, extracting such features directly from a large, dense 3D representation is computationally expensive. In order to reduce the computational complexity, at the same time preserving necessary information for place recognition, we project these local maps into a 2D BEV representation as discussed in Section 4.2.

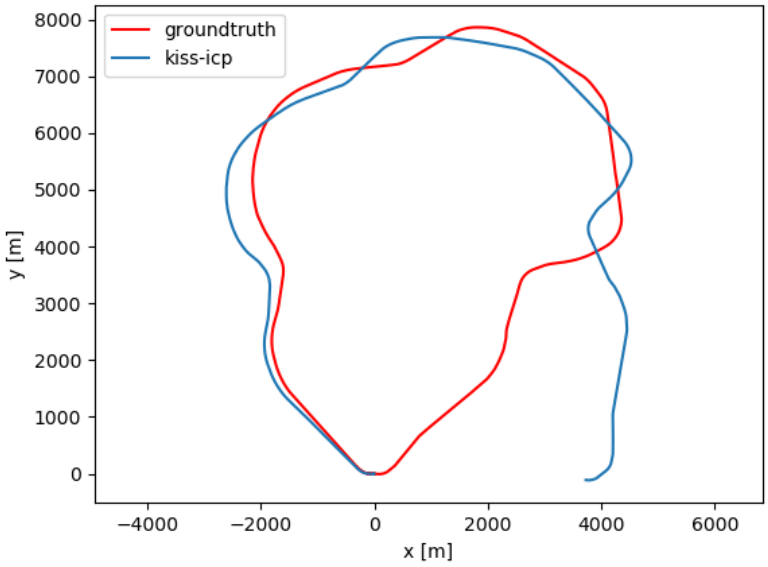
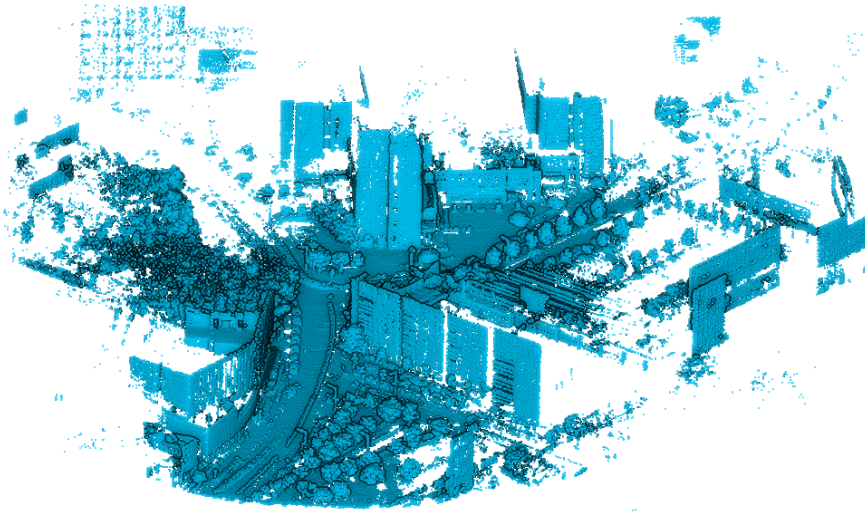
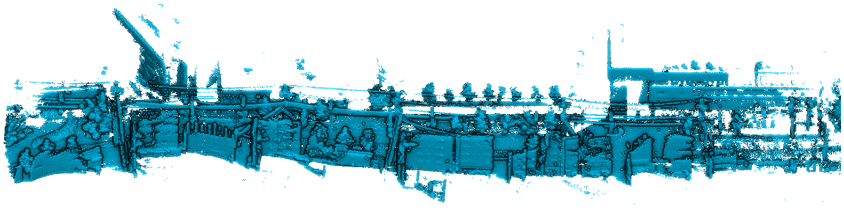


Figure 4.2: KISS-ICP odometry estimates for MulRan dataset Sejong01 sequence. It can be seen that even though the odometry estimates (blue) drift over time, the local shape of the trajectory is maintained with respect to the ground-truth pose estimates (red).



(a) MulRan KAIST03: Ouster OS1-64 LiDAR.



(b) HeLiPR Town01: Livox Avia LiDAR.

Figure 4.3: Example of local maps generated using the LiDAR odometry pose estimates for two different LiDAR sensors.

4.2 Density Images

The BEV projection of point clouds is a widely used approach in place recognition using LiDARs [24, 26, 28, 32, 33]. Such a projection preserves local 2D geometry and, at the same time, also reduces the computational complexity of algorithms due to a reduced dimension to process. However, a lot of the traditional BEV projection approaches [24, 26, 28, 33] store the maximum height (elevation map) of the points in each bin, not point densities. The elevation map is sensitive to the orientation of the sensor as the maximum height recorded varies with the distance between the scanner and the object. The density of points scanned on a surface, on the other hand, is less sensitive to viewpoint changes.

For a local map $\mathcal{M}_i$, we project all the points to the local ground plane through an orthographic BEV projection. When the reference frame T_i of the local map $\mathcal{M}_i$ has its XY-plane parallel to the ground, as is the case for ground robots/vehicles, this would correspond to simply dropping the Z-coordinate of all the individual points. For other robots, such as UAVs, we require a gravity vector that can be directly obtained from an inertial measurement unit (IMU). The BEV projection gives us a set of points $\boldsymbol{B}_i \in \mathbb{R}^2$ bounded by a rectangular window from $[x_i^l,\, y_i^l]^\mathsf{T}$ to $[x_i^u,\, y_i^u]^\mathsf{T}$ meters where,

$$\begin{aligned} \begin{bmatrix} x_i^u \\ y_i^u \end{bmatrix} &= \max_{x,\,y} \boldsymbol{B}_i \\ \begin{bmatrix} x_i^l \\ y_i^l \end{bmatrix} &= \min_{x,\,y} \boldsymbol{B}_i. \end{aligned} \tag{4.2}$$

We further discretize $\boldsymbol{B}_i$ into a 2D Cartesian grid $\boldsymbol{N}_i(u,\, v) \in \mathbb{N}_0^{W_i \times H_i}$ of resolution $\nu_{res} \times \nu_{res}$ square meters per cell where,

$$\begin{aligned} W_i &= \left\lceil \frac{x_i^u - x_i^l}{\nu_{res}} \right\rceil \\ H_i &= \left\lceil \frac{y_i^u - y_i^l}{\nu_{res}} \right\rceil. \end{aligned} \tag{4.3}$$

Each cell in this grid $N(u, v)$ stores the number of points contained in that cell after discretization. The grayscale density image $\boldsymbol{I}_i(u,\, v)$ of the local map $\mathcal{M}_i$ is then defined as,

$$\boldsymbol{I}_i(u, v) = \frac{N(u, v) - N_{min}}{N_{max} - N_{min}} \in \mathbb{R}^{W_i \times H_i}, \tag{4.4}$$

$$\begin{aligned} \text{where, } N_{max} &= \max_{u,\,v} N_i(u, v) \\ N_{min} &= \min_{u,\,v} N_i(u, v). \end{aligned} \tag{4.5}$$

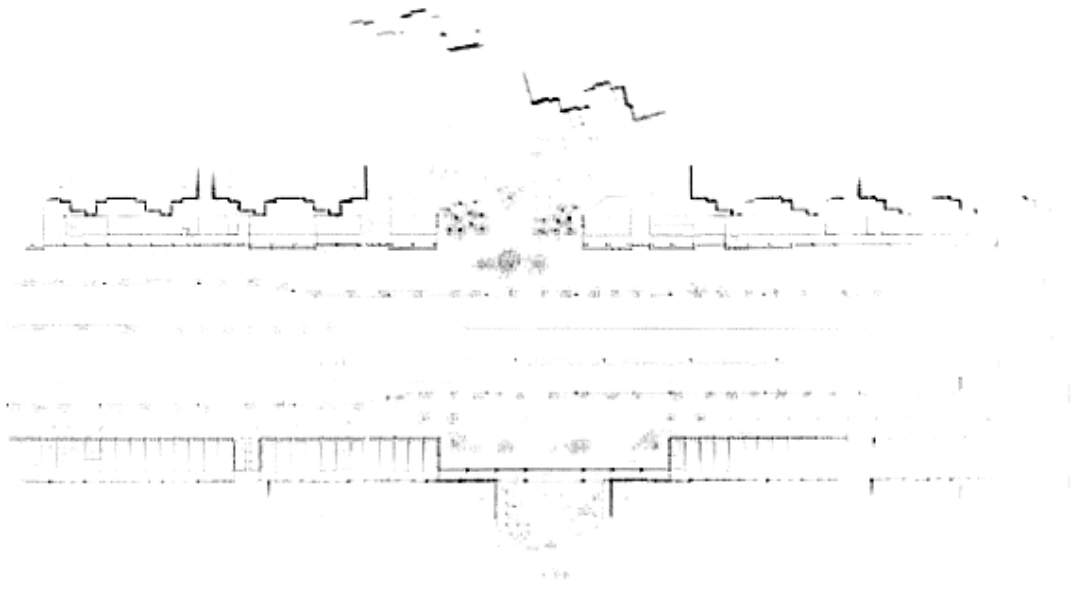


Figure 4.4: A density image generated from the BEV projection of a local map. The darker pixels represent a higher density of points, and vice versa. The facades of the structures along the path and roadside features such as streetlamps and electric poles are captured by the density map.

We set all the image pixels (u, v) with density lower than 5 % of the maximum value to be zero. This helps to reduce the noise in the local maps and to remove the ground plane, which is typically insignificant for feature detection. These density images capture 2D floorplan-like information from the local maps, like building facades, streetlamps, and trees, as can be seen in the density image in Figure 4.4. These rigid structures provide reliable features to track for place recognition.

4.3 Feature: Detection, Matching and Database

The task of place recognition requires to have a description of the scene as perceived by the sensors. These descriptions could be either at a global scale, summarizing the entire scene into one descriptor, or at a local scale, summarizing multiple local patches in the scene separately. We follow the latter paradigm in this work. Facilitated by the accumulation of information in the form of the local map density images I_j, we utilize the well-known Oriented FAST and Rotated BRIEF (ORB) [43] feature descriptors to capture corner-like features from them. To speed up the computation of these features, we use ORB features without scale invariance. We can do this since the density image is an orthographic projection of the 3D world and has no scale ambiguity as compared to regular camera

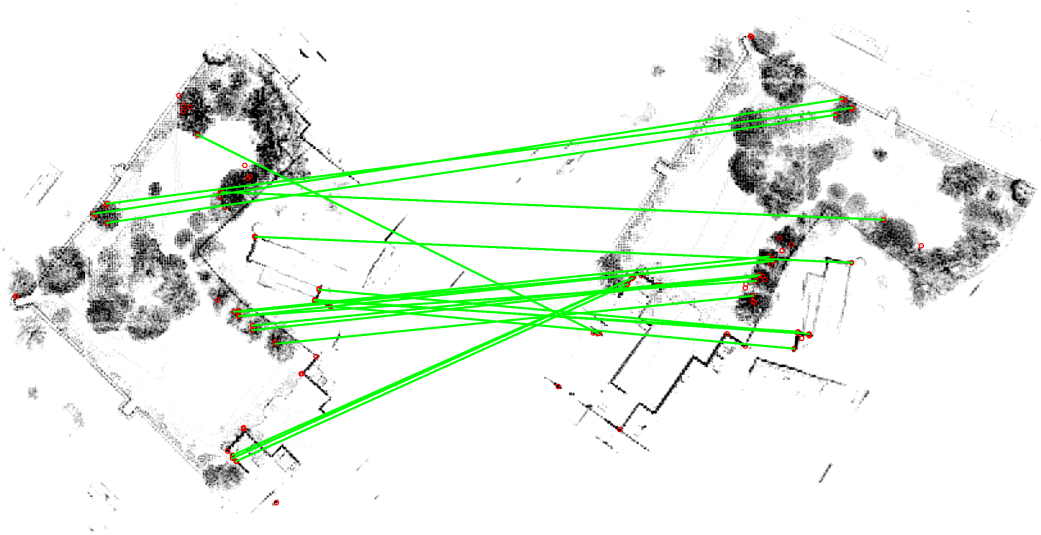


Figure 4.5: ORB features (red) and matches (green) between two density images from the same location, revisited from a different viewpoint.

images, which are perspective projections of the real world. Figure 4.5 shows in red the ORB features computed on two density images belonging to the same physical location.

The binary domain of ORB descriptors furthermore allows for efficient feature storage and matching by leveraging the Hamming distance metric. In particular, we employ the Hamming distance embedding binary search tree (HBST) [47] to store the set of feature descriptors D_i obtained from each density image $I_i(u, v)$ along with the corresponding map index i. The depth of the HBST is bounded by the number of bits in the binary descriptor (256). This means that a query descriptor would have a maximum of 256 bitwise comparisons with the tree's nodes before it terminates in one of the leaf nodes. Furthermore, the capacity of each leaf node is limited to a maximum of 100 descriptors per leaf node. Fixing these two design parameters imposes an upper bound on the computational time of the feature-matching process while not introducing any restriction in the practical use of our approach. A binary tree, however, is not a complete data structure for nearest-neighbor computation, i.e., the nearest-neighbor estimate from such a tree is not guaranteed to be the true nearest-neighbor always. As a result, it can only be used as a data structure to get good initial estimates on the feature matches, requiring a subsequent geometric verification and refinement strategy.

After obtaining a new set of descriptors D_q from the query local map's density image I_q, we find the nearest match for each descriptor in $d_{qj} \in D_q$ from the binary tree database. We use a threshold of 50 bits on the Hamming distance between two 256 bit ORB descriptors to call them a match. Since the binary tree stores descriptors along with an index i of the corresponding map they were obtained

from, we cast a vote over the map indices for each such match. Once all the query descriptors have been processed, we select the reference maps corresponding to top-N votes from this voting scheme. We set N to be equal to half the number of local maps in the database at any time. This dynamic factor allows us to find multiple potential loop closures at the same physical location, the chances of which increase with the number of local maps in the database. As a result, we obtain a list of feature matches between the query map and the reference maps in the database, on which we perform a geometrical verification. Figure 4.5 also visualizes the corresponding set of feature matches between the ORB features computed on two density images from the same physical location.

4.4 Loop Detection and Map Alignment

The geometrical verification step involves a 2D alignment of the matched features. This implies computing a 2D rigid body transformation (3 degrees of freedom) that aligns the matched features from the binary tree in the best way possible based on a distance metric. This is similar to the image-alignment problem but limited to an $\mathbb{SE}(2)$ transform instead of a homography. We use a Random Sampling Consensus (RANSAC) [12] paradigm for our alignment strategy to reject outlier associations accounting for the incompleteness of the binary tree-based matching. This verification step is performed for matches between the query map and each of the top-N reference maps separately.

We only need two sets of matching keypoints to compute a rigid body alignment in 2D. Using this fact, we design a RANSAC scheme, randomly drawing 2 feature matches from the entire set of feature matches between a query (I_q) and a reference (I_p) density image. We compute the relative 2D alignment between them using the Kabsch-Umeyama [23, 58] algorithm. It provides us with a rotation matrix $R \in \mathbb{SO}(2)$ and a translation vector $\boldsymbol{t} \in \mathbb{R}^2$, which can be composed into a homogenous transformation ${}^{q}\mathsf{T}_{p} \in \mathbb{SE}(2)$. Using this 2D alignment, we transform all the features from I_p to the reference frame of I_q.

For verification within RANSAC, we compute the point-wise error between the matching features in terms of the Euclidean distance between the matched point sets. Matches with distances larger than 1.5 m (3 pixels for $\nu_{res} = 0.5\,\mathrm{m}$) are considered as outliers. The RANSAC scheme terminates after a fixed number of iterations or if a high-quality solution is found, i.e., in case more than 30 inliers are obtained.

We define a threshold γ for the minimum number of inliers required from the RANSAC alignment to conclude whether two local maps belong to the same location. The associated 2D transform between the two density images is already a good initial estimate for the complete 3D transform between local maps in the

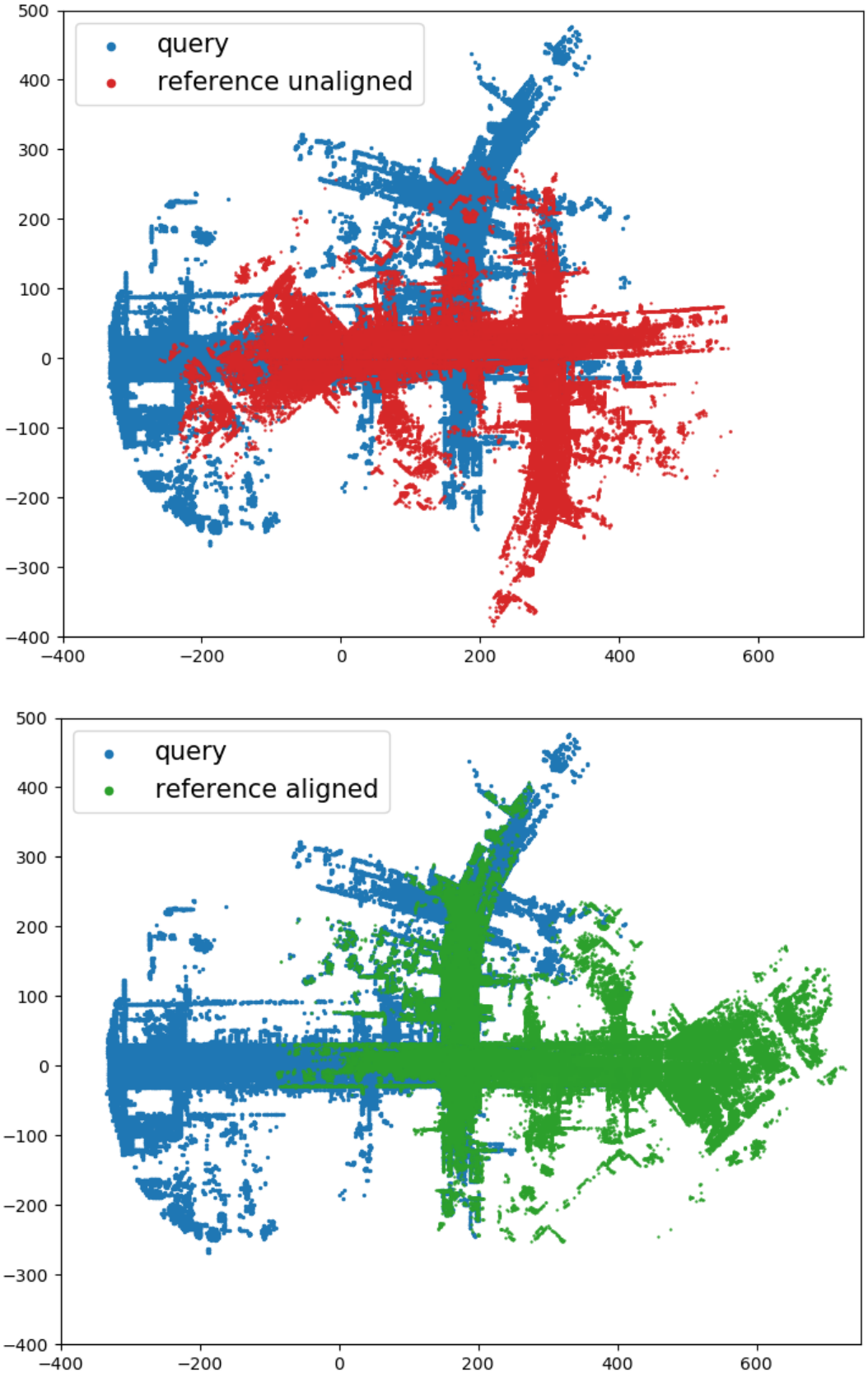


Figure 4.6: Bird eye views of two local maps belonging to the same physical location visited from opposing viewpoints, before alignment (top) and after alignment (bottom) using the 2D initial guess from our pipeline.

case of ground robots/vehicles. Note that we scale the translation vector by the voxel size ($\nu_{res}\boldsymbol{t}$) to undo the effect of the discretization while generating the density images. This initial alignment can later be used as an initial guess for a fine point cloud registration in 3D. An illustration of the quality of the 2D alignment can be seen in Figure 4.6 for the case of a revisit from a $180\,^\circ$ opposite viewpoint.

Chapter 5

Experiments

This work provides a pipeline to compute loop closures for SLAM using local maps. We present our experiments to show the capabilities of our method. The results of our experiments also support our key claims, which are: (i) We can effectively detect loop closures between two temporally separated local maps in a variety of environments, (ii) We provide a 2D rigid body transform between detected loop closures and (iii) Our approach is agnostic of the sensor scan pattern by exploiting local maps.

5.1 Evaluation Criteria

The robotics community lacks consensus on what a true loop closure is, with each approach defining its own criteria. Related works [20, 26, 63] usually consider two scans as a true positive closure if the distance between the corresponding reference locations is below a certain threshold. This implies a potentially wrong assumption that a sensor observes the same objects when being at a similar location. This does not hold when other objects occlude the previously seen area, when the sensor has a limited field of view, or when the orientation differs strongly. Jiang et al. [20] also highlights this issue in their recent publication, suggesting a complete rethinking of the evaluation criteria. This work also makes an attempt to define such a criterion. To assess the performance of loop closure detection, we identify scan-wise reference closures based on the volumetric overlap of measured point clouds.

We propose to use the volumetric overlap of scans to decide if a loop should be closed. We argue that if the point clouds recorded from two different points in time have sufficient overlap, then we can find the relative pose between them and integrate the loop closure into a pose-graph. The first step of our reference identification is to sample the reference trajectory at equidistant locations (2 m) to reduce the number of candidates. Next, we accumulate the registered scans

between two consecutive key locations to have a dense representation of the local environment. We find all possible pairs of key locations within a distance of the sensor's maximum range (r) with a minimum travel distance. Further, we voxelize the accumulated point clouds of both keyframes. We use a voxel size of 0.5 m. Finally, we compute the overlap $o \in [0, 1]$ between two non-empty voxel grids $\mathcal{V}_i$ and $\mathcal{V}_j$ using the overlap coefficient [59] as in Equation (5.1), also called the Szymkiewicz-Simpson coefficient:

$$o\left(\mathcal{V}_i, \mathcal{V}_j\right) = \frac{\left|\mathcal{V}_i \cap \mathcal{V}_j\right|}{\min\left(\left|\mathcal{V}_i\right|, \left|\mathcal{V}_j\right|\right)}. \tag{5.1}$$

We consider two keyframes to be a loop closure if their overlap $o > 0.5$. Also, since all other benchmarks detect loop closures between individual scans, we cannot make a direct comparison of these baselines with our approach, which detects loop closures between local maps. So, for a fair evaluation, we need to compute scan index closures from the local map closures. To achieve this, we apply the 2D pose correction ${}^{q}\mathsf{T}_p$ between maps to the individual scans comprising these maps, transforming them to a common reference frame. Then, we compare all the pairs of scan-level poses between the two maps, computing the pairwise distance between them. Finally, we use a distance threshold τ_d to classify whether two scan indices are predicted to be a loop closure or not.

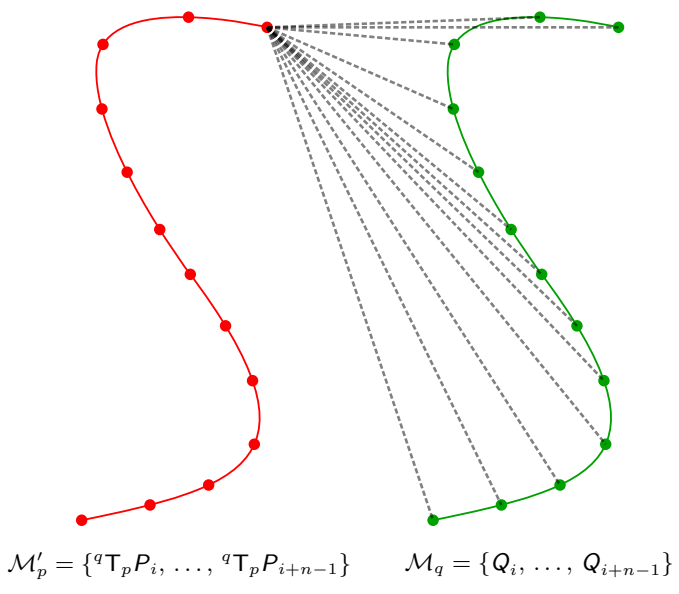


Figure 5.1: The individual poses comprising the two loop closed local maps $\mathcal{M}'_p$ (left) and $\mathcal{M}_q$ (right) are represented by dots. A pairwise distance comparison is made between the two sets of poses (example shown with dotted lines) to get loop closures between scan indices.

5.2 Experimental Setup

We evaluate our work on the MulRan [25], Newer College [39], and HeLiPR [22] datasets. From the MulRan [25] dataset, which is recorded from an Ouster OS1-64 LiDAR mounted on a car, we use the KAIST03 (MK03) sequence having a city-like environment typical of many other self-driving car datasets, the Riverside02 (MR02) sequence which is recorded along a river bank, and the Sejong01 (MS01) sequence which is a long traversal on a highway. Together, they contain loop closures from varying viewpoints and at different magnitudes of travel distances. The Newer College [39] 01_short_experiment (NCD) sequence also provides data recorded from an Ouster OS1-64 LiDAR, but it is mounted on a hand-held scanning platform providing a different motion profile and viewpoints for detecting loop closures. Finally, to support the claim of our method being sensor agnostic, we use the Town01 (HT01) sequence from the HeLiPR [22] dataset. It has laser scans from a Livox Avia LiDAR, which has a small field of view and a rather unusual scanning pattern.

We compare our method with three other baselines: ScanContext-10 (SC) [26], ContourContext (CC) [20], and Stable Triangle Descriptors (STD) [63]. The ScanContext is a popular approach and also among the current state-of-the-art methods. It is widely used for place recognition in the context of SLAM. On the other hand, ContourContext and Stable Triangle Descriptors are a couple of recently published approaches for detecting loop closures using 3D LiDARs. A major challenge in selecting the baselines was the availability of a working code repository to obtain the evaluation metrics on the datasets being used. All the above baselines provide a publicly available working code for evaluation. We slightly modify these baseline implementations, enabling them to provide multiple closure candidates for each query scan instead of only the best candidate, keeping all other parameters as default. This ensures a fair comparison, as our approach also provides multiple closure candidates for each query local map. Furthermore, since the STD pipeline proposes accumulation of 10 consecutive scans into a local map, we utilize the same method as in our approach (See Figure 5.1), to obtain scan index closures from map index closures, with $\tau_d = 6\,\text{m}$.

For our approach, we set the maximum range of the scanner r as well as the local map truncation distance τ_m to be equal to 100 m. For the KISS-ICP [61] odometry pipeline, we use the default parameters provided. We use a voxel size of $\nu_{map} = 1.0\,\text{m}$ for the voxel grid used to generate the local maps and a resolution of $\nu_{res} = 0.5\,\text{m}$ for the density images. For ORB feature descriptors, we use the default parameters provided, only disabling the parameters controlling scale invariance for reasons discussed in Section 4. We require a minimum of 25 feature matches from the HBST within a Hamming distance of 50 bits. All other parameters of the binary tree (HBST) are set to the defaults. Finally, we predict

two local maps to be loop closures based on the number of inliers obtained from the RANSAC-based alignment strategy, the threshold for which is set to be $\gamma = 10$. Finally, to compute scan index loop closures for quantitative evaluation, we use a distance threshold of $\tau_d = 6\,\text{m}$. These parameters are maintained across all datasets except for the HeLiPR Town01 dataset, for which we use a maximum range of $r = 50\,\text{m}$, $\nu_{map} = 0.5\,\text{m}$, and a minimum of 10 feature matches from the HBST. This is to account for the small field-of-view and unusual scan pattern of the Livox Avia LiDAR.

5.3 Performance Evaluation

The performance of our approach is shown in Table 5.1 as a comparison of precision, recall, and F1 scores between each of the baselines and our approach for all the aforementioned datasets. Note that we report the precision and recall scores corresponding to the best achievable F1 scores for each of the approaches, including ours. The F1 score, being the geometric mean of precision and recall, is a suitable statistic for comparison. It can be observed from Table 5.1 that our approach has the best F1 score among all baselines for all datasets. Furthermore, we achieve these F1 scores at a significantly high precision as well, which is important for confidently incorporating the detected loop closures into a SLAM pipeline.

Since the MulRan KAIST03 (MK03) sequence is recorded in a typical city environment with a lot of geometrical features to track along the drive path, we can observe comparable performance across all the approaches. However, the precision-recall performance on the MulRan Riverside02 (MR02) and MulRan Sejong01 (MS01) sequences highlights the effectiveness of our proposed approach. The Sejong01 sequence, which is a 23.4 km long highway sequence, is particularly challenging due to sparsely distributed features (streetlamps and guard rails), as well as dynamic obstacles (moving vehicles). Our approach not only detects loop closures on these datasets but also manages to do so at a fairly high precision and F1 score as compared to other baselines. The Stable Triangle De-

Table 5.1: Precision (P), Recall (R) and F1 scores of state-of-the-art baselines and our approach, for all datasets.

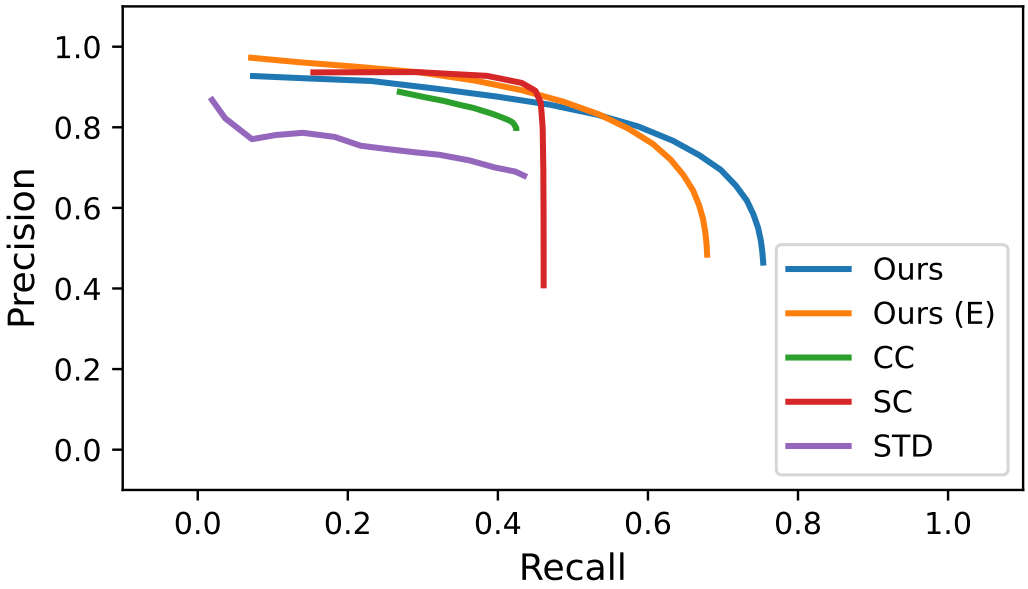
Datasets	MK03			MR02			MS01			NCD			HT01		
	P	R	F1	P	R	F1	P	R	F1	P	R	F1	P	R	F1
SC [26]	**0.890**	0.450	0.598	0.259	0.071	0.111	0.027	0.126	0.045	0.415	0.007	0.013	N/A	N/A	N/A
CC [20]	0.798	0.424	0.554	0.695	0.080	0.144	0.244	0.076	0.116	0.226	0.030	0.053	N/A	N/A	N/A
STD [63]	0.679	0.435	0.530	0.491	0.042	0.078	-	-	-	-	-	-	-	-	-
Ours	0.730	**0.668**	**0.698**	**0.939**	**0.731**	**0.822**	**0.983**	**0.298**	**0.458**	0.927	**0.068**	**0.127**	**0.473**	**0.669**	**0.554**
Ours (E)	0.759	0.606	0.674	0.639	0.118	0.199	0.978	0.296	0.454	**0.933**	0.043	0.083	-	-	-

scriptor (STD) [63] baseline even fails to run over such longer sequences having more than 15000 scans.

Another important evaluation criterion for loop closures is the precision-recall curve over the design parameters. These curves highlight the range of performance offered by the methods in terms of the trade-off between high precision and high recall. We provide in Figure 5.2 the precision-recall curves for all the baselines on the MulRan KAIST03, Riverside02, and Sejong01 sequences, as well as the Newer College short experiment sequence. Our approach not only provides the best F1 score for all these datasets as highlighted in Table 5.1 but also can be adapted to provide either high precision or high recall based on the distance threshold parameter (τ_d).

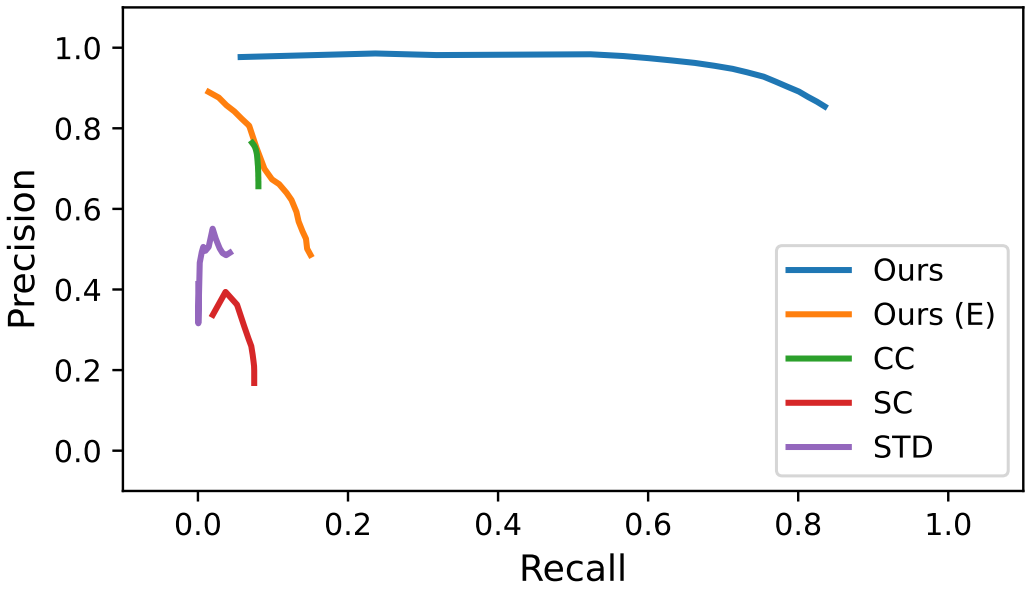
We also ablate our choice of density images over elevation maps by modifying our pipeline to use elevation images of the local maps. We present the results for the same in the last row (Ours(E)) of Table 5.1. It can be seen that the elevation image-based approach performs at par or worse than the density image-based approach that we propose. Also, it completely fails on challenging datasets like the HeLiPR Town01 (HT01).

Overall, this evaluation highlights our approach's ability to detect loop closures in a variety of environments. Furthermore, it is an indirect evaluation of our 2D pose estimates between detected loops, as we use these estimates when computing the scan index level closures.

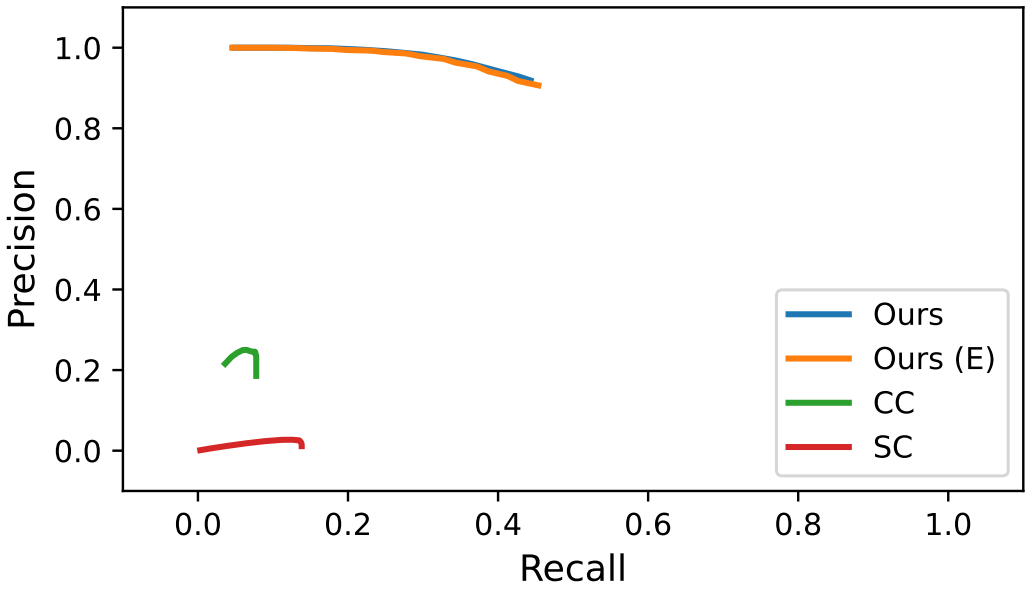


(a) MulRan KAIST03

Figure 5.2: The precision-recall curves for our approach and other baselines for the mentioned datasets.



(b) MulRan Riverside02

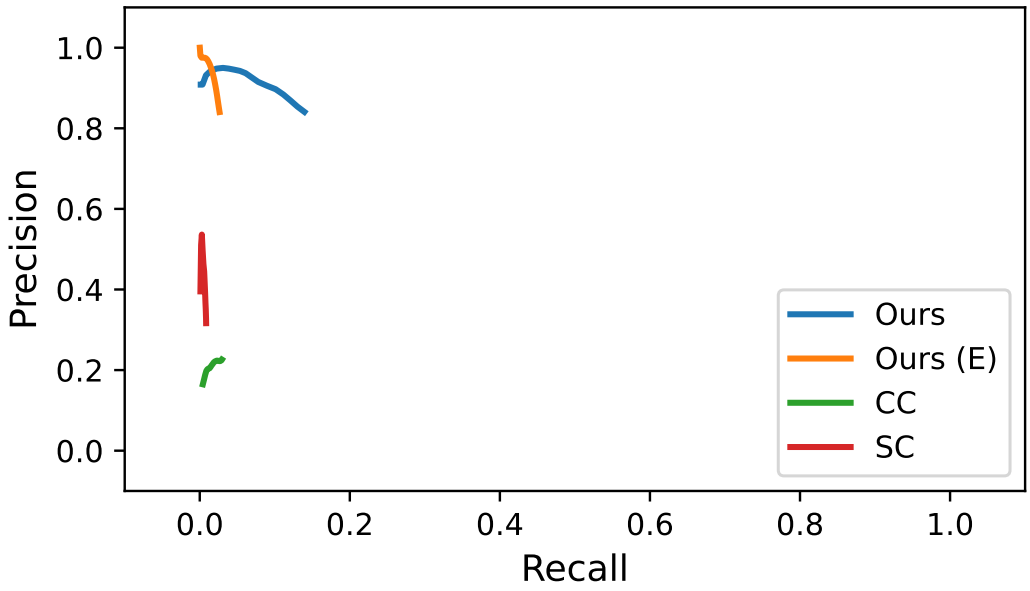


(c) MulRan Sejong01

Figure 5.2: The precision-recall curves for our approach and other baselines for the mentioned datasets. (contd.)

5.4 Livox LiDAR Scan Pattern

The HeLiPR dataset provides scans recorded from a Livox Horizon LiDAR, which has a very unusual scanning pattern as against traditional LiDARs, with a higher density but small field-of-view. This poses a significant challenge in detecting loop closures. However, our approach is capable of overcoming challenges, primarily



(d) Newer College Dataset 01_short_experiment

Figure 5.2: The precision-recall curves for our approach and other baselines for the mentioned datasets. (contd.)

due to the use of local maps. As can be seen from the last column (HT01) in Table 5.1, we achieve an F1 score of 0.554 on this dataset, also being the only method to even work on this dataset. STD [63], even though capable of processing Livox scans due to aggregation of a fixed number of scans, fails on this dataset due to the large size of the HT01 sequence.

5.5 Offline Optimization with Loop Closures

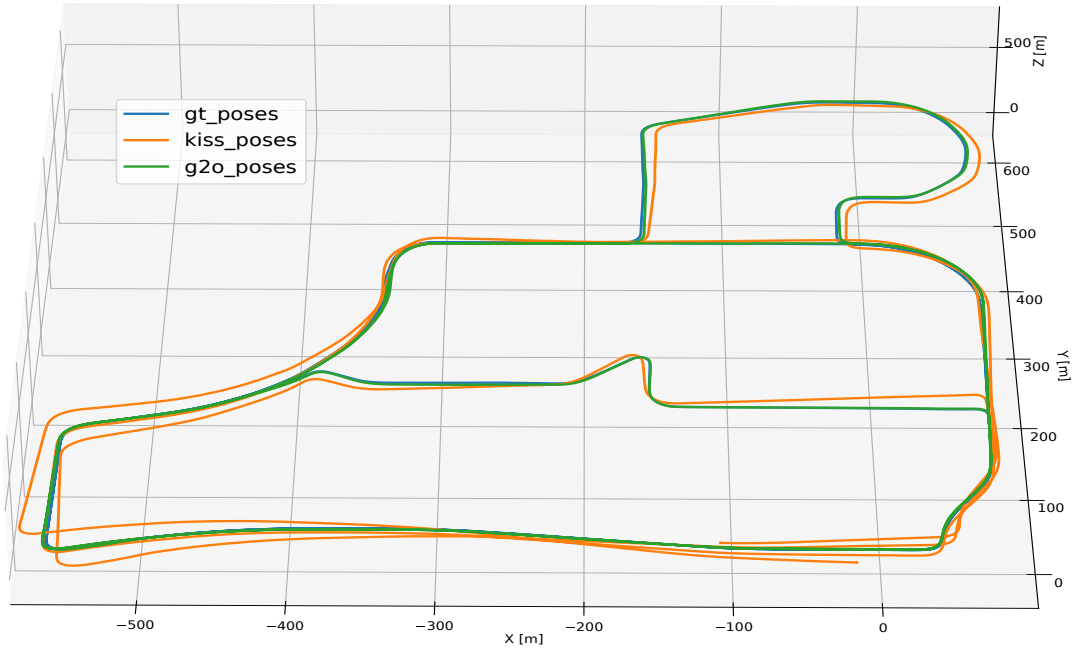
In this final experiment, we showcase the ability of our approach to perform drift correction in a SLAM pipeline. We perform an offline pose-graph optimization, including all the detected map-level loops. A fine ICP registration is performed on the detected loops between local maps with the initial guess provided by our pipeline. The refined 3D pose estimate between the local maps is then incorporated as a constraint in the pose-graph along with the odometry constraints. We use the open-sourced g2o [27] pipeline.

In Figure 5.3, we show a comparison of pose estimates for the MulRan Sejong01 dataset, which is a 23.4 km long highway sequence, with an odometry drift in the order of a few thousand meters. It can be seen that the inclusion of our detected loop closures reduced the drift by a significant amount. This is further confirmed by the values of absolute pose error (APE) in translation and rotation with respect to the ground-truth poses, as shown in Table 5.2. We provide the

Table 5.2: Absolute Pose Error (APE) in translation and rotation, with (w) and without (w/o) loop closures.

Datasets	# Closures	APE tra. (m) (rms)		APE rot. (rms)	
		w	w/o	w	w/o
MK03	73	3.04	14.64	0.05	0.09
MR02	33	12.70	32.58	0.08	0.11
MS01	2	206.6	1248.6	0.48	0.63
NCD	7	0.61	0.62	0.03	0.03
HT01	2	12.92	18.48	0.14	0.15

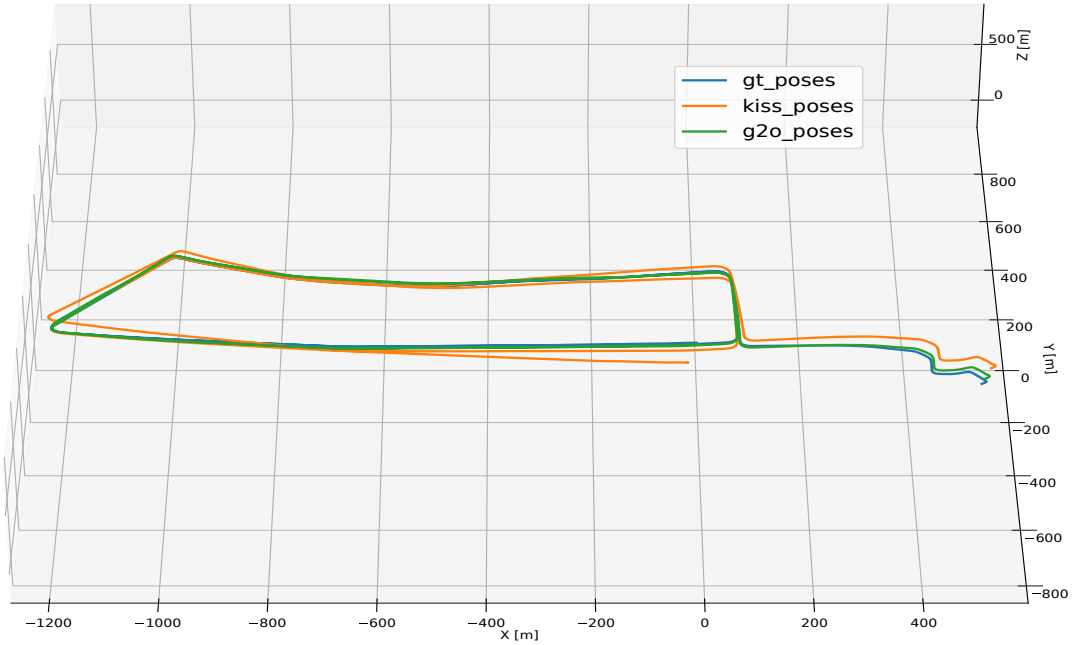
APE values with and without the inclusion of loop closures for all the datasets to highlight the significant impact our approach for detecting loop closures can have over the odometry estimation. Furthermore, we achieve this improvement in APE with a significantly lower number of loop closures, as can be seen in Table 5.2, allowing for faster optimization of the pose-graph.



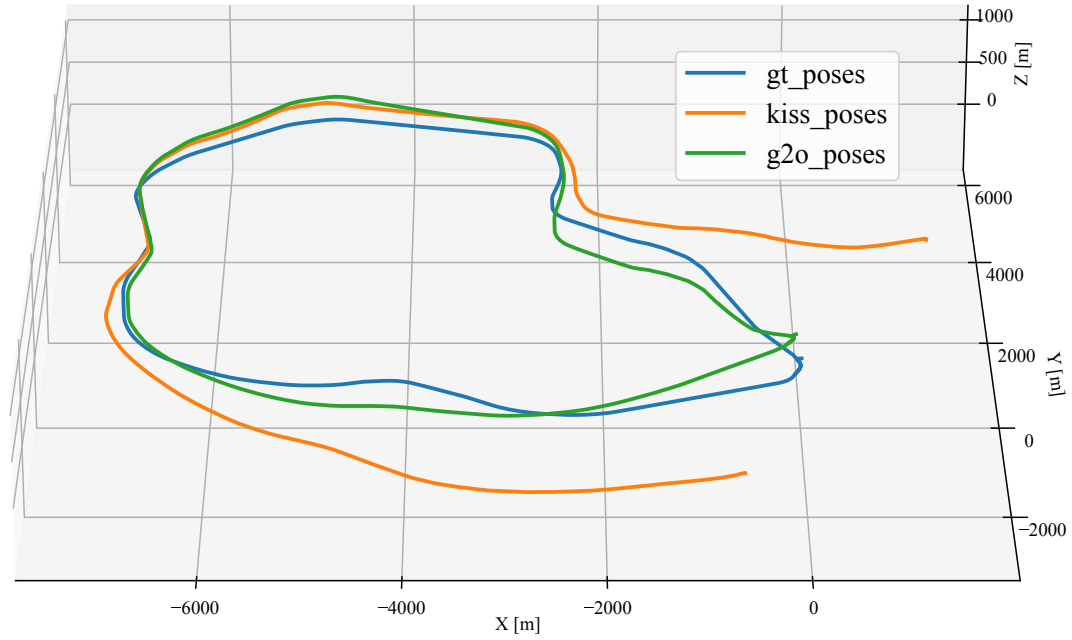
(a) MulRan KAIST03

Figure 5.3: Comparison of pose estimates obtained after pose-graph optimization with loop closures, compared against the LiDAR ICP odometry estimates and reference poses provided along with each dataset.

In summary, our evaluation suggests that our method can effectively detect loop closures with local maps, with a better or at par performance compared to

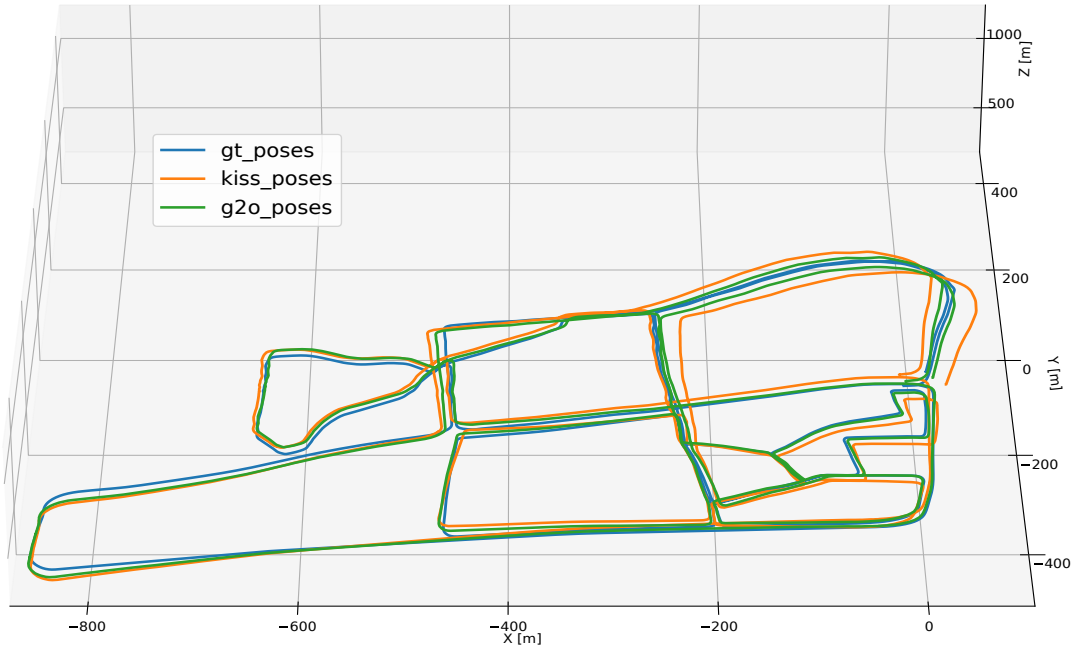


(b) MulRan Riverside02

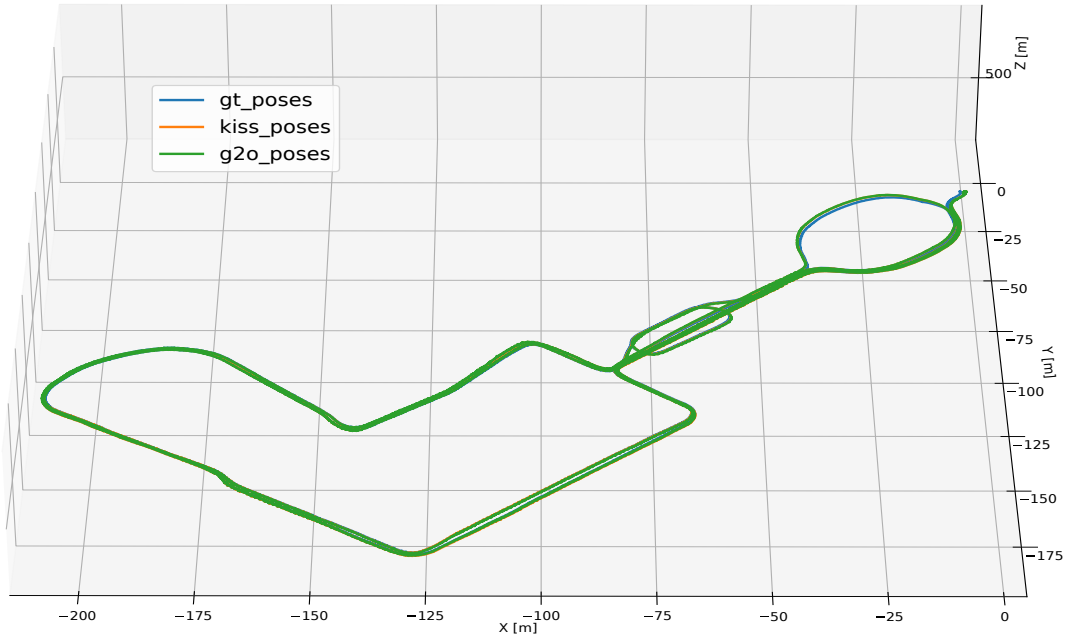


(c) MulRan Sejong01

Figure 5.3: Comparison of pose estimates obtained after pose-graph optimization with loop closures, compared against the LiDAR ICP odometry estimates and reference poses provided along with each dataset. (contd.)



(d) HeLiPR Town01



(e) Newer College Dataset 01_short_experiment

Figure 5.3: Comparison of pose estimates obtained after pose-graph optimization with loop closures, compared against the LiDAR ICP odometry estimates and reference poses provided along with each dataset. (contd.)

other approaches, on a variety of datasets. The accumulation of scans into local maps makes our method agnostic of the sensor scan pattern and density. Finally, we also demonstrated that our detected loops provide a significant drift reduction when incorporated into a pose-graph. Thus, we supported all our claims with this experimental evaluation.

Chapter 6

Conclusion

In this thesis, we presented a novel approach to computing loop closures for SLAM. Our approach exploits local maps generated using local odometry estimates and compresses them into a BEV density image representation. This allows us to successfully detect map-level closures that can be effectively incorporated into a pose-graph for optimization. We implemented and evaluated our approach on different datasets to show the generalizability of our approach to different scenarios and provided comparisons to other existing techniques. We provide a thorough experimental evaluation supporting all claims made in this thesis. Our conclusion from these experiments is that utilizing local maps for detecting loop closures is a useful direction for building SLAM systems.

6.1 Short summary of key contributions

In conclusion, the key contributions of the place recognition pipeline proposed in this thesis are listed below:

- It is capable of effectively detecting revisited places in a variety of environments using point cloud data obtained from LiDAR sensors, performing at par or better than other state-of-the-art methods for place recognition and loop closing in 3D LiDAR SLAM.
- This approach is agnostic of the sensor scan pattern and works even for unorthodox solid-state LiDARs such as the Livox Avia sensor.
- It proposes one of the first methods that exploit local maps instead of individual scans for performing place recognition.
- We also showcase the ability of our approach to work effectively within a SLAM pipeline for drift correction with pose-graph optimization.

6.2 Open source contributions

We provide an open-source repository of our proposed pipeline, which is efficiently written in C++ and supported with a Python API for easy usage by the community. We also made some modifications to the codebase of some of the baselines we used, resolving a few bugs, enabling them to provide multiple predictions per query scan, as well as providing an easy-to-use and easy-to-install Python bindings for some of them. Likewise, we list below the links to all these repositories for evaluation and use by the robotics community.

- https://gitlab.ipb.uni-bonn.de/ssg1002/map_closures
- https://github.com/saurabh1002/scancontext
- https://github.com/saurabh1002/STD

Bibliography

[1] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool. Speeded-up robust features (SURF). *Journal of Computer Vision and Image Understanding (CVIU)*, 110(3):346–359, 2008.

[2] J. Behley and C. Stachniss. Efficient Surfel-Based SLAM using 3D Laser Range Data in Urban Environments. In *Proc. of Robotics: Science and Systems (RSS)*, 2018.

[3] P.J. Besl and N.D. McKay. A Method for Registration of 3D Shapes. *IEEE Trans. on Pattern Analysis and Machine Intelligence (TPAMI)*, 14(2):239–256, 1992.

[4] M. Bosse and R. Zlot. Place Recognition Using Keypoint Voting in Large 3D Lidar Datasets. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2013.

[5] M. Calonder, V. Lepetit, C. Strecha, and P. Fua. BRIEF: Binary Robust IndependentElementary Features. In *Proc. of the Europ. Conf. on Computer Vision (ECCV)*, 2010.

[6] F. Cao, Y. Zhuang, H. Zhang, and W. Wang. Robust Place Recognition and Loop Closing in Laser-Based SLAM for UGVs in Urban Environments. *IEEE Sensors Journal*, PP:1–1, 03 2018.

[7] X. Chen, T. Läbe, A. Milioto, T. Röhling, O. Vysotska, A. Haag, J. Behley, and C. Stachniss. OverlapNet: Loop Closing for LiDAR-based SLAM. In *Proc. of Robotics: Science and Systems (RSS)*, 2020.

[8] Yang Chen and Gérard Medioni. Object Modelling by Registration of Multiple Range Images. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 1991.

[9] P. Dellenbach, J. Deschaud, B. Jacquet, and F. Goulette. CT-ICP Real-Time Elastic LiDAR Odometry with Loop Closure. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2022.

[10] P. Dellenbach, J.-E. Deschaud, B. Jacquet, and F. Goulette. CT-ICP: Real-time Elastic LiDAR Odometry with Loop Closure. *arXiv preprint*, arXiv:2109.12979, 2021.

[11] K. Ebadi, Y. Chang, M. Palieri, A. Stephens, A. Hatteland, E. Heiden, A. Thakur, N. Funabiki, B. Morrell, S. Wood, L. Carlone, and A. Aghamohammadi. LAMP: Large-Scale Autonomous Mapping and Positioning for Exploration of Perceptually-Degraded Subterranean Environments. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, pages 80–86, 09 2020.

[12] 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*, 24(6):381–395, 1981.

[13] W. Förstner and E. Gülch. A Fast Operator for Detection and Precise Location of Distinct Points, Corners and Centres of Circular Features. In *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, 1987.

[14] A. Frome, D. Huber, R. Kolluri, T. Bülow, and J. Malik. Recognizing Objects in Range Data Using Regional Point Descriptors. In *Proc. of the Europ. Conf. on Computer Vision (ECCV)*, pages 224–237, 2004.

[15] L. Giammarino, I. Aloise, C. Stachniss, and G. Grisetti. Visual Place Recognition using LiDAR Intensity Information. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2021.

[16] Z. Gong, J. Li, Z. Luo, C. Wen, C. Wang, and J. Zelek. Mapping and Semantic Modeling of Underground Parking Lots Using a Backpack LiDAR System. *IEEE Trans. on Intelligent Transportation Systems (T-ITS)*, 22(2):734–746, 2021.

[17] G. Grisetti, R. Kümmerle, C. Stachniss, and W. Burgard. A tutorial on graph-based SLAM. *IEEE Trans. on Intelligent Transportation Systems Magazine*, 2(4):31–43, 2010.

[18] C. Harris and M. Stephens. A combined corner and edge detector. In *Alvey vision conference*, 1988.

[19] L. He, X. Wang, and H. Zhang. M2DP: A Novel 3D Point Cloud Descriptor and Its Application in Loop Closure Detection. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2016.

[20] B. Jiang and S. Shen. Contour Context: Abstract Structural Distribution for 3D LiDAR Loop Detection and Metric Pose Estimation. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2023.

[21] Andrew Johnson and Martial Hebert. Using spin images for effcient object recognition in cluttered 3D scenes. *IEEE Trans. on Pattern Analysis and Machine Intelligence (TPAMI)*, 21(5):433–449, 1999.

[22] M. Jung, W. Yang, D. Lee, H. Gil, G. Kim, and A. Kim. HeLiPR: Heterogeneous LiDAR Dataset for inter-LiDAR Place Recognition under Spatial and Temporal Variations. https://sites.google.com/view/heliprdataset, 2023.

[23] Wolfgang Kabsch. A solution for the best rotation to relate two sets of vectors. *Acta Crystallographica Section A: Crystal Physics, Diffraction, Theoretical and General Crystallography*, 32(5):922–923, 1976.

[24] G. Kim, S. Choi, and A. Kim. Scan Context++: Structural Place Recognition Robust to Rotation and Lateral Variations in Urban Environments. *IEEE Trans. on Robotics (TRO)*, pages 21–27, 2021.

[25] G. Kim, Y.S. Park, Y. Cho, J. Jeong, and A. Kim. Mulran: Multimodal range dataset for urban place recognition. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2020.

[26] Giseop Kim and Ayoung Kim. Scan Context: Egocentric Spatial Descriptor for Place Recognition within 3D Point Cloud Map. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2018.

[27] R. Kümmerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard. g2o: A general framework for graph optimization. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2011.

[28] Y. Li and H. Li. LiDAR-Based Initial Global Localization Using Two-Dimensional (2D) Submap Projection Image (SPI). In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2021.

[29] Jiarong Lin and Fu Zhang. Loam livox: A fast, robust, high-precision LiDAR odometry and mapping package for LiDARs of small FoV. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, pages 3126–3131, 2020.

[30] D.G. Lowe. Object recognition from local scale-invariant features. In *Proc. of the IEEE Intl. Conf. on Computer Vision (ICCV)*, 1999.

[31] S. Lowry, N. Sunderhauf, P. Newman, J.J Leonard, D. Cox, P. Corke, and M.J. Milford. Visual Place Recognition: A Survey. *IEEE Trans. on Robotics (TRO)*, 32(1):1–19, 2016.

[32] L. Luo, S. Cao, B. Han, H. Shen, and J. Li. BVMatch: Lidar-Based Place Recognition Using Bird's-Eye View Images. *IEEE Robotics and Automation Letters (RA-L)*, 6:6076–6083, 2021.

[33] L. Luo, S. Cao, Z. Sheng, and H. Shen. LiDAR-Based Global Localization Using Histogram of Orientations of Principal Normals. *IEEE Trans. on Intelligent Vehicles*, 7(3):771–782, 2022.

[34] M. Magnusson, H. Andreasson, A. Nuechter, Achim, and J. Lilienthal. Appearance-Based Loop Detection from 3D Laser Data Using the Normal Distributions Transform. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2009.

[35] Ellon Mendes, Pierrick Koch, and Simon Lacroix. ICP-based pose-graph SLAM. In *Proc. of the IEEE Intl. Sym. on Safety, Security, and Rescue Robotics (SSRR)*, pages 195–200, 2016.

[36] R. A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. J. Davison, P. Kohli, J. Shotton, S. Hodges, and A. Fitzgibbon. KinectFusion: Real-Time Dense Surface Mapping and Tracking. In *Proc. of the Intl. Symposium on Mixed and Augmented Reality (ISMAR)*, 2011.

[37] C. R. Qi, H. Su, K. Mo, and L. J. Guibas. PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[38] C.R. Qi, K. Yi, H. Su, and L. J. Guibas. PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space. In *Proc. of the Conf. on Neural Information Processing Systems (NeurIPS)*, 2017.

[39] M. Ramezani, Y. Wang, M. Camurri, D. Wisth, M. Mattamala, and M. Fallon. The newer college dataset: Handheld lidar, inertial and vision with ground truth. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2020.

[40] T. Röhling, J. Mack, and D. Schulz. A Fast Histogram-Based Similarity Measure for Detecting Loop Closures in 3-D LIDAR Data. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2015.

[41] E. Rosten and T. Drummond. Machine learning for high-speed corner detection. In *Proc. of the Europ. Conf. on Computer Vision (ECCV)*, 2006.

[42] N. Rottmann, R. Bruder, A. Schweikard, and E. Rueckert. Loop Closure Detection in Closed Environments. In *Proc. of the Europ. Conf. on Mobile Robotics (ECMR)*, pages 1–8, 2019.

[43] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski. Orb: an efficient alternative to sift or surf. In *Proc. of the IEEE Intl. Conf. on Computer Vision (ICCV)*, 2011.

[44] R. Rusu, N. Blodow, Z. Marton, and M. Beetz. Aligning Point Cloud Views using Persistent Feature Histograms. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, pages 3384–3391, 09 2008.

[45] R.B. Rusu, N. Blodow, and M. Beetz. Fast point feature histograms (fpfh) for 3d registration. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2009.

[46] S. Salti, F. Tombari, and L. Stefano. SHOT: Unique signatures of histograms for surface and texture description. *Journal of Computer Vision and Image Understanding (CVIU)*, 125:251–264, 2014.

[47] D. Schlegel and G. Grisetti. HBST: A Hamming Distance embedding Binary Search Tree for Visual Place Recognition. *IEEE Robotics and Automation Letters (RA-L)*, 3:3741–3748, 2018.

[48] A. Segal, D. Haehnel, and S. Thrun. Generalized-ICP. In *Proc. of Robotics: Science and Systems (RSS)*, 2009.

[49] J. Serafin and G. Grisetti. NICP: Dense Normal Based Point Cloud Registration. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2015.

[50] T. Shan, B. Englot, D. Meyers, W. Wang, C. Ratti, and D. Rus. LIO-SAM: Tightly-coupled Lidar Inertial Odometry via Smoothing and Mapping. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2020.

[51] Tixiao Shan and Brendan Englot. LeGO-LOAM: Lightweight and Ground-Optimized Lidar Odometry and Mapping on Variable Terrain. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2018.

[52] J. Shi and C. Tomasi. Good Features to Track. Technical report, Cornell University, 1993.

[53] J. Sivic and A. Zisserman. Video Google: A Text Retrieval Approach to Object Matching in Videos. In *Proc. of the IEEE Intl. Conf. on Computer Vision (ICCV)*, 2003.

[54] B. Steder, G. Grisetti, and W. Burgard. Robust Place Recognition for 3D Range Data Based on Point Features. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2010.

[55] B. Steder, M. Ruhnke, S. Grzonka, and W. Burgard. Place Recognition in 3D Scans Using a Combination of Bag of Words and Point Feature Based Relative Pose Estimation. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2011.

[56] B. Steder, R.B. Rusu, K. Konolige, and W. Burgard. NARF: 3D range image features for object recognition. In *Workshop on Defining and Solving Realistic Perception Problems in Personal Robotics at the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2010.

[57] F. Tombari, S. Salti, and L. Di Stefano. Unique Shape Context for 3d Data Description. In *Proceedings of the ACM Workshop on 3D Object Retrieval*, page 57–62, 2010.

[58] S. Umeyama. Least-squares estimation of transformation parameters between two point patterns. *IEEE Trans. on Pattern Analysis and Machine Intelligence (TPAMI)*, 13(4):376–380, 1991.

[59] M.K. Vijaymeena and K. Kavitha. A Survey on Similarity Measures in Text Mining. *Machine Learning and Applications: An International Journal*, 3(1):19–28, 2016.

[60] Ignacio Vizzo, Tiziano Guadagnino, Jens Behley, and Cyrill Stachniss. VDB-Fusion: Flexible and Efficient TSDF Integration of Range Sensor Data. *Sensors*, 22(3):1296, 2022.

[61] Ignacio Vizzo, Tiziano Guadagnino, Benedikt Mersch, Louis Wiesmann, Jens Behley, and Cyrill Stachniss. KISS-ICP: In Defense of Point-to-Point ICP – Simple, Accurate, and Robust Registration If Done the Right Way. *IEEE Robotics and Automation Letters (RA-L)*, 8(2):1029–1036, 2023.

[62] J. Yang, Q. Zhang, Y. Xiao, and Z. Cao. TOLDI: An effective and robust approach for 3D local shape description. *Pattern Recognition*, 65:175–187, 2017.

[63] C. Yuan, J. Lin, Z. Zou, X. Hong, and F. Zhang. STD: Stable Triangle Descriptor for 3D place recognition. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2023.

[64] J. Zhang and S. Singh. LOAM: Lidar Odometry and Mapping in Real-time. In *Proc. of Robotics: Science and Systems (RSS)*, 2014.

[65] Y. Zhang, P. Shi, and J. Li. LiDAR-Based Place Recognition For Autonomous Driving: A Survey. *arXiv preprint*, arXiv:2306.10561, 2023.

[66] Bo Zhou, Yi He, Kun Qian, Xudong Ma, and Xiaomao Li. S4-slam: A real-time 3d lidar slam system for ground/watersurface multi-scene outdoor applications. *Autonomous Robots*, 45(1):77–98, 2021.

[67] Y. Zhuang, N Jiang, H. Hu, and F. Yan. 3-D-Laser-Based Scene Measurement and Place Recognition for Mobile Robots in Dynamic Indoor Environments. *IEEE Transactions on Instrumentation and Measurement*, 62:438–450, 02 2013.

List of Figures

List of Tables

List of Algorithms

Appendix A

Paper

Effectively Detecting Loop Closures using Point Cloud Density Maps

Saurabh Gupta Tiziano Guadagnino Benedikt Mersch Ignacio Vizzo Cyrill Stachniss

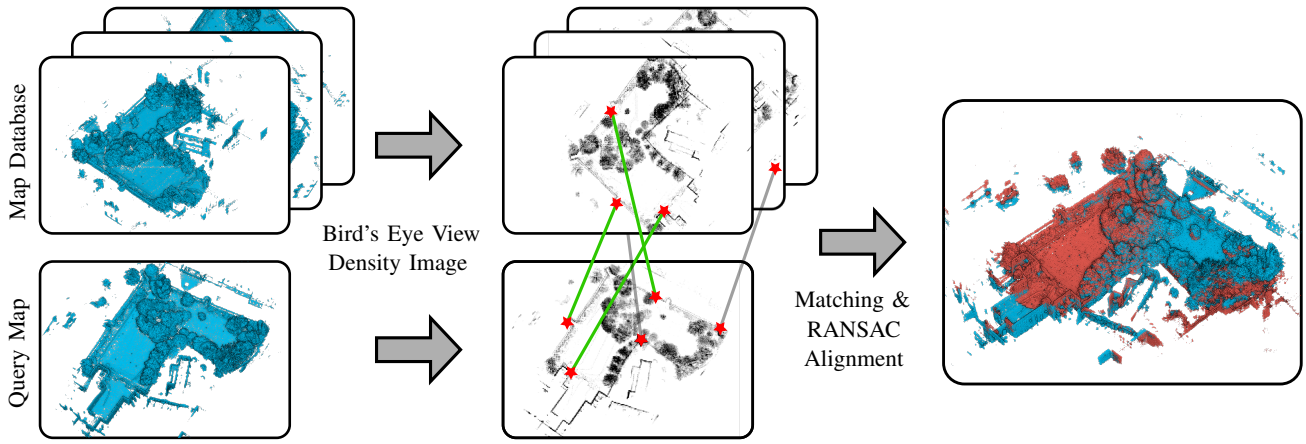


Fig. 1: A database over previously seen local maps is queried for matching a new local map (left). We perform feature matching on features computed on the bird eye view density images of these maps to detect closures (center). Finally, a RANSAC-based 2D rigid body alignment is used to obtain the 2D pose estimate between loop closed local maps (right).

***Abstract*— The ability to detect loop closures plays an important role in any SLAM system. Loop closures allow for correcting the drifting pose estimates from a sensor odometry pipeline. In this paper, we address the problem of effectively detecting loop closures in LiDAR SLAM systems in a variety of environments, with longer lengths of sequences and agnostic of the scanning pattern of the sensor. While many approaches for loop closures using 3D LiDAR sensors rely on individual scans, we propose the usage of local maps generated from locally consistent odometry estimates. Several of the recent approaches compute the maximum elevation map on a bird eye view projection of point clouds to compute feature descriptors. In contrast, we use a density image bird eye view representation which is robust to viewpoint changes. The utilization of dense local maps allows us to reduce the complexity of features describing these maps, as well as the size of the database required to store these features over a long sequence. This yields a real-time application of our approach for a typical robotic 3D LiDAR sensor. We perform extensive experiments to evaluate our approach against other state-of-the-art approaches and show the benefits of our proposed approach.**

I. Introduction

Autonomous mobile robots are required to have an accurate estimate of their ego motion while traversing through previously unseen environments. This task forms a core element of a simultaneous localization and mapping (SLAM) system. Sequential odometry estimates often suffer from drift, especially over long traversals. This drift can occur due to the inherent noise in the robot motion and sensor data, the presence of dynamics in the environment, as well as non-trivial data association problems. When a robot revisits a known place, its own belief may indicate a completely different location because of such drift. To overcome this problem, the detection of revisited locations, also called loop closures, is of utmost importance for any SLAM pipeline. Robust loop closure detection is a necessary prerequisite for drift correction through the integration of pose constraints often done in the underlying pose-graph [5].

Detecting previously seen places requires crafting an as-unique-as-possible description of the sensed environment. At the same time, this description should be invariant to changes in the 3D viewpoint, scan pattern of LiDARs, and dynamics in the sensed environment. Furthermore, to incorporate loop closures in a SLAM system, this description should preferably capture the geometry of the scene to enable relative pose estimation.

Traditional methods [4, 6, 8, 13, 15] rely on computing such a description for each LiDAR scan. Subsequently, a matching algorithm is used to compare these descriptions and detect loop closures. The bird eye view (BEV) projection of point clouds is one popular approach [13, 15, 15] towards detecting loop closures, providing a compact 2D representation for faster feature detection and matching.

The main contribution of this paper is a simple yet effective approach, see Fig. 1, for detecting loop closures in online LiDAR SLAM systems. Local maps generated using

All authors are with the University of Bonn, Germany. Cyrill Stachniss is additionally with the Department of Engineering Science at the University of Oxford, UK, and with the Lamarr Institute for Machine Learning and Artificial Intelligence, Germany.

This work has partially been funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy, EXC-2070 – 390732324 – PhenoRob, by the European Union's Horizon 2020 research and innovation program under grant agreement No 101017008 (Harmony) and the Horizon Europe research and innovation program under grant agreement No 101070405 (DigiForest).

locally consistent sensor odometry estimates form the core element of our approach. We show that a simple feature detection and matching pipeline exploiting density images from the BEV projections of the local maps can effectively detect loop closures. We also provide a validation step to perform a geometric verification of detected loop closures and provide a complete 2D rigid body transform between the loop-closed local maps. Our approach (i) can effectively detect loop closures between two temporally separated local maps in a variety of environments (ii) provides a 2D rigid body transform between the detected loop closures and (iii) is agnostic of the sensor scan pattern by exploiting local maps. We will open-source the code for our approach upon acceptance of the manuscript.

II. Related Work

The task of loop closing in the domain of 3D ranging sensors has been a topic of discussion in many recent works, as laid out thoroughly by Zhang et al. [32]. Many approaches [1, 8, 22, 23, 30] propose 3D point features, inspired by image-based localization. They discretize the neighborhood around a point into a geometrical grid and compute the local descriptor based on the height, density, distance, or angle of the points contained within. These methods are partially affected by the sparsity of typical LiDAR point clouds and are not viewpoint-invariant.

Learning-based approaches such as SegMatch [3] extract semantic information from point clouds to detect loop closures. OverlapNet [2] computes overlap between the range-image representation of two scans using a deep neural network. Such approaches though require a training step along with GPU acceleration.

Several approaches project the 3D point cloud into 2D in the form of a spherical view range image [25, 26] or a BEV image [15, 16, 17], and then use established techniques in image processing to compute descriptors. Some methods [6, 11, 13, 19] use BEV projections to define a global descriptor capturing holistic information from each scan.

The popular ScanContext [13] proposes a polar elevation map, centered at the local reference frame of the LiDAR scan. This allows the computation of a rotation-invariant global descriptor but loses translational invariance. They try to overcome this by manually shifting the reference frame of scans by the expected translation during revisits. ScanContext++ [11] computes Cartesian elevation maps along with polar elevation maps. However, most of these global descriptors assume a spherical scan pattern of conventional LiDARs and, as such, they are not well suited for other scan patterns such as the Livox LiDAR.

ContourContext [7] generates BEV projections at predefined heights and clusters (contours) the projected points within these elevation maps. Statistics computed from these contours are used to define a global descriptor for matching. Another recent work by Yuan et al. [31] proposes using stable triangle descriptors (STD) computed using keypoints within a local neighborhood. These keypoints too, are based on a local BEV projection of the points within a voxel at the boundary of a planar region. They accumulate a fixed number of consecutive scans for computing these descriptors to have a sufficient point cloud density. However, these methods use the elevation map as the 2D representation, which is sensitive to the sensor viewpoint. BVMatch [16] proposes using a density map representation but requires training a bag-of-words model on features detected from individual scans to detect loop closures. Thus, it is not ideal for being used for online SLAM in novel environments.

Among the 2D projection based approaches, BEV projections [11, 13, 15, 16, 17] seem to be preferred over the range image based approaches [2, 18]. A key reason is that the range images lose depth information while projecting spherically. On the other hand, BEV projections preserve 2D geometry along the local XY plane, which is crucial in the case of most autonomous ground robots/vehicles.

Our proposed method makes use of local maps as the primary representation, where we accumulate scans based on travel distance criteria, rather than naively accumulating a fixed number of consecutive scans. Such maps allow us the benefit of having a dense collection of points representing the environment, which show a stronger invariance towards viewpoint changes and sensor scan pattern. We further make use of 2D density images [16] of these local maps instead of elevation images, to increase robustness against viewpoint changes and noise in the sensing process. Utilizing existing feature descriptors from the computer vision community [21], we compute local features on these density maps and store them in an efficient binary search tree [24] for matching with query maps. This allows us to provide a complete 2D relative pose estimate between detected map-level closures, which aids the final 3D registration step involved in basically any geometric SLAM pipeline.

III. Our approach to Loop Closure Detection

Our proposed approach detects loop closures at a local map level and provides a complete 2D initial guess on the relative pose between matched maps. First, we generate local maps using odometry estimates and represent them as a density image as explained in Sec. III-A and Sec. III-B. Then, we compute local features on these images and create a database for matching with query maps, see Sec. III-C. Finally, in Sec. III-D we show the geometrical verification and alignment of the loop closures. See Fig. 1 for the overall pipeline.

A. Local Maps

Our approach for detecting loop closures in SLAM uses local maps. We build a local map by accumulating consecutive scans with their relative pose estimates obtained from an ICP odometry pipeline. In particular, we use the KISS-ICP [29] odometry pipeline, a state-of-the-art approach for LiDAR odometry with strong open-source community support.

To generate these local maps, we consider a set of n consecutive 3D point clouds $\{P_i, \ldots, P_{i+n-1}\}$ in their local reference frame, and transform them into the reference frame

of the i^{th} scan $\{{}^{i}\mathsf{P}_i, \ldots, {}^{i}\mathsf{P}_{i+n-1}\}$ as follows:

$$ {}^{i}\mathsf{P}_{i+n-1} = \mathsf{T}_i^{-1}\mathsf{T}_{i+n-1}\mathsf{P}_{i+n-1}, \tag{1} $$

where $\mathsf{T}_i \in \mathbb{SE}(3)$ is the odometry estimate of the pose connected to the i^{th} scan. We aggregate all these scans into a local map $\mathcal{M}_i$ using a voxel grid with a resolution of ν_{map} meters per voxel. The local map $\mathcal{M}_i$ is centered at the frame T_i, the first scan of the map. To ensure a uniform density of points in the voxel grid cells, we retain a maximum of 20 points per cell. The size of this map $\mathcal{M}_i$ is decided based on the distance traveled by the sensing platform. We accumulate n consecutive scans until the travel distance $\|\boldsymbol{t}_{i+n-1} - \boldsymbol{t}_i\|_2$ exceeds a threshold of τ_m meters, where $\boldsymbol{t}_i \in \mathbb{R}^3$ is the translational component of the odometry pose estimate T_i.

B. Density Images

The BEV projection of point clouds is a widely used approach in place recognition using LiDARs [11, 13, 15, 16, 17]. Such a projection preserves local 2D geometry, and at the same time also reduces the computational complexity of algorithms due to a reduced dimension to process. However, a lot of the traditional BEV projection approaches [11, 13, 15, 17] store the maximum height (elevation map) of the points in each bin, not point densities. The elevation map is sensitive to the orientation of the sensor as the maximum height recorded varies with the distance between the scanner and the object. The density of points scanned on a surface, on the other hand, is less sensitive to viewpoint changes.

For a local map $\mathcal{M}_i$, we project all the points to the local ground plane through an orthographic BEV projection. When the reference frame T_i of the local map $\mathcal{M}_i$ has its xy-plane parallel to the ground, as is the case for ground robots/vehicles, this would correspond to simply dropping the z-coordinate of all the individual points. For other robots such as UAVs, we require a gravity vector that can be directly obtained from an IMU. The BEV projection gives us a set of points B_i in $\mathbb{R}^2$ bounded by a rectangular window from $[x_i^l,\, y_i^l]^\mathsf{T}$ to $[x_i^u,\, y_i^u]^\mathsf{T}$ meters where,

$$ \begin{bmatrix} x_i^u \\ y_i^u \end{bmatrix} = \max_{x,y} \mathsf{B}_i \;;\; \begin{bmatrix} x_i^l \\ y_i^l \end{bmatrix} = \min_{x,y} \mathsf{B}_i. \tag{2} $$

We further discretize B_i into a 2D Cartesian grid $\mathsf{N}_i(u,v) \in \mathbb{N}_0^{W_i \times H_i}$ of resolution ν_{res} meters per cell where,

$$ W_i = \left\lceil \frac{x_i^u - x_i^l}{\nu_{res}} \right\rceil \;;\; H_i = \left\lceil \frac{y_i^u - y_i^l}{\nu_{res}} \right\rceil. \tag{3} $$

Each cell in this grid $N(u,v)$ stores the number of points contained in that cell after discretization. The grayscale density image $\mathsf{I}_i(u,v)$ of the local map $\mathcal{M}_i$ is then defined as,

$$ \mathsf{I}_i(u,v) = \frac{N(u,v) - N_{min}}{N_{max} - N_{min}} \in \mathbb{R}^{W_i \times H_i}, \tag{4} $$

$$ N_{max} = \max_{u,v} N_i(u,v) \;;\; N_{min} = \min_{u,v} N_i(u,v). \tag{5} $$

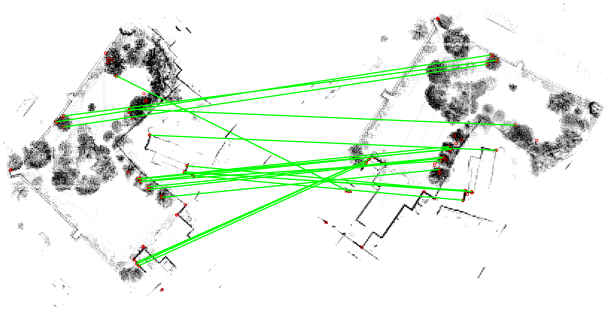


Fig. 2: ORB features (red) and matches (green) between two density images from the same location, revisited from a different viewpoint.

We set all the image pixels (u,v) with density lower than $5\,\%$ of the maximum value to be zero. This helps to reduce the noise in the local maps and to remove the ground plane, which is typically insignificant for feature detection. These density images capture 2D floorplan-like information from the local maps, like building facades, and trees as can be seen in the density maps in Fig. 2. These rigid structures provide reliable features to track for place recognition.

C. Feature Detection and Database Creation

Facilitated by the accumulation of information in the form of the local map density images I_j, we utilize the well-known ORB [21] feature descriptors to capture corner-like features from them. To speed up computation, we compute ORB features without scale-invariance. We can do this since the density image is an orthographic projection of the 3D world and has no scale ambiguity as regular camera images. Fig. 2 shows in red the ORB features computed on sample density images.

The binary domain of ORB descriptors furthermore allows for efficient feature storage and matching by leveraging the Hamming distance metric. In particular, we employ the Hamming distance embedding binary search tree (HBST) [24] to store the set of feature descriptors D_i obtained from each density image $\mathsf{I}_i(u,v)$ along with the corresponding map index i. The depth of the HBST is bounded by the number of bits in the binary descriptor (256). This means that a query descriptor would have a maximum of 256 bitwise comparisons with the tree's nodes before it terminates in one of the leaf nodes. Furthermore, the capacity of each leaf node is limited to a maximum of 100 descriptors per leaf node. Fixing these two design parameters imposes an upper bound on the computational time of the feature-matching process while not introducing any restriction in the practical use of our approach. Although the binary tree is not a complete data structure for a nearest-neighbor computation, it provides with a good initial guess on the feature matches for subsequent geometric verification and refinement.

After obtaining a new set of descriptors D_q from the query local map's density image I_q, we find the nearest match for each descriptor in D_q from the binary tree database. We use a threshold of 50 bits on the Hamming distance between two ORB descriptors to call them a match. Since the binary tree stores descriptors along with an index i of the

corresponding map they were obtained from, we cast a vote over the map indices for each such match. Once all the query descriptors have been processed, we select the reference maps corresponding to top-N votes from this voting scheme. We set N to be equal to half the number of local maps in the database at any time. This dynamic factor allows us to find multiple potential loop closures at the same physical location, the chances of which increase with the number of local maps in the database. As a result, we obtain a list of feature matches between the query map and the reference maps in the database, on which we perform a geometrical verification. Fig. 2 visualizes the corresponding set of feature matches between two density images.

D. Loop Detection and Map Alignment

The geometrical verification step involves a 2D alignment of the matched features. This implies computing a 2D rigid body transformation (3 degrees of freedom) that aligns the matched features from the binary tree in the best way possible based on a distance metric. This is similar to the image-alignment problem, but limited to an $\mathbb{SE}(2)$ transform instead of a homography. We use a Random Sampling Consensus (RANSAC)-based alignment strategy to reject outlier associations due to the incompleteness of the binary tree-based matching. This verification step is performed for matches between the query map and each of the top-N reference maps separately.

We only need 2 sets of matching keypoints to compute a rigid body alignment in 2D. Using this fact, we design a RANSAC scheme, randomly drawing 2 feature matches from the entire set of feature matches between a query (I_q) and a reference (I_p) density image. We compute the relative 2D alignment between them using the Kabsch-Umeyama [10, 27] algorithm. It provides us with a rotation matrix $\boldsymbol{R} \in \mathbb{SO}(2)$ and a translation vector $\boldsymbol{t} \in \mathbb{R}^2$, which can be composed into a homogenous transformation ${}^{q}\mathsf{T}_{p} \in \mathbb{SE}(2)$. Using this 2D alignment, we transform all the features from I_p to the reference frame of I_q.

For verification within RANSAC, we compute the point-wise error between the matching features, in terms of the Euclidean distance between the matched point sets. Matches with distances larger than $1.5\,\mathrm{m}$ ($3\,\mathrm{pixels}$ for $\nu_{res} = 0.5\,\mathrm{m}$) are considered as outliers. The RANSAC scheme terminates after a fixed number of iterations or if a high quality solution is found, i.e., in case more than 30 inliers are obtained.

We define a threshold γ for the minimum number of inliers required from the RANSAC alignment to conclude whether two local maps belong to the same location. The associated 2D transform between the two density images is already a good initial estimate for the complete 3D transform between local maps in the case of ground robots/vehicles. Note that we scale the translation vector by the voxel size ($\nu_{res}\boldsymbol{t}$) to undo the effect of the discretization while generating the density images. This initial alignment can later be used as an initial guess for a fine point cloud registration in 3D.

IV. Experimental Evaluation

This work provides a pipeline to compute loop closures for SLAM using local maps. We present our experiments to show the capabilities of our method. The results of our experiments also support our key claims, which are: (i) we can effectively detect loop closures between two temporally separated local maps in a variety of environments, (ii) we provide a 2D rigid body transform between detected loop closures, and (iii) our approach is agnostic of the sensor scan pattern by exploiting local maps.

A. Evaluation Criteria

To assess the performance of loop closure detection, we identify scan-wise reference closures based on the volumetric overlap of measured point clouds. The community lacks consensus on what a true loop closure is, as highlighted by Jiang et al. [7]. Related works [7, 13, 31] usually consider two scans as a true positive closure if the distance between the corresponding reference locations is below a certain threshold. This implies a potentially wrong assumption that a sensor observes the same objects when being at a similar location. This does not hold when other objects occlude the previously seen area, when the sensor has a limited field of view, or if the orientation differs strongly.

Instead, we propose to use the volumetric overlap of scans to decide if a loop should be closed. We argue that if the point clouds from two different points in time overlap, we can find their relative pose and integrate it into a pose-graph. The first step of our reference identification is to sample the reference trajectory at equidistant locations ($2\,\mathrm{m}$) to reduce the number of candidates. Next, we accumulate the registered scans between two consecutive key locations to have a dense representation of the local environment. We find all possible pairs of key locations within a distance of the sensor's maximum range with a minimum travel distance. Further, we voxelize the accumulated point clouds of both keyframes. We use a voxel size of $0.5\,\mathrm{m}$. Finally, we compute the overlap $o \in [0, 1]$ between two non-empty voxel grids $\mathcal{V}_i$ and $\mathcal{V}_j$ using the overlap coefficient [28] as in Eq. (6), also called the Szymkiewicz-Simpson coefficient:

$$o\left(\mathcal{V}_i, \mathcal{V}_j\right) = \frac{|\mathcal{V}_i \cap \mathcal{V}_j|}{\min\left(|\mathcal{V}_i|, |\mathcal{V}_j|\right)}. \tag{6}$$

We consider two keyframes to be a loop closure if their overlap $o > 0.5$. To also evaluate and compare predicted closures between local maps, we need to compute scan index closures eventually, since other benchmarks provide closures on individual scans. To achieve this, we apply the 2D pose correction ${}^{q}\mathsf{T}_{p}$ between maps to the individual scans comprising these maps, transforming them to a common reference frame. Then, we compare all the pairs of scan-level poses between the two maps, computing the pairwise distance between them. Finally, we use a distance threshold τ_d to classify whether two scan indices are predicted to be a loop closure or not.

TABLE I: Precision (P), Recall (R) and F1 scores of state-of-the-art baselines and our approach, for all datasets.

Datasets	MK03			MR02			MS01			NCD			HT01		
	P	R	F1	P	R	F1	P	R	F1	P	R	F1	P	R	F1
SC [13]	**0.890**	0.450	0.598	0.259	0.071	0.111	0.027	0.126	0.045	0.415	0.007	0.013	N/A	N/A	N/A
CC [7]	0.798	0.424	0.554	0.695	0.080	0.144	0.244	0.076	0.116	0.226	0.030	0.053	N/A	N/A	N/A
STD [31]	0.679	0.435	0.530	0.491	0.042	0.078	-	-	-	-	-	-	-	-	-
Ours	0.730	**0.668**	**0.698**	**0.939**	**0.731**	**0.822**	**0.983**	**0.298**	**0.458**	0.927	**0.068**	**0.127**	**0.473**	**0.669**	**0.554**
Ours (E)	0.759	0.606	0.674	0.639	0.118	0.199	0.978	0.296	0.454	**0.933**	0.043	0.083	-	-	-

B. Experimental Setup

We evaluate our work on the MulRan [12], Newer College [20], and HeLiPR [9] datasets. From the MulRan dataset, we use the KAIST03 (MK03), Riverside02 (MR02), and Sejong01 (MS01) sequences, which contain loop closures from varying viewpoints. They also cover a variety of driving scenarios like city and highway scenes, and longer trajectories. The Newer College 01_short_experiment (NCD) sequence provides data recorded from a hand-held scanning platform in contrast to the car-mounted scanner in MulRan. For HeLiPR, we use the Town01 (HT01) sequence with scans from a Livox LiDAR with a small field of view and a rather unusual scanning pattern.

We compare our method with three baselines: ScanContext-10 (SC) [13], ContourContext (CC) [7], and STD [31]. SC is widely used in the context of SLAM and also among the current state-of-the-art methods. CC and STD are recently published approaches for loop closures using LiDARs. All these baselines provide a publicly available working code for evaluation. We modify them to provide multiple closure candidates for each query scan instead of only the best candidate, keeping all other parameters as default. This ensures a fair comparison, as our approach also provides multiple closure candidates. The STD pipeline proposes accumulation of 10 consecutive scans into a local map, so we utilize the same method as in our approach, to obtain scan index closures from map index closures, with $\tau_d = 6\,\text{m}$.

For our approach, we set the maximum range of the scanner r as well as the local map truncation distance τ_m to $100\,\text{m}$. For the KISS-ICP odometry, we use the default parameters provided. We use a voxel size of $\nu_{map} = 1.0\,\text{m}$ for the voxel grid used to generate the local maps, and a resolution of $\nu_{res} = 0.5\,\text{m}$ for the density images. For ORB feature descriptors we use the default parameters provided, only disabling the parameters controlling scale invariance. We require a minimum of 25 feature matches from the HBST within a Hamming distance of $50\,\text{bits}$. All other parameters of the binary tree are set to the defaults. Finally, we predict two local maps to be loop closures based on the number of inliers obtained from the RANSAC-based alignment, the threshold for which is set to be $\gamma = 10$. Finally, to compute scan index loop closures for quantitative evaluation, we use a distance threshold of $\tau_d = 6\,\text{m}$. These parameters are maintained across all datasets except for the HeLiPR Town01 dataset, for which we use a maximum range of $r = 50\,\text{m}$, $\nu_{map} = 0.5\,\text{m}$, and a minimum of 10 feature matches from the HBST. This is to account for the small field-of-view and unusual scan pattern of the Livox LiDAR.

C. Performance Evaluation

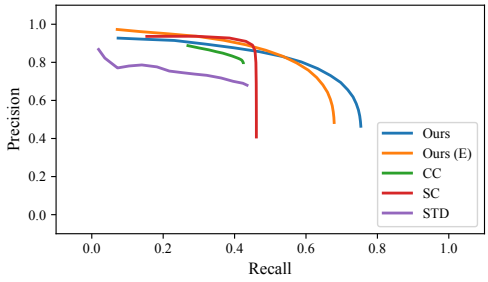
The performance of our approach is shown in Tab. I, as a comparison of precision, recall, and F1 scores between each of the baselines and our approach, for all the aforementioned datasets. Note that we report the precision and recall scores corresponding to the best achievable F1 scores for each of the approaches, including ours. The F1 score, being the geometric mean of precision and recall, is a suitable statistic for comparison. It can be observed from Tab. I that our approach has the best F1 score among all baselines for all datasets. Furthermore, we achieve these F1 scores at a significantly high precision as well, which is important for incorporating loop closures into a SLAM pipeline.

One can observe comparable performance across all approaches for the MulRan KAIST03 (MK03) dataset since it is a regular city-like environment about $6.1\,\text{km}$ long. However, a significant performance gap can be seen between our approach and other baselines for the MulRan Sejong01 (MS01) sequence which is a $23.4\,\text{km}$ long highway sequence, with sparsely distributed features as well as dynamic obstacles. The STD [31] baseline even fails to run over such long sequences having more than 15000 scans.

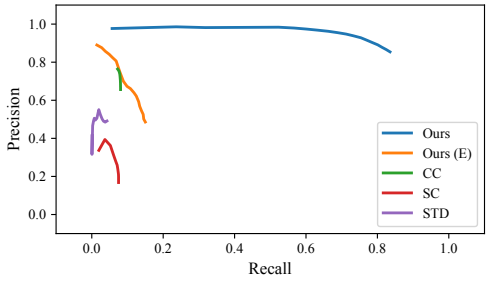
In Fig. 3, we also provide the precision-recall curves for the MulRan KAIST03 and Riverside02 sequences, which highlight the range of performance offered by each of the approaches. Our approach not only provides the best F1 score for these datasets as highlighted in Tab. I, but also can be adapted to provide either high precision or high recall based on the distance threshold parameter (τ_d).

We also ablate our choice of density images over elevation maps by modifying our pipeline to use elevation images of the local maps. We present the results for the same in the last row (Ours(E)) of Tab. I. It can be seen that the elevation image-based approach performs at par or worse than the density image-based approach that we propose. Also, it completely fails on challenging datasets like the HeLiPR Town01 (HT01).

Overall, this evaluation highlights our approach's ability to detect loop closures in a variety of environments. Furthermore, it is an indirect evaluation of our 2D pose estimates between detected loops, as we use these estimates when computing the scan index level closures.



(a) MulRan KAIST03



(b) MulRan Riverside02

Fig. 3: The precision-recall curves for our approach and other baselines.

D. Livox LiDAR Scan Pattern

The HeLiPR dataset provides scans recorded from a Livox Horizon LiDAR, which has a very unusual scanning pattern as against traditional LiDARs, with a higher density but small field-of-view. This poses a significant challenge in detecting loop closures. However, our approach is capable of overcoming challenges, primarily due to the use of local maps. As can be seen from the last column (HT01) in Tab. I, we achieve an F1 score of 0.554 on this dataset, also being the only method to even work on this dataset. STD [31], even though capable of processing Livox scans due to aggregation of a fixed number of scans, fails on this dataset due to the large size of the HT01 sequence.

E. Offline Optimization with Detected Loop Closures

In this final experiment, we showcase the ability of our approach to perform drift correction in a SLAM pipeline. We perform an offline pose-graph optimization including all the detected map-level loops. A fine ICP registration is performed on the detected loops between local maps with the initial guess provided by our pipeline. The refined 3D pose estimate between the local maps is then incorporated as a constraint in the pose-graph along with the odometry constraints. We use the open-sourced g2o [14] pipeline.

TABLE II: Absolute Pose Error (APE) in translation and rotation, with (w) and without (w/o) loop closures.

		APE tra. (m) (rms)		APE rot. (rms)	
Datasets	# Closures	w	w/o	w	w/o
MK03	73	3.04	14.64	0.05	0.09
MR02	33	12.70	32.58	0.08	0.11
MS01	2	206.6	1248.6	0.48	0.63
NCD	7	0.61	0.62	0.03	0.03
HT01	2	12.92	18.48	0.14	0.15

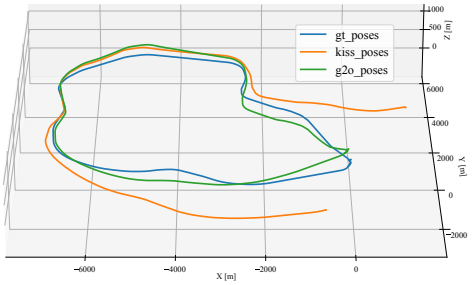


Fig. 4: MulRan Sejong01 sequence poses before and after pose-graph optimization.

In Fig. 4, we show a comparison of pose estimates for the MulRan Sejong01 dataset, which is a $23.4\,\text{km}$ long highway sequence, with an odometry drift in the order of a few thousand meters. It can be seen that the inclusion of our detected loop closures reduced the drift by a significant amount. This is further confirmed by the values of absolute pose error (APE) in translation and rotation with respect to the ground-truth poses, as shown in Tab. II. We provide the APE values with and without the inclusion of loop closures for all the datasets to highlight the significant impact our approach for detecting loop closures can have over the odometry estimation. Furthermore, we achieve this improvement in APE with a significantly lower number of loop closures as can be seen in Tab. II, allowing for faster optimization of the pose-graph.

In summary, our evaluation suggests that our method can effectively detect loop closures with local maps, with a better or at par performance compared to other approaches, on a variety of datasets. The accumulation of scans into local maps makes our method agnostic of the sensor scan pattern and density. Finally, we also demonstrated that our detected loops provide a significant drift reduction when incorporated into a pose-graph. Thus, we supported all our claims with this experimental evaluation.

V. Conclusion

In this paper, we presented a novel approach to computing loop closures for SLAM. Our approach exploits local maps generated using local odometry estimates and compresses them into a BEV density image representation. This allows us to successfully detect map-level closures that can be effectively incorporated into a pose-graph for optimization. We implemented and evaluated our approach on different datasets to show the generalizability of our approach to different scenarios and provided comparisons to other existing techniques. We provide a thorough experimental evaluation supporting all claims made in this paper. Our conclusion from these experiments is that utilizing local maps for detecting loop closures is a useful direction for building SLAM systems.

REFERENCES

[1] M. Bosse and R. Zlot. Place Recognition Using Keypoint Voting in Large 3D Lidar Datasets. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2013.

[2] X. Chen, T. Läbe, A. Milioto, T. Röhling, O. Vysotska, A. Haag, J. Behley, and C. Stachniss. OverlapNet: Loop Closing for LiDAR-based SLAM. In *Proc. of Robotics: Science and Systems (RSS)*, 2020.

[3] R. Dubé, D. Dugas, E. Stumm, J. Nieto, R. Siegwart, and C. Lerma. SegMatch: Segment Based Place Recognition in 3D Point Clouds. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2017.

[4] L. Giammarino, I. Aloise, C. Stachniss, and G. Grisetti. Visual Place Recognition using LiDAR Intensity Information. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2021.

[5] G. Grisetti, R. Kümmerle, C. Stachniss, and W. Burgard. A tutorial on graph-based SLAM. *IEEE Trans. on Intelligent Transportation Systems Magazine*, 2(4):31–43, 2010.

[6] L. He, X. Wang, and H. Zhang. M2DP: A Novel 3D Point Cloud Descriptor and Its Application in Loop Closure Detection. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2016.

[7] B. Jiang and S. Shen. Contour Context: Abstract Structural Distribution for 3D LiDAR Loop Detection and Metric Pose Estimation. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2023.

[8] A. Johnson and M. Hebert. Using spin images for effcient object recognition in cluttered 3D scenes. *IEEE Trans. on Pattern Analysis and Machine Intelligence (TPAMI)*, 21(5):433–449, 1999.

[9] M. Jung, W. Yang, D. Lee, H. Gil, G. Kim, and A. Kim. HeLiPR: Heterogeneous LiDAR Dataset for inter-LiDAR Place Recognition under Spatial and Temporal Variations. https://sites.google.com/view/heliprdataset, 2023.

[10] W. Kabsch. A solution for the best rotation to relate two sets of vectors. *Acta Crystallographica Section A: Crystal Physics, Diffraction, Theoretical and General Crystallography*, 32(5):922–923, 1976.

[11] G. Kim, S. Choi, and A. Kim. Scan Context++: Structural Place Recognition Robust to Rotation and Lateral Variations in Urban Environments. *IEEE Trans. on Robotics (TRO)*, pages 21–27, 2021.

[12] G. Kim, Y. Park, Y. Cho, J. Jeong, and A. Kim. Mulran: Multimodal range dataset for urban place recognition. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2020.

[13] G. Kim and A. Kim. Scan Context: Egocentric Spatial Descriptor for Place Recognition within 3D Point Cloud Map. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2018.

[14] R. Kümmerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard. g2o: A general framework for graph optimization. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2011.

[15] Y. Li and H. Li. LiDAR-Based Initial Global Localization Using Two-Dimensional (2D) Submap Projection Image (SPI). In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2021.

[16] L. Luo, S. Cao, B. Han, H. Shen, and J. Li. BVMatch: Lidar-Based Place Recognition Using Bird's-Eye View Images. *IEEE Robotics and Automation Letters (RA-L)*, 6:6076–6083, 2021.

[17] L. Luo, S. Cao, Z. Sheng, and H. Shen. LiDAR-Based Global Localization Using Histogram of Orientations of Principal Normals. *IEEE Trans. on Intelligent Vehicles*, 7(3):771–782, 2022.

[18] J. Ma, J. Zhang, J. Xu, R. Ai, W. Gu, and X. Chen. OverlapTransformer: An Efficient and Yaw-Angle-Invariant Transformer Network for LiDAR-Based Place Recognition. *IEEE Robotics and Automation Letters (RA-L)*, 7(3):6958–6965, 2022.

[19] M. Magnusson, H. Andreasson, A. Nuechter, Achim, and J. Lilienthal. Appearance-Based Loop Detection from 3D Laser Data Using the Normal Distributions Transform. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2009.

[20] M. Ramezani, Y. Wang, M. Camurri, D. Wisth, M. Mattamala, and M. Fallon. The newer college dataset: Handheld lidar, inertial and vision with ground truth. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2020.

[21] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski. Orb: an efficient alternative to sift or surf. In *Proc. of the IEEE Intl. Conf. on Computer Vision (ICCV)*, 2011.

[22] R. Rusu, N. Blodow, and M. Beetz. Fast point feature histograms (fpfh) for 3d registration. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2009.

[23] S. Salti, F. Tombari, and L. Stefano. SHOT: Unique signatures of histograms for surface and texture description. *Journal of Computer Vision and Image Understanding (CVIU)*, 125:251–264, 2014.

[24] D. Schlegel and G. Grisetti. HBST: A Hamming Distance embedding Binary Search Tree for Visual Place Recognition. *IEEE Robotics and Automation Letters (RA-L)*, 3:3741–3748, 2018.

[25] B. Steder, G. Grisetti, and W. Burgard. Robust Place Recognition for 3D Range Data Based on Point Features. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2010.

[26] B. Steder, M. Ruhnke, S. Grzonka, and W. Burgard. Place Recognition in 3D Scans Using a Combination of Bag of Words and Point Feature Based Relative Pose Estimation. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2011.

[27] S. Umeyama. Least-squares estimation of transformation parameters between two point patterns. *IEEE Trans. on Pattern Analysis and Machine Intelligence (TPAMI)*, 13(4):376–380, 1991.

[28] M. Vijaymeena and K. Kavitha. A Survey on Similarity Measures in Text Mining. *Machine Learning and Applications: An International Journal*, 3(1):19–28, 2016.

[29] I. Vizzo, T. Guadagnino, B. Mersch, L. Wiesmann, J. Behley, and C. Stachniss. KISS-ICP: In Defense of Point-to-Point ICP – Simple, Accurate, and Robust Registration If Done the Right Way. *IEEE Robotics and Automation Letters (RA-L)*, 8(2):1029–1036, 2023.

[30] J. Yang, Q. Zhang, Y. Xiao, and Z. Cao. TOLDI: An effective and robust approach for 3D local shape description. *Pattern Recognition*, 65:175–187, 2017.

[31] C. Yuan, J. Lin, Z. Zou, X. Hong, and F. Zhang. STD: Stable Triangle Descriptor for 3D place recognition. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2023.

[32] Y. Zhang, P. Shi, and J. Li. LiDAR-Based Place Recognition For Autonomous Driving: A Survey. *arXiv preprint*, arXiv:2306.10561, 2023.

Appendix B

Poster

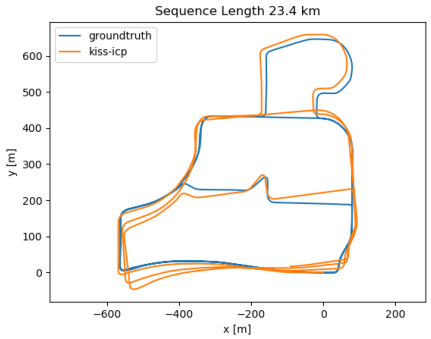
3D LIDAR Place Recognition for Autonomous Driving Exploiting Local Density Maps

Saurabh Gupta, Tiziano Guadagnino, Benedikt Mersch, Ignacio Vizzo, and Cyrill Stachniss

UNIVERSITÄT BONN

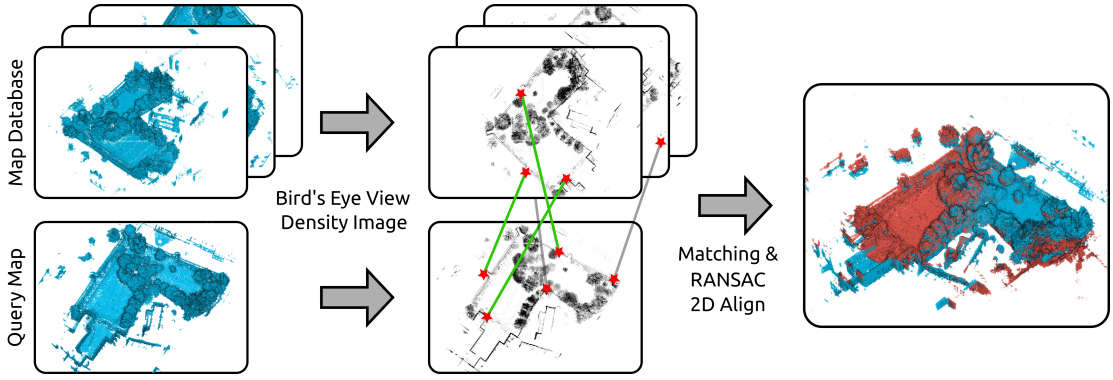
Abstract

- Robot pose estimation suffers from drift over longer traversals
- Detecting revisited places allows to recover from such a drift
- We suggest utilizing local maps and their bird-eye-view projections to detect revisited places
- Aggregation of data allows usage of simple features to describe local maps, agnostic of the sensor scan pattern



Method

- We aggregate consecutive LiDAR scans into a local map using the odometry estimates from a LiDAR odometry pipeline
- We project these local maps into 2D density image using bird-eye-view projection
- We compute binary ORB features from these density images and store them in a binary tree
- We match query features with the database and perform geometric verification to detect loops
- We provide a complete 2D alignment between matched local maps



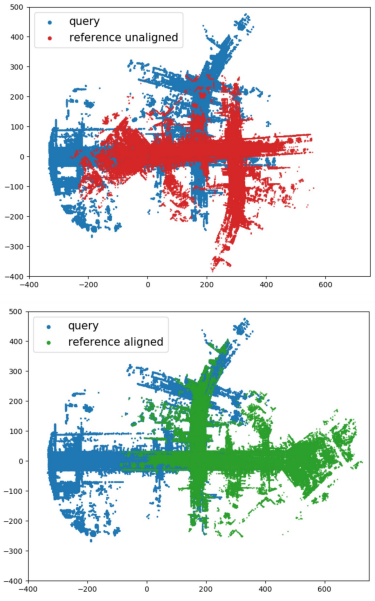
An overview of our approach

Experiments and Results

- Our proposed approach performs better or at least on-par with other state-of-the-art methods on various datasets
- It has the best F1 score among all baselines
- It provides a reliable solution for the 2D alignment between matched local maps
- This alignment can be used for a fine 3D registration for loop closing in SLAM

Datasets	MK03			MR02			MS01			NCD			HT01		
	P	R	F1	P	R	F1	P	R	F1	P	R	F1	P	R	F1
SC	**0.890**	0.450	0.598	0.259	0.071	0.111	0.027	0.126	0.045	0.415	0.007	0.013	N/A	N/A	N/A
CC	0.798	0.424	0.554	0.695	0.080	0.144	0.244	0.076	0.116	0.226	0.030	0.053	N/A	N/A	N/A
STD	0.679	0.435	0.530	0.491	0.042	0.078	-	-	-	-	-	-	-	-	-
Ours	0.730	**0.668**	**0.698**	**0.939**	**0.731**	**0.822**	**0.983**	**0.298**	**0.458**	0.927	**0.068**	**0.127**	**0.473**	**0.669**	**0.554**
Ours (E)	0.759	0.606	0.674	0.639	0.118	0.199	0.978	0.296	0.454	**0.933**	0.043	0.083	-	-	-

Quantitative results: Precision, Recall and F1 scores



Qualitative results