UNIVERSITÀ DI PARMA

Dottorato di Ricerca in Tecnologie dell'Informazione

XXXV Ciclo

Geometric Approach to Registration and Mapping with Multi-layer LIDARs

Coordinatore:

Chiar.mo Prof. Marco Locatelli

Tutor:

Chiar.mo Prof. Dario Lodi Rizzini

Dottorando: *Asad Ullah Khan*

December 2022

Contents

Abstract

The advancement of autonomous navigation technology has enabled a wide range of applications such as autonomous driving, industrial automation, and remote sensing. One of the most important components of autonomous navigation technology is LIDAR odometry and mapping. Odometry involves estimating the movement of a mobile platform, such as a self-driving car, in relation to its surroundings without using measurements from a fixed reference point. This is particularly important in situations where a fixed reference point is not available, or when physical landmarks may be obscured by other objects. Even when a fixed reference point is present, odometry can still be used to improve position estimates by limiting the changes in position over a short period of time. Odometry is essential for a wide range of mobile robotics applications, including self-driving cars operating in GPS-denied environments.

In mobile robotics, both cameras and LIDAR sensors are often used for odometry. LIDAR sensors measure the time it takes for laser pulses to travel to and reflect off objects in the environment, allowing them to collect range and bearing information about the scene. Despite their higher cost, LIDAR sensors are sometimes preferred over cameras for certain applications because they are not affected by ambient lighting conditions and do not suffer from glare or motion blur in low-light situations. Additionally, the direct range measurement capability of LIDAR sensors can simplify

the odometry algorithm by eliminating the need to estimate distances from multiple sensor measurements.

The main contribution in this thesis is features extraction for the LIDAR Odometry and Mapping (LOAM) algorithm, which often rely on features extracted from point clouds. This thesis proposes a novel detection algorithm SKIP-3D (SKeleton Interest Point) for extraction of features namely edges and planner patches from multi-layer LIDAR scan. SKIP-3D make use of the organization of LIDAR measurements to search for salient points in each layer through an iterative bottum-up procedure. In the process it removes the low curvature points to find edges and classifies the points from point clouds acquired from different view points are associated and used for their alignment. Evaluation is carried out using the KITTI odometry metric on the KITTI odometry dataset as well as a dataset collected at University of Parma and demonstrated our method on several robotic datasets in both structured and unstructured large, three dimensional environments. Experimental results showed that Fast LIDAR odometry and Mapping (F-LOAM) based on SKIP-3D feature extractor performs at least better than original F-LOAM feature extractor. Additionally, this thesis also addresses the issue of using heuristic procedures to exploit the organized point cloud structure of LIDARs, and presents alternative methods for improving feature point search. Presenting specific techniques for leveraging the structure of LIDAR point clouds, and introduces two key contributions in this area: efficient range search and nearest neighbor search. These methods allow for more effective exploitation of the organized structure of LIDAR point clouds.

To summarize, the results of the evaluation indicate that incorporating geometric information can improve the performance of the method. However, there is potential for further improvement through the development of a more sophisticated scene appearance model and a more effective application of this model.

Acknowledgments

Pursuing my doctoral studies was a wonderful and one of a kind experience in my life. I am deeply grateful to my supervisor and mentor, Dario Lodi Rizzini, for providing me with the opportunity to pursue my doctoral studies in the Robotics and Intelligent Machines Laboratory at University of Parma. His guidance and support have been invaluable in helping me carry out my research and achieve my goals. I have learned so much from him, including the importance of approaching challenges with a positive attitude. Thank you for your invaluable guidance, support, and encouragement throughout the entire process of writing this thesis.

I want to extend my heartfelt thanks to my wife and baby girl, who is now 7 months old. Their love and support have been vital to me and I am so grateful to have them in my life.

Finally, I would like to take a moment to remember and honor my mother, who passed away years ago but remains a constant source of inspiration in my life.

1 INTRODUCTION AND LITERATURE REVIEW

1.1 Motivation

Recently, in the area of mobile robotics three dimensional representations of our real world has assumed high importance. Particularly, effective 3D mapping can advance robotic systems in navigation and in a variety of related tasks like transportation, surveillance, modeling of buildings, and environmental issues. Our surrounding is characterized by natural and continuously changing processes. Developing fully autonomous mobile robots is a challenging process that involves a range of tasks, including navigation, motion planning, and control. Among these tasks, navigation is often considered a critical aspect because it enables the robot to accurately perceive and understand its surroundings and accurately determine its location. Simultaneous localization and mapping (SLAM) is a key technology that helps with these tasks by allowing the robot to create and maintain a map of its environment while simultaneously determining its location within that environment. SLAM involves creating a map of an unknown environment and continuously updating it as the mobile robot navigates through the environment. This computational problem involves two main components: localization, or determining the robot's location within the environment, and mapping, or creating the map itself. There are many different types of SLAM algorithms, and the most suitable one for a particular application depends on various

factors such as the nature of the features in the environment, the resolution and time constraints of the map, the sensors being used, and the available computation power.

Generally, to accomplish this objective, various set of sensors are deployed on robots. Some of these sensors acquire measurements about the environment in the form of points and ranges. This is the case of infrared or ultrasound range finders [1] or of other sensors reading scanning in a plane (e.g, LIDARs). These types of sensors data structure permits a safe movement of an autonomous robots in unknown environment. The affordability of both monocular cameras and IMUs makes visual-inertial algorithms quite popular in the literature [2]. Initially limited by computational power, online state estimation was mostly relying on filter-based methods such as the one presented in [3]. Later frameworks like [4] and [5] combined both local and global optimizations to estimate the system trajectory in real-time as well as creating consistent maps with loop-closure detections.

LIDAR (Light Detection and Ranging) is a remote sensing technology that uses lasers to measure the distance to an object or surface. LIDAR can be used for a variety of applications, including mapping, surveying, and object detection. There are several advantages to using LIDAR over other sensors for registration and mapping, which are discussed in detail in the following sections. Multi-layer LIDARs have emerged as an important sensing technology due to their capability to provide direct, dense, active, and accurate depth measurements of 3D shapes. In the last decade, their role has increased in many applications such as self-driving cars [11] and autonomous unmanned aerial vehicle (UAVs) [12], [13]. In the context of visual odometry (VO), it is common to simplify the brief exposure period of cameras as a single instant in time and use a discrete-time state formulation. On the other hand, rotating LIDARs like the Velodyne VLP-16 continuously sample the environment.

Despite advances in technology, it remains difficult to use LIDAR efficiently and accurately for odometry and mapping tasks. To use LIDAR-based localization and mapping in various applications, it is necessary to obtain precise and timely state estimation and a detailed 3D map while using limited computational resources on the device. It is necessary to overcome certain challenges in order to successfully use LIDAR for odometry and mapping.

- LIDAR sensors produce a significant amount of 3D points per second, with some generating upwards of millions. These points can be used to create a detailed 3D map of the environment, as well as to determine the position and orientation of the sensor in the environment. However, processing such a large volume of data in real-time can be challenging. To handle this large volume of data in real-time on limited onboard computational resources, it is necessary for the LIDAR odometry methods to be highly efficient in their calculations.

- There are several techniques that can be used to reduce the size of a LIDAR point cloud data set: Sampling (this involves selecting a subset of points from the original point cloud to represent the overall shape and structure), Filtering (this involves removing points from the point cloud that are not necessary for the task at hand). However, extracting feature points, such as edge points or plane points, is often used to reduce the amount of processing needed. Yet, the effectiveness of this feature extraction can be impacted by the environment. As a result, using a LIDAR odometry method may require significant manual adjustments to achieve optimal results.

- LIDAR point clouds can be distorted by motion for several reasons. One common cause is when the sensor is moving too quickly, causing points to be sampled at different locations at different times. This can result in points being displaced from their true positions, leading to inaccurate representations of the environment. Additionally, if the sensor is mounted on a moving platform that experiences significant acceleration or deceleration, the points may appear to be distorted due to the motion of the platform. Other factors that can contribute to motion distortion in LIDAR point clouds include vibrations, oscillations, and changes in orientation. To mitigate these issues, it is important to ensure that the LIDARs system is mounted securely and that the sensor is moving at a consistent and controlled speed. In some cases, it may also be helpful to use additional sensors, such as IMUs, to compensate for motion distortion.

- LIDAR sensors generate point cloud data that is often both dense and sparse at the same time. The density of the point cloud refers to the number of points

that are measured within a given area, while the sparsity refers to the distribution of those points. One reason why LIDAR point clouds are often dense is that the sensor is able to measure a large number of points in a short period of time. LIDAR sensors use lasers to measure distances, and the speed at which the laser beam is emitted and the speed at which the sensor rotates or moves can affect the density of the point cloud. Additionally, the density of the point cloud can be influenced by the configuration of the sensor, such as the number of lasers and the size of the beam. At the same time, LIDAR point clouds are often sparse because the resolution between scanning lines is relatively low compared to other types of sensors. This means that the points are not evenly distributed throughout the 3D space, and there may be large gaps between points. The sparsity of the point cloud can also be affected by the environment in which the sensor is operating. For example, a LIDAR sensor may be able to measure a dense point cloud in a flat, featureless environment, but a more complex or cluttered environment may result in a sparser point cloud.

One of the primary challenges of using LIDAR for estimating the motion and creating maps of a robot's environment is the need for accurate and customized algorithms. Without these algorithms, the measurements obtained from LIDAR may not be effectively utilized for these purposes. The aim of this thesis is to address this challenge by developing and implementing algorithms that can accurately and effectively utilize LIDAR measurements for robot motion estimation and mapping. Overall, the primary motivation for using LIDAR over other sensors for registration and mapping is the ability to produce high-resolution, accurate results over long ranges and in a variety of weather conditions.

1.2 Overview of Sensing Technology

Robotic platforms that interact with and move through their environment are made up of three main components: sensors for perceiving the external environment, hardware for processing this information in real-time, and actuators for performing control

actions. The optimal integration of these systems and components will determine the capabilities and level of autonomy of the vehicle.

To ensure the safety of all individuals involved for instance, in industrial environment, both inside and outside the vehicle, it is important to equip the robot with multiple sensors that provide diverse and redundant information about the environment. In this context, we will review the advantages and disadvantages of commonly used sensors on robotic platforms and autonomous vehicles, such as cameras, radars, lasers, and ultrasounds.

1.2.1 Camera

Optical RGB sensors, which use computer vision techniques to analyze color images for tasks such as object detection, segmentation, tracking, and optical flow, are a common source of information due to their versatility and low cost. However, these images can be compromised by external factors such as harsh weather conditions or the changing appearance of a scene at different times of day. Additionally, in the robotic context, the perception systems of a robot can be disrupted by other lights at night. The use of deep learning technologies has significantly improved the accuracy of camera-based perception systems in recent years.

1.2.2 Ultrasonic

Ultrasonic sensors measure distance based on the principle of reflected sound waves and are commonly used in driving assistance applications such as obstacle warning and parking distance measurement. However, their accuracy and robustness decline with increasing distance, and their measurements can be greatly affected by atmospheric conditions like pressure and humidity. This limits the use of ultrasonic sensors to close range applications, preventing them from providing more detailed and comprehensive information.

1.2.3 Ultrasonic

Radar sensors emit radio signals and capture their echoes to obtain sparse measurements of the dynamics of a scene using the Doppler effect. While these sensors are useful for robots because they can detect dynamic elements that may pose a danger to the vehicle, they have a short wavelength that limits their ability to detect small objects and introduces noise. Additionally, radar sensors do not provide robust information about the static structure of the nearby environment, which is also important for robotic applications.

1.2.4 LIDAR

Early forms of Light Detection and Ranging (LIDAR) sensors were introduced over 60 years ago, with early detectors being more akin to today's common household laser based measuring tool. A common Light Detection and Ranging (LIDAR) sensor emits light in the pattern of pulses into the surrounding environment and the reflected light pulse reaches back to the original source. The sensor operation uses Time of Flight (ToF) principles, where time is measured between the target object and the reflected pulse after hitting the surface arrives back to the center of the sensor. The distance between the object hitting and source can be measured using:

$$D = \frac{ct_f}{2} \tag{1.1}$$

In other words, assuming insignificant atmospheric attenuation distance D can be measured by multiplying t_f which is the time of flight and c which is the speed of light and dividing by half to retrieve one way distance.

There are three levels of LIDAR Technologies. 1D, 2D and 3D and all of them works with the same principle of ToF. 1D is the simplest form of LIDAR sensor, a single laser emitter used in a static position and directed towards a certain target. sensors that possess one-dimensional distances are categorized as 1D sensors. A 2D LIDAR sensor uses a single emitter but in rotation or motion to capture information in a planar environment. The source can be operated as a rotating or moving beam on one level, this yields distance and angle so, the resultant captured information is in

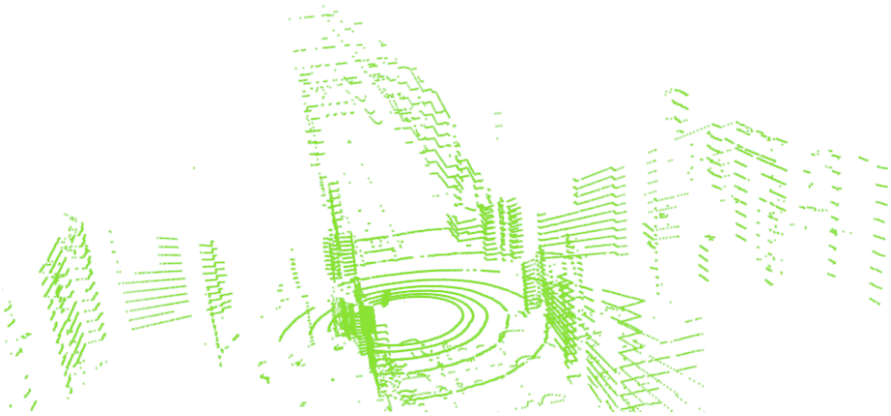


Figure 1.1: A scan from a Velodyne VLP-16 LIDAR mounted on pioneer 3DX robot

two dimension. Moreover, with the addition of third dimension to LIDAR technology when the sensor is pivoted. This way the LIDAR provides information about the distance and position along the x-axis, y-axis and z-axis. A 3D LIDAR operates the same functionality as that of 2D LIDAR. However, in the 3D case multiple emitters are positioned in the vertical axis of the sensor. This can be termed as multi-layer scanner. Lets take the example of a 3D Velodyne Puck (VLP-16), This LIDAR is composed of 16 vertical scan layers or rings. when the LIDAR rotates entire cycle and completes a full revolution captures 360° view of its surrounding. A complete revolution of the LIDAR scan is referred to as a sweep.

The sampling pattern of LIDAR sensors used in automotive applications can be anisotropic, making it difficult to apply general scan matching algorithms.

Figure 1.1 is an example of a scan collected from a LIDAR. This type of LIDAR sensor generates distinct rings using laser-detector pairs. The azimuthal resolution, determined by the sensor's rotation rate, is finer than the elevation resolution, which is based on the number and arrangement of the laser-detector pairs. In addition to displaying the distribution of points collected by LIDAR, the figure also illustrates the quality of the data. The geometric features of the scene are clearly visible, especially in areas with high elevation resolution. The figure also demonstrates the usefulness of appearance information, as walls and sharp segments are easily distinguishable in

the scan.

In terms of localization, LIDAR can be used to accurately determine the position of an autonomous vehicle within a map of its environment. This is achieved by continuously measuring the distance to nearby objects and landmarks using the laser beams, and comparing these measurements to a pre-existing map of the environment. By continuously updating its position based on these measurements, the autonomous vehicle can maintain a high level of accuracy in its localization.

In terms of mapping, LIDAR can be used to create detailed and accurate 3D maps of an environment. These maps can include information about the location and shape of objects and features in the environment, as well as their reflectivity and other physical properties. These maps can be used by autonomous vehicles to navigate and make decisions about their surroundings, as well as by other applications such as robotic inspection and maintenance of infrastructure.

Overall, LIDAR is an important technology for autonomous vehicle localization and mapping due to its ability to provide accurate and detailed information about the surrounding environment in real-time.

1.3 State of the Art

1.3.1 Robot Localization and Mapping

Localization is the process of determining the position of a robot or device within an environment. Mapping is the process of creating a representation of the environment, often in the form of a map. These two processes are often used together in robotics and autonomous systems to enable the system to navigate and understand its surroundings.

SLAM algorithms can be classified into two main categories: online and offline. Online algorithms operate in real time, continually updating the map and robot's pose as new sensor data becomes available. Offline algorithms, on the other hand, process all the sensor data at once after the robot has completed its exploration of the environment. The problem is hard when estimation is achieved using only on-board sensors. SLAM algorithms can use a variety of sensors, including laser rangefinders, vision

systems, and inertial measurement units (IMUs). The choice of sensor depends on the specific requirements of the application and the characteristics of the environment in which the robot is operating. SLAM algorithms typically rely on probabilistic techniques, such as Kalman filters and particle filters, to estimate the robot's pose and map the environment. These techniques allow the algorithm to account for uncertainties and errors in the sensor data and make probabilistic predictions about the robot's position and the structure of the environment.

There are several formulations of localization and mapping, and the choice of which approach to use depends on the specific requirements and constraints of the system. Some common formulations include:

- Simultaneous localization and Mapping (SLAM): This approach involves using sensors and algorithms to simultaneously build a map of the environment and determine the position of the robot within that environment. SLAM algorithms can be used with a variety of sensor types, including laser rangefinders, cameras, and inertial measurement units (IMUs).

- Marker-based Localization: This approach involves using distinct visual features or markers within the environment to determine the position of the robot. The robot can use these markers to determine its position by comparing its observations to a pre-defined map of the markers.

- Dead Reckoning: This approach involves using sensors such as IMUs and encoders to track the movement and orientation of the robot as it moves through the environment. This information can be used to determine the robot's position relative to its starting point.

- GPS: Global Positioning System (GPS) can be used to determine the position of a robot or device in outdoor environments where satellite signals are available. GPS is not typically suitable for use in indoor environments.

There are many other approaches to localization and mapping, and the choice of which approach to use depends on the specific requirements and constraints of the

system. Some other approaches include visual odometry, map-based localization, and beacon-based localization.

In the early days of SLAM, researchers used sensor data to identify features and combined this information with motion controls to estimate the position and orientation of a robot within an unknown environment using an Extended Kalman Filter (EKF) [6], [7]. These methods were implemented and tested by Moutarlier and Chatila [8], who used laser scans to identify line segments as features, and Leonard and Durrant-Whyte, [9] who used sonar to find unique geometric features. The EKF linearizes the motion and sensor models, which can be problematic in highly nonlinear situations, such as when using range-only beacons to form a ring of probability [10]. One advantage of the EKF is that it maintains a complete covariance matrix and mean vector for all features, which is important for data association (determining that two observations are of the same feature). However, updating the full covariance matrix is computationally intensive, requiring $O(n^2)$ time, where n is the number of features, limiting the use of EKF SLAM to environments with a few hundred features. Researchers have devoted significant effort to developing efficient methods for large-scale SLAM. These approaches can be classified according to their map representation or estimation algorithm, but they all share the fundamental property of handling spatial sparsity or independence [11].

To avoid the challenge of identifying features, Lu and Milios [12] developed a featureless (non-Bayesian) approach for SLAM that utilizes odometry measurements and raw sensor data to build a sparse network of constraints between robot poses. This network can be solved using batch-iterative nonlinear optimization techniques. Gutmann and Konolige [13] later extended this method to allow for incremental updates to the constraint network, including the incorporation of loop closures. Duckett et al [14] employed relaxation methods to create an iterative algorithm with $O(n^2)$ updates, except for loop closures which required $O(n^3)$. Frese et al [15] further improved this approach by using multilevel relaxation methods to perform updates to the network in $O(n^2)$ time, even for loops. Hahnel et al [16] also contributed to the efficiency of SLAM through the use of lazy data association, a technique in which the maximum likelihood data association is determined while also tracking other associ-

ation hypotheses in a tree that can be expanded to consider alternate data associations if the map becomes inconsistent.

Features extraction is an essential function for UGVs to realize surrounding environment and automate driving decision making in urban areas. This section surveys several widely used feature extraction methods based on 3D point clouds to realize fast and accurate object recognition. Depending on the environment, indoor, outdoor or hybrid and performance requirement SLAM suffers in challenging environments. The addition of cameras in odometry is rigorously studied as cameras being low priced. System based on vision odometry [17] [18] [19] is precise in localization, but on the other hand deteriorate features map. Cameras do not measure the depth information while capturing the scene, as a result the obtained map is not observable. The monocular visual odometry has the property of scale ambiguity, as the camera capture features in multiple images chasing features over various frames, the position of feature as well the camera motion both can not be recovered accurately. This problem of recovery can be addressed by introducing inertial measurement unit (IMU) with visual odometry (VO) which makes the system visual inertial odometry (VIO). IMU can measure transient changes in the pose, which permits the recovery of scale ambiguity. However, introducing more sensors to the system means more precise coordination and accurate calibration among the sensors which leads to more complexity.

One of the major limitation of cameras is that they are dependent of ambient light. Each pixel in a photo depends on the exposure of impinging light to produce an intensity measurement. Hence, it is not practical to adjust the intensity in a dynamic environment, long exposure may produce blur in the image and VO can lose the track of features.

1.3.2 Registration with Point Cloud

3D laser scanners are widely used for collecting data under static conditions and can quickly generate large amounts of accurate 3D points. However, the post-processing stage of point cloud registration, which involves aligning multiple point clouds, can be laborious and time-consuming. This process typically involves manually gath-

ering scans from different positions and using either common features or iterative methods to register the data. Despite the advancements in 3D scanning technology, point cloud registration remains a challenging and resource-intensive task. The integration of multiple point clouds is a critical aspect of 3D scanning and modeling, as it allows for a comprehensive representation of an object or scene. Point cloud registration, also known as surface registration, is a key technology that enables the fusion of multiple point clouds into a single, unified point cloud with a common coordinate system. This process involves analyzing point clouds to accurately represent the real-world surface of an object and is essential for various applications, including autonomous driving [20], [21], [22], [23]. 3D reconstruction [24], [25], [26], [27], simultaneous localization and mapping (SLAM) [28], [29], [30], [31], and virtual and augmented reality (VR/AR) [32]. In order to perform effective point cloud registration, it is important to collect point clouds in an appropriate manner and to accurately determine the orientation of the scanned object in its local coordinate system. This typically involves matching a standard point cloud in the dataset with the scanned point cloud.

The goal is to determine the transformation, including translation and rotation, that is needed to bring these point clouds into a common coordinate system. There are two main types of point cloud registration: pairwise registration, which involves two point clouds and estimates the transformation between them, and multiview or groupwise registration, which involves multiple point clouds and calculates the transformation of each scan to a reference data set. Point cloud registration can also be divided into two stages based on accuracy: coarse registration and fine registration. Coarse registration is used to roughly align the point clouds, while fine registration is used to achieve higher accuracy in the alignment.

The literature on registration is extensive and includes different formulations for several (sometimes loosely) related problems. The application contexts (computer vision, navigation, etc.) and the formulation of registration (objective function, feature-based or correlation, etc.) result into a variety of works. While there are several classification criteria, the categorization into local and global methods is suitable for thoroughly discussing our contribution w.r.t. the literature.

Iterative Closest Point (ICP) [33] is likely the most popular registration algorithm. The estimation is achieved by iteratively alternating the point association and the estimation of the rigid transform that better aligns the associated points. ICP is sensitive to initial estimation and is prone to local minima. Over the years, several variants, like ICP with soft assignment [34], ICP with surface matching [35], affine ICP [36], KISS-ICP [37] have been proposed. Generalized Procrustes Analysis (GPA) has also been proposed to simultaneously register multiple point sets in a single optimization step [38, 39]. In some cases, the association among multiple point sets increases the robustness of estimation.

Alternative representations to point sets have been proposed to avoid explicit assessment of correspondences. Biber and Strasser [40] propose the Normal Distributions Transform (NDT) to model the probability of measuring a point as a mixture of normal distributions. Instead of establishing hard association, NDT estimates the transformation by maximizing the probability density function of the point set matched with the mixture distribution. The approach has been extended to 3D point clouds [41] and, with ICP, is part of standard registration techniques [42]. Other registration techniques are based on GMMs computed on point sets [43–45].

Several registration algorithms exploit rigidity of isometric transformation for selecting robust and consistent associations. The general procedure operates in two steps. The first step establishes an initial set of putative associations based on geometric criteria (e.g., correspondence to closest neighbor) or similarity of descriptors. The second step filters the outlier associations based on rigidity constraints. RANSAC [46] and its many variants like MLESAC [47] implements this principle through a heuristic random consensus criterion. Coherence point drift (CPD) algorithm [48] represents point sets with a GMM and discriminates outliers by forcing GMM centroids to move coherently as a group. Ma et al. [49] proposed more formally consistent assessment of associations using Vector Field Consensus (VFC). This method solves correspondences by interpolating a vector field between the two point sets. Consensus approach has also been adopted to the non-rigid registration of shapes represented as GMMs [50, 51]. The hypothesize-and-verify approach is often successful in the estimation of associations, but it depends on the initial evaluation of

putative correspondences. Moreover, it does not provide any guarantee of optimality of the solution.

Global registration methods search the rigid transformation between two point sets on the complete domain of solutions. Heuristic global registration algorithms are based on particle swarm optimization [52], particle filtering [53], simulated annealing [54].

Another category includes the global registration methods that compare features, descriptors and orientation histograms extracted from the original point clouds. Spin Images [55], FPFH [42, 56], SHOT [57] and shape context [58] are descriptors that could be matched according to similarity measure and used for coarse alignment of point cloud. Similar histogram-based methods are applied to rotation estimation of 3D polygon meshes through spherical harmonics representation [59–63]. All these techniques extend the searching domain and attenuate the problem of local minima, but their computation is prone to failure or achieves extremely coarse estimation. Moreover, the global optimality of the assessed solution is not guaranteed.

In planar registration problem, some effective global methods exploiting specific descriptors of orientation have been proposed. Hough spectrum registration [64] assesses orientation through correlation of histogram measuring point collinearity. The extension of this method to 3D space [65] suffers from observability issues due to symmetry in rotation group. Reyes-Lozano et al. [66] propose to estimate rigid motion using geometric algebra representation of poses and tensor voting. Angular Radon Spectrum [67, 68] estimates the optimal rotation angle that maximizes correlation of collinearity descriptors by performing one-dimensional BnB optimization.

BnB paradigm is the basis for most of global registration methods. Breuel [69] proposes a BnB registration algorithm for several planar shapes in image domain. The shapes are handled by a *matchlist* containing the shapes to be matched. The bounds are computed using generic geometric properties, which partially exploit pixel discretization. No accurate management of lower bounds is presented. The BnB algorithm in [70, 71] computes the rigid transformation that matches two point sets, under the hypothesis that the set of correspondences is given, although with outliers. To our knowledge, Go-ICP [72] is the most general algorithm for the estimation of the glob-

ally optimal registration based on BnB. Recently, other formally guaranteed methods like TEASER [73] have been proposed to effectively estimate the relative pose with high number of outlier associations, but they may still fail with highly inaccurate initial correspondences.

Although effective and guaranteed in motion estimation, globally optimal registration methods cannot be applied to the odometry context due to their runtime limitations. They are far from real-time when there is much overlap between the point clouds, as in the odometry context.

1.3.3 LIDAR Odometry and Mapping

Multi-layer LIDARs acquire a large amount of range measurements in a single shot. Such sensing technology is naturally suitable for robot localization and mapping. In particular, registration of LIDAR scans allows estimation of robot motion, also called *sensor odometry*, and map construction by merging multiple aligned scans. Standard registration algorithms and specifically those relying on point correspondences, such as those derived from ICP, poorly performs when applied to the sparse point clouds of LIDARs. An effective approach based on customized geometric features for LIDAR has been developed in the last years.

The original method was called *LIDAR odometry and mapping* (LOAM) [74, 75] and such denomination has been inherited by the derived methods. LOAM belongs to the feature extraction and matching category. LOAM matches scan to scan points based on their geometry. The IMU measurements are used to undistort the distortion occured during a scan from non-zero acceleration. Lego-LOAM [76] is another variant of LOAM which, introduces loop closure to reduce the drift accumulated over time. F-LOAM [77] is an extension to original LOAM, it brings improvement to the system by performing double stage distortion compensation. Other variants like LIO (Lidar Inertial Odometry) [78, 79] and IN2LAMA (INertial Lidar Localisation And MApping) [80] combine registration and inertial sensors. An augmented version of LOAM combines the geometric features with vision keypoint features [77].

While inertial measurements or vision features can effectively improve registration through an initial guess to the estimation, they also increase the constraints on

the input data. The contribution of this thesis is to built on the simple and unconstrained approach represented by LOAM and F-LOAM. Thus, LOAM and F-LOAM are accurately discussed in this section.

Since LIDAR odometry is always in motion, the system accumulate errors during the process of integration. This accumulated error leads to a drift in odometry. The LIDAR Odometry and Mapping in Real-time (LOAM), proposed an algorithm to solve the problem of real-time odometry by feature extraction from a 3D point clouds using 3D LIDAR and ultimately matching the scans. This is not a simple task as the measurements received from multi-layer LIDAR are not at the same time. This can give rise to miss-association while matching two consecutive point clouds hence, results in a less accurate final map.

Previously this problem was only solved efficiently with off-line batch methods, often using loop closure to correct for drift over time. The LOAM algorithm is able to achieve both low-drift and low-computational complexity, without the requirement of high accuracy ranging or inertial measurements. This level of performance comes from the division of the complex problem of simultaneous localization and mapping in two algorithms. Instead of optimizing a large number of variables simultaneously, one algorithm performs odometry at a high frequency, with less accuracy, having the objective of estimating the velocity of the LIDAR. The other algorithm runs at a lower rate performing fine matching and registration of the point cloud. The combination of both algorithms allows the mapping in real-time. The problem addressed by LOAM is commonly known as ego-motion estimation that is the motion estimation using visual data. In this case, this data is a point cloud perceived by a 3D LIDAR, where simultaneously a map for the traversed environment is built. It starts by assuming that the LIDAR is pre-calibrated, the angular and linear velocities of the LIDAR are smooth, continuous over time and without abrupt changes. The authors defined as convention, a sweep as being one complete LIDAR scan, Where j indicates the individual sweeps, and $\mathbf{P}_j$ is the point cloud perceived during a sweep j. Two coordinate systems are defined as follows;

- LIDAR coordinate system L : This is a 3D coordinate system with it's origin at the geometric center of the LIDAR. The x-axis is pointing to the left, the

y-axis is pointing upward, and the z-axis is pointing forward. The coordinates of a point i, i belongs to $\mathbf{P}_j$, in L_j are denoted as $\mathbf{X}^L_{(j,i)}$.

- World coordinate system W : It is also a 3D coordinate system coinciding with R at the initial position. The coordinates of a point i, i belongs to $\mathbf{P}_j$, in $\mathbf{W}_j$ are $\mathbf{X}^w_{(j,i)}$.

After this, the LIDAR odometry and mapping problem can be better defined as: given a sequence of LIDAR clouds $\mathbf{P}_j$, compute the ego-motion of the LIDAR during each sweep j, and build a map with $\mathbf{P}_j$ for the traversed environment.

1.3.4 LIDAR Odometry

LIDAR odometry is a technique for estimating the pose (position and orientation) of a vehicle or robot using data from LIDAR sensor. LIDAR sensor measures the distance to objects in the environment, and generates a 3D point cloud of the environment. To perform LIDAR odometry using the 3D point cloud data to estimate the pose of the sensor over time, the system must first align the current LIDAR scan with a previous scan to determine the relative motion between them. Once the relative motion between the scans has been estimated, the system updates the pose estimate of the robot based on its motion. This process of estimation is repeated until the last point cloud in the data. LIDAR odometry has several advantages over other methods of pose estimation, for example relying on the wheel odometry or visual odometry. it is less sensitive to errors caused by wheel slippage or changes in surface conditions.

LIDAR odometry can be used in a variety of applications, including autonomous vehicles, robotics, and mapping. Building a high precision map for navigation requires accurate odometry estimation. Moreover, odometry is fundamental in localization and environment perception, which guarantees safety. considering dynamic environment, odometry method should be vigorous to different environments including indoor and outdoor [2]. In variation to methods of using video cameras [81], [82], the odomtery of LIDAR is invariant to large changing in lightning conditions as the sensor actively emits the laser beams.

The odometry algorithm can be divided in three distinct steps; feature point extraction, followed by feature Point Correspondence and motion estimation step.

3D mapping is a widely used technology, and one challenge that arises in laser ranging is accurately aligning the resulting point cloud when the sensor is moving. This is because the registration of the point cloud must take into account both the internal kinematics of the LIDAR and the external motion of the sensor. A common solution to this problem is to use an external method of pose estimation, such as a GPS/INS system, to align the range data with a fixed coordinate frame. Alternatively, when such measurements are not available, odometry techniques can be used to estimate the sensor's position and orientation over time based on factors such as wheel motion, gyroscopes, or tracking features in the range or visual images. High-rate pose estimation is particularly important for lasers, which can take thousands of measurements per second. The odometry block is responsible for the estimation of sensor motion between two consecutive scans. Performing point-to-feature scan matching yields the transformation between two scans.

After the segmentation each feature is labeled and the correspondence between two scans is achieved by matching the same labeled features from two consecutive scans. Edge labeled feature in current scan is compared with Edge labeled feature in a previous scan. The same way planar features from current and previous scans are compared. To find the minimum distance transformation, a two step optimization method Levenberg-Marquardt (L-M) is applied to two consecutive scans. Finally, the $6D$ transformation is applied between the two neighboring scans.

Local odometry in LEGO-LOAM is achieved using a sliding window optimization approach, in which the current LIDAR scan is aligned with a window of previous scans to estimate the local motion of the sensor. The window size is adaptively adjusted based on the motion of the sensor, allowing the algorithm to handle changes in the environment and maintain high accuracy over time.

To reduce the complexity of the problem, the motion during a sweep is modeled with constant angular and linear velocities. This allows the use of linear interpolation to calculate the pose transform, for points that are being received at different times. The LIDAR 6-DOF pose transform is stored in a state vector, that contains

the translations and rotations. This vector also encodes the motion of the LIDAR and is represented by $\mathbf{T}_j^R(t) = [T_j^R(t), \theta_j^R(t)]$, where t is the current time stamp, and as stated before, t_j represents the starting time of the current sweep j. The translations are represented by $T_j^R(t) = [t_x, t_y, t_z]^T$ and the rotation by $\theta_j^R(t) = [\theta_x, \theta_y, \theta_z]^T$ in R_j. The rotations $\theta_j^R(t)$ can be enconded in a rotation matrix, defined with the help of the Rodrigues formula [41].

1.3.5 Key points Extraction

Scan matching and feature extraction are the two main approaches to build a complete map of the surrounding with LIDARs data. Features or landmarks usually depend on the environment: indoor settings, sharp structures, planar surfaces. For instance extraction of lines, corners have already been used in [83], [84], [85]. The fact that 3D LIDARs provide multiple scan lines, for a SLAM system, many applications uses LIDAR sensor to detect the surrounding. Man, Ye [86] proposed corner feature extraction method from 2D LIDAR data. To process LIDAR data Harris corner detector is used by Li [87] and an algorithm to detect the roadway by detecting geometric features is presented in [88]. Due to the high cost of 3D LIDARs a spinning or rotating laser scan has been used to measure the environment [89], [90].

This step consists in the extraction of feature points present on the LIDAR cloud, $\mathbf{P}_j$. The feature points are extracted from individual scans $\mathbf{P}_j$ and selected depending on their co-planar geometric relationships. The desired points to be selected, are the ones present on sharp edges and planar surface patches. This process starts by analyzing the smoothness of the local surface, this is accomplished by sorting the points present in a scan, using a defined threshold called C value. The feature points inside the defined maximum threshold are labeled as edge points, in opposition the points within the minimum threshold correspond to the planar points.

In LOAM the individual scans are divided into four identical sub-regions. Where each one only can provide in maximum two edge points and four planar points. The selection of these points needs to obey the following restrictions to ensure an even distribution of the feature points in the environment;

- The selected edge or planar points cannot exceed the maximum quantity defined for the equivalent sub-region.
- The surrounding points were not selected yet.
- Cannot make part of a surface patch perpendicular within 10 degree to the laser beam, or on the boundary of an obstructed region.

The selected points are used for feature extraction and the unwanted points are discarded. Taking a set S of points p_i from each row of range image and calculate the roughness of each point p_i in a continuous set S.

$$c = \frac{1}{|\mathbf{S}| \cdot \|r_i\|} \sum_{j \in S, j \neq i} \| (r_j - r_i) \| \tag{1.2}$$

The range images are equally divided into multiple sub-images and sorted based on the roughness values c. r_i and r_j are the range values associated with every point, represents the Euclidean distance from the corresponding point to the sensor. The same criterion like LOAM is applied here to differentiate between the types of features, which is basically to fix a threshold. Points with roughness c value greater then the threshold are recognized as Edge features and points with roughness c value less then the threshold are classifies as plane features.

The LEGO-LOAM algorithm consists of two main components: global registration and local odometry. Global registration estimates the global pose (position and orientation) of the sensor based on the alignment of the current LIDAR scan with a map of the environment. Local odometry estimates the local motion of the sensor based on the changes in the LIDAR scan data over time. Global registration in LEGO-LOAM is achieved using an efficient scan registration method that aligns the current LIDAR scan with the map using a least-squares optimization approach. The method is based on the Iterative Closest Point (ICP) algorithm, which iteratively minimizes the distance between the points in the current scan and the points in the map.

LEGO-LOAM is an algorithm for real-time 3D LIDAR-based localization and mapping. It was developed by researchers at the Robotics Institute of the Chinese Academy of Sciences and the University of Oxford. LEGO-LOAM system is composed of five different modules. A single point cloud scan obtained with 3D LIDAR

is feed into the segmentation module and the segmented point cloud is forwarded to the feature extraction stage. Point cloud acquired at time t is projected onto a range image in the segmentation block, where each point is represented by a unique pixel. Points are grouped into clusters and assigned with unique label. Clusters are removed on a set threshold, based on minimum number of points. This way the noisy points are filtered out and only the reliable points can be used for feature extraction. To further refine the pose transformation and obtain a map the mapping block matches the extracted features at a low frequency. To obtain the final transformation L-M method is used here again. A single surrounding map is achieved by saving all the previous features from every scan and focusing just on the features sets that are in the pre-defined range of the current sensor position.

LEGO-LOAM and LOAM are both algorithms for real-time 3D LIDAR-based localization and mapping. Both algorithms are widely used in robotics and autonomous vehicle applications and have been demonstrated to achieve high accuracy and robustness in a variety of environments.

One key difference between LEGO-LOAM and LOAM is the approach they use for estimating the movement of the sensor. LEGO-LOAM uses a combination of global registration and local odometry, while LOAM uses a single step scan matching approach.

In the global registration of LEGO-LOAM , the current LIDAR scan is aligned with the map using an optimization algorithm, such as the Iterative Closest Point (ICP) algorithm, to estimate the global pose (position and orientation) of the sensor. Local odometry estimates the local motion of the sensor based on the changes in the LIDAR data over time.

In contrast, LOAM uses a single step scan matching approach, in which the current LIDAR scan is directly aligned with the previous scan to estimate the movement of the sensor. This approach is faster than global registration, but may not be as accurate in environments with large changes or dynamic objects.

Another difference between LEGO-LOAM and LOAM is the way they handle changes in the environment over time. LEGO-LOAM uses a sliding window optimization approach, in which the current LIDAR scan is aligned with a window of

previous scans to estimate the local motion of the sensor. The window size is adaptively adjusted based on the motion of the sensor, allowing the algorithm to handle changes in the environment and maintain high accuracy over time. In contrast, LOAM uses a combination of scan matching and map optimization to handle changes in the environment. There are a few potential advantages of LOAM over LEGO-LOAM:

- Computational efficiency: LOAM uses a single step scan matching approach, which may be faster than the global registration and local odometry approach used by LEGO-LOAM. This can be particularly important in applications where computational resources are limited.
- Robustness to large changes in the environment: LOAM uses a combination of scan matching and map optimization to handle changes in the environment, which may make it more robust to large changes than LEGO-LOAM, which uses a sliding window optimization approach.
- Ease of implementation: LOAM has been widely used and studied, and there are many existing implementations and resources available. This may make it easier to implement and use than LEGO-LOAM, which may be less well-known.

It's worth noting that these potential advantages of LOAM over LEGO-LOAM may not always hold true, and the choice of which algorithm to use will depend on the specific requirements of the application and the trade-off between accuracy and computational complexity.

Other techniques may involve the use descriptors for feature extraction from LIDAR point clouds [91], [56], which are mathematical representations of local features in the point cloud. These descriptors can capture various properties of the point cloud, such as shape, texture, and orientation. By computing descriptors at each point in the point cloud, a set of high-dimensional feature vectors can be generated, which can then be used for further analysis or classification.

Several types of descriptors have been proposed for LIDAR point clouds, including spin images, 3D shape contexts, and spherical harmonics. These descriptors have

been shown to be effective for various applications, such as object recognition, segmentation, and registration.

1.4 Contributions

In this thesis, we introduce a novel detection algorithm SKIP-3D (SKeleton Interest Point) for extraction of features from multi-layer LIDAR scans for large scale scene understanding. The foundation of our approach is the LOAM pipeline proposed by [77]. We extend the pipeline to new features, especially autonomous driving scenarios. Combined with the FLOAM algorithm we add our novel features detector. Our approach acquires point level organization of LIDAR measurements to search salient points in each layer. In order to assess the effectiveness of our approach, we gathered a dataset of consecutive LIDAR point clouds using the Pioneer 3DX teleoperated robot fitted with the Velodyne VLP-16 sensor. To supplement this quantitative analysis, we conducted a qualitative assessment on a real-world dataset to confirm the validity of our methodology.

Additionally, Multi-layer 3D LIDAR allow acquiring 3D data in the shape of dense point clouds composed of several thousand 3D points. Due to the irregularity of collected points, this dense and complex point cloud put forward main challenges on different types of structures in the observed scene. Nevertheless, the peculiar arrangement of collected 3D data points for a particular structure it is enough to recognize geometric property of a surface. Consequently, we follow the exploration for feature extraction from point clouds and focus on the use of geometric features. The main reason behind the choice is that in a dense collection of points point-to-point correspondences between structures relies locally on geometric descriptor. In this regard, searching neighborhood for each 3D point and reconstruct structures based on the arrangement of locally positioned points is more appealing to us humans.

Moreover, I presents two methods for exploiting the specific organization of LIDAR point clouds in order to improve the efficiency and accuracy of various tasks, including efficient range search and nearest neighbor search. I also introduced Scan-Context, a method for place recognition within 3D point cloud maps that leverages

the spatial and temporal context of LIDAR scans to provide valuable information about the location and orientation of the scanner within the environment, improving the accuracy of place recognition and the robustness of feature extraction. These contributions represent important advances in the field of 3D point cloud analysis.

The main contributions of our proposed method are:

- Full exploitation of LIDAR point cloud structure and index for efficient neighborhood search and for place recognition based on existing signatures.
- The novel geometric features SKIP-3D for better detect salient parts in LIDAR point clouds based on bottom-up extraction procedure that preserves shapes.
- Extensive experiments using indoor and outdoor datasets of a F-LOAM using the proposed SKIP-3D features.

This thesis is organized as follows: Chapter 2 discusses how LIDAR point clouds can be exploited for efficient neighborhood search. It will explore techniques for organizing and processing the point cloud data, with the aim of reducing computational resources required for searching and querying the data. Chapter 3 proposes a novel feature detection method for LIDAR called SKIP-3D. This will describe the principles behind SKIP-3D, and demonstrate its performance through a series of experiments and comparisons with F-LOAM on several datasets. Finally, Chapter 4 summarizes the contributions of this thesis and presents thoughts for future research in this area.

2

Organized Point Clouds from Multi-layer LIDAR

In chapter 1 the advantages and the potential applications of multi-layer LIDARs have been extensively discussed. This sensing technology can guarantee robustness and accuracy, occupancy information in 3D space, large range and scale measurements along several directions. However, the acquired ranges are non-uniformly sampled on all directions, since horizontal angular resolution is higher than the vertical one. Hence, the resulting point clouds are generally sparse. The point clouds acquired by multi-layer LIDARs are naturally organized in the form of polar depth map. Each point is associated to a pair of indices, one corresponding to the elevation angle and the other to azimuth angle. The organization of range measurements is related to the physical structure of LIDARs that almost simultaneously fire vertical samples from the rotating head. The organization of scan point clouds has been implicitly exploited to efficiently perform operations involving the neighborhood of points. In [92], a method for extracting features from 3D point clouds that are sparse, noisy, and unorganized. The method is based on a two-stage process: first, a set of robust and stable keypoints is identified in the point cloud; second, local features are extracted from the keypoints. However, the exploitation of this polar structure is mostly limited to heuristic procedures and it has not been formally treated. The main focus of this chapter is on discussing the specific ways in which the organized point cloud structure of LIDARs can be exploited to improve the search for feature points. The key contribu-

tions of this chapter includes two ways to exploit the specific organized point cloud of LIDARs i.e, efficient range and nearest neighbor search. In addition this chapter also provides place recognition within 3D point cloud maps i.e. ScanContext. The position and orientation of a LIDAR scan can provide valuable information about the location and orientation of the scanner within the environment. This information can be used to improve the accuracy of place recognition by providing additional context about the spatial layout of the environment. Exploiting the scan context can also help to improve the robustness of feature extraction, which is an important component of place recognition. By using the spatial and temporal context of the scans, it may be possible to extract more stable and reliable features, which can improve the accuracy of place recognition.

2.1 Indexing Point Cloud Scan

Multi-layer 3D LIDAR caries multiple set of emitters and receivers, the data points are the measurements of light energy in the shape of pulses fired from the laser source and a reflected light energy returns to the LIDAR where the measurement is recorded.The rotational angle of the sensor along its axis is the *azimuth* φ. The sensor simultaneously register N number of rays with different *elevation* angles θ w.r.t the rotation plane. The measurements are recorded in such a way that from each emitter-receiver pair it belongs to the same ring or layer in a point cloud.

Let i be the ring index and $i \in \mathscr{R}_\theta$ where the set of indices $\mathscr{R}_\theta = \{0, 1, \ldots, N-1\}$, each ring is distinguished by the elevation angle θ_i. Furthermore, ring elevations are equally spaced as

$$\theta_i = \Delta\theta \; i + \theta_{min} \tag{2.1}$$

For instance, considering the example of Velodyne VLP-16, $\theta_{min} = -15\,deg$, $\Delta\theta = 2\,deg$ and $N = 16$. Let j is any azimuth index and $j \in \mathscr{R}_\varphi = \{0, \ldots, w-1\}$. Any point $\mathbf{p}_{ij}$ of the scan is identified by ring and azimuth indices. The azimuth φ_{ij} of each point is defined as

$$\varphi_{ij} = \varphi_{start} + \Delta\varphi \; j + \varphi_{ring,i} \tag{2.2}$$

Where φ_{start} is the azimuth of pivoted sensor when cloud acquisition begins, $\Delta\varphi$ is the regular resolution, and $\varphi_{ring,i}$ the angular offsets of each ring. Several Velodyne-like sensors provides the range measurements grouped in data packets, each including $p \times h$ measurements. A point cloud covering at least a complete azimuth turn is obtained by $w = p \lceil \frac{2\pi}{p\Delta\varphi} \rceil$ azimuth samples. Thus, due to the packing of measurements, a cloud usually covers more than an azimuth complete turn, i.e, w is s.t. $\Delta\varphi\ w > 2\pi$. The final part of a point cloud is overlapped (or "wrapped") with the initial measurements. Such wrapping requires specific treatment, since for some values of elevation θ and azimuth φ there are two values of the indices. Moreover, the initial offset φ_{start} usually changes at each new acquisition, since the first measurement of the new point cloud is not aligned with the azimuth of the previous cloud. The ring offsets $\varphi_{ring,i}$ are set to avoid interference among rays firing simultaneously. In our analysis their contribution is negligible $\varphi_{ring,i} \simeq 0$. Given the distances ρ_{ij} corresponding to each ray, the Cartesian coordinates of a point are equal to

$$\begin{aligned} p_{ij,x} &= \rho_{ij}\ \cos(\theta_i)\ \cos(\varphi_{ij}) \\ p_{ij,y} &= \rho_{ij}\ \cos(\theta_i)\ \sin(\varphi_{ij}) \\ p_{ij,z} &= \rho_{ij}\ \sin(\theta_i) \end{aligned} \tag{2.3}$$

This equation straightforwardly shows that the indices i and j are directly related to the point coordinates. Efficient neighbor search could be performed on reduced index intervals, if there are bounds corresponding to spherical neighborhoods.

2.1.1 Index Bounds

Indexing data structures like kd-trees are often used to efficiently search neighbors in a point cloud, since this primitive operation is required by several algorithms. However, the specific organization of *scan point clouds* described above can be exploited to avoid external data structures. In particular, our goal is to limit the number of points to be evaluated using proper bounds on indices i and j corresponding respectively to elevation and azimuth. Let $\mathscr{D}_{\mathbf{q},r} \subseteq \mathbb{R}^3$ be an open ball centered in query point $\mathbf{q} \in \mathbb{R}^3$ with radius $r > 0$, s.t. $\mathbf{p}_{ij} \in \mathscr{D}_{\mathbf{q},r}$ if $\|\mathbf{p}_{ij} - \mathbf{q}\| < r$. Let $\hat{\rho}$, $\hat{\theta}$ and $\hat{\varphi}$ be the polar coordinates of $\mathbf{q}$, and $(\hat{i}, \hat{j})$ the corresponding indices defined according to eq. (2.1)-(2.2).

A point $\mathbf{p}_{ij}$ of the point cloud $\mathscr{C}$ belongs to $\mathscr{D}_{\mathbf{q},r}$ if

$$r^2 > \|\mathbf{p}_{ij} - \mathbf{q}\|^2 = \rho_{ij}^2 + \hat{\rho}^2 - 2\rho_{ij}\hat{\rho}\ \cos\alpha_{ij} \tag{2.4}$$

where α_{ij} is the angle between vectors $\mathbf{p}_{ij}$ and $\mathbf{q}$. The cosine of α_{ij} can be written w.r.t. azimuths and elevations using spherical cosine law

$$\cos\alpha_{ij} = \cos\theta_{ij}\cos\hat{\theta}\cos\left(\varphi_{ij} - \hat{\varphi}\right) + \sin\theta_{ij}\sin\hat{\theta} \tag{2.5}$$

Under the hypotheses $\rho_{ij}, \hat{\rho} > 0$, inequality (2.4) can be expressed w.r.t. $\cos\alpha_{ij}$. Moreover, $\cos\alpha_{ij}$ has maximum value 1 and, if also $\hat{\rho} > r$, another minimum bound can be found as

$$0 \leqslant \frac{\sqrt{\hat{\rho}^2 - r^2}}{\hat{\rho}} \leqslant \frac{\rho_{ij}^2 + \hat{\rho}^2 - r^2}{2\rho_{ij}\hat{\rho}} < \cos\alpha_{ij} \leqslant 1 \tag{2.6}$$

This condition enables to bound values of ρ_{ij} and α_{ij} for the points $\mathbf{p}_{ij} \in \mathscr{D}_{\mathbf{q},r}$ as

$$|\rho_{ij} - \hat{\rho}| < r, \qquad |\alpha_{ij}| < \bar{\alpha}_{ij} = \arccos\left(\frac{\sqrt{\hat{\rho}^2 - r^2}}{\hat{\rho}}\right) \tag{2.7}$$

Angle $\bar{\alpha}_{ij}$ can also be computed as $\arcsin(r/\hat{\rho})$, which is equivalent to the above expression. The condition on α_{ij} can be used to find bounds on both θ_{ij} and ϕ_{ij}. Since azimuth angles θ_{ij} and $\hat{\theta}$ belongs to interval $\left[-\frac{\pi}{2}, \frac{\pi}{2}\right]$ and the corresponding cosines are positive, the inequalities (2.7) and (2.5) imply that

$$\cos\bar{\alpha}_{ij} < \cos\left(\theta_{ij} - \hat{\theta}\right) \tag{2.8}$$

Thus, the elevation angles in the set are $|\theta_{ij} - \hat{\theta}| < \bar{\alpha}_{ij}$. Under the hypothesis that $\theta_{ij}, \hat{\theta} \neq \pm\pi/2$, φ_{ij} can be bound as

$$\cos\left(\varphi_{ij} - \hat{\varphi}\right) > \frac{\cos\bar{\alpha}_{ij} - \sin\theta_{ij}\sin\hat{\theta}}{\cos\theta_{ij}\cos\hat{\theta}} \geqslant \frac{\cos\bar{\alpha}_{ij}}{\cos\hat{\theta}} \tag{2.9}$$

The inequality (2.9) implies that

$$\left|\varphi_{ij} - \hat{\varphi}\right| < \beta_{ij} = \arccos\left(\frac{\sqrt{\hat{\rho}^2 - r^2}}{\hat{\rho}\cos\hat{\theta}}\right) \tag{2.10}$$

Thus, the corresponding bounds on the elevation and azimuth indices are given by

$$|i-\hat{i}| < \frac{\bar{\alpha}_{ij} - \theta_{min}}{\Delta\theta} \qquad |j-\hat{j}| < \frac{\bar{\beta}_{ij} - \varphi_{start}}{\Delta\varphi} \tag{2.11}$$

These bounds on indices can be used to reduce the candidate points in the neighborhood of query point as shown in the next section.

2.2 Radius and K-Nearest Neighbor Search

The index interval bounds derived before can be used to perform efficient neighbor search. There are two main kinds of search: radius range search and k-nearest neighbor search. The goal of radius range search is to find all the points distant from a query point $\mathbf{q}$ less than a given radius r. An efficient procedure is obtained from straightforward application of bounds on indices (2.11). The Algorithm 1 illustrates the algorithm `SearchWithinRadius`. At line 6 it checks whether the condition $r < \bar{\rho}$, necessary for the index bounds, holds. If this is the case, the search is limited to a rectangular interval. Otherwise, it is equivalent to a brute-force search.

The k-nearest search is more articulated as shown in Algorithm 2. The critical point is the choice of an anchor point $\mathbf{a}$ angularly close to the query point (lines 3-7). The anchor point is used to limit the search of nearest points to a range search with radius $\|\mathbf{a}-\mathbf{q}\|$ around query point $\mathbf{q}$. The idea is that, if there are no closer point than $\mathbf{a}$ in this range, it is guaranteed that no closer point to $\mathbf{q}$ exists. To extend the same reasoning to $k > 1$ points, the priority queue $\mathscr{Q}_{\mathbf{q}}$ is used. In particular, if the queue already contains k items, other candidate points are compared with $\text{top}(\mathscr{Q}_{\mathbf{q}})$, i.e, the farthest point from $\mathbf{q}$ in the queue. The search domain $\mathscr{D}^{curr}$ is expanded until there are no other candidates and avoiding previously visited (in $\mathscr{D}^{prev}$).

The Pioneer 3DX robot equipped with the Velodyne VLP-16 sensor has been teleoperated to collect a dataset of LIDAR point clouds. In particular, a point cloud have been collected from the LIDAR sensor every time the robot travels about $0.50\,m$. During the teleoperation the operator and other moving people have been captured in the measurements. The acquired dataset consists of 384 scan clouds, each containing 29184 points (including the invalid measurements) organized into 16×1824 matrix.

Algorithm 1 SearchWithinRadius ($\mathscr{C}$, $\mathbf{q}$, r)

Require: $\mathscr{C}$: the input point cloud organized in $w \times h$ items;
$\Delta\theta, \Delta\varphi$: elevation;
$\mathbf{q}$: the query point;
r: the radius of spherical search domain;
Ensure: $\mathscr{Q}$: the set of points in $\mathscr{D}_{\mathbf{q},r}$;
$\mathscr{Q} \leftarrow \emptyset$;
find indices of query point $\mathbf{q}$: $(\hat{i}, \hat{j})$;
if $\hat{\rho} > r$ **then**
    $\Delta i \leftarrow (\bar{\alpha}_{ij} - \theta_{min})/\Delta\theta$; (see ineq. (2.11))
    $\Delta j \leftarrow (\bar{\alpha}_{ij} - \varphi_{min})/\Delta\varphi$; (see ineq. (2.11))
    $i_{min} \leftarrow \max\{0, \hat{i} - \Delta i\}$, $i_{max} \leftarrow \min\{h-1, \hat{i} + \Delta i\}$;
    $j_{min} \leftarrow \max\{0, \hat{j} - \Delta j\}$, $j_{max} \leftarrow \min\{w-1, \hat{j} + \Delta j\}$;
else
    $i_{min} \leftarrow 0$, $i_{max} \leftarrow h-1$;
    $j_{min} \leftarrow 0$, $j_{max} \leftarrow w-1$;
end if
for $i \leftarrow i_{min}$ to i_{max} **do**
    for $j \leftarrow j_{min}$ to j_{max} **do**
        if $\|\mathbf{p}_{ij} - \mathbf{q}\| < b$ **then**
            $\mathscr{Q} \leftarrow \mathscr{Q} \cup \{\mathbf{p}_{ij}\}$;
        end if
    end for
    visit wrapped parts, if any;
end for

The first set of experiments have been performed on a subset of 20 point clouds acquired consecutively in order to assess the efficiency of the proposed searching algorithms, `SearchWithinRadius` and `SearchKNearest`. Our approach, henceafter referred as *Lidar*, has been compared with *Brute-force* search and the methods provided by *Flann* [93], a mainstream library for efficent approximate search based on kd-tree. For each point $\mathbf{q}$ of cloud $\mathscr{C}_i$ both radius range and k-nearest searches have been performed on the previous cloud $\mathscr{C}_{i-1}$ (the travelled distance between the two clouds is about 0.50 m). During the different tests the radius r and the number k of

Algorithm 2 SearchKNearest ($\mathscr{C}$, $\mathbf{q}$, k)

Require: $\mathscr{C}$: the input point cloud organized in $w \times h$ items;
$\mathbf{q}$: the query point;
k: the number of neighbors to be found (if exist);
Ensure: $\mathscr{Q}_\mathbf{q}$: the priority queue sorted with decreasing distance from $\mathbf{q}$;
find indices of query point $\mathbf{q}$: $(\hat{i}, \hat{j})$;
if ($\hat{i} < 0$ or $\hat{i} \geqslant h$) or ($\hat{j} < 0$ or $\hat{j} \geqslant w$) **then**
    saturate $\hat{i}$ or $\hat{j}$ s.t. inside domain;
end if
find anchor point $\mathbf{a}$ with indices $(\hat{i}, \hat{j})$;
$\mathscr{D}^{prev} \leftarrow \emptyset$;
$\mathscr{D}^{curr} \leftarrow$ searchWithinRadius($\mathscr{C}, \mathbf{q}, \|\mathbf{a}-\mathbf{q}\|$);
if $|\mathscr{D}^{curr}| < k$ **then**
    expand $\mathscr{D}^{curr}$ s.t. $\mathscr{D}^{curr}$ contains at least k items;
end if
push $\mathbf{a}$ in $\mathscr{Q}_\mathbf{q}$;
while $\mathscr{D}^{curr} \setminus \mathscr{D}^{prev} \neq \emptyset$ **do**
    for all $\mathbf{p} \in \mathscr{D}^{curr} \setminus \mathscr{D}^{prev}$ **do**
        if $|\mathscr{Q}_\mathbf{q}| < k$ **then**
            push $\mathbf{p}$ in $\mathscr{Q}_\mathbf{q}$;
        else if $\|\mathbf{p}-\mathbf{q}\| < \|\text{top}(\mathscr{Q}_\mathbf{q}) - \mathbf{q}\|$ **then**
            pop $\mathscr{Q}_\mathbf{q}$ and push $\mathbf{p}$ in $\mathscr{Q}_\mathbf{q}$;
        end if
    end for
    visit wrapped parts, if any
    $\mathbf{a} \leftarrow \text{top}(\mathscr{Q}_\mathbf{q})$;
    new anchor
    $\mathscr{D}^{prev} \leftarrow \mathscr{D}^{curr}$;
    $\mathscr{D}^{curr} \leftarrow$ searchWithinRadius($\mathscr{C}, \mathbf{q}, \|\mathbf{a}-\mathbf{q}\|$);
    if $|\mathscr{D}^{curr} \setminus \mathscr{D}^{prev}| < k - |\mathscr{Q}_\mathbf{q}|$ **then**
        expand $\mathscr{D}^{curr}$ s.t. $\mathscr{D}^{curr}$ contains at least $k - |\mathscr{Q}_\mathbf{q}|$ items;
    end if
end while

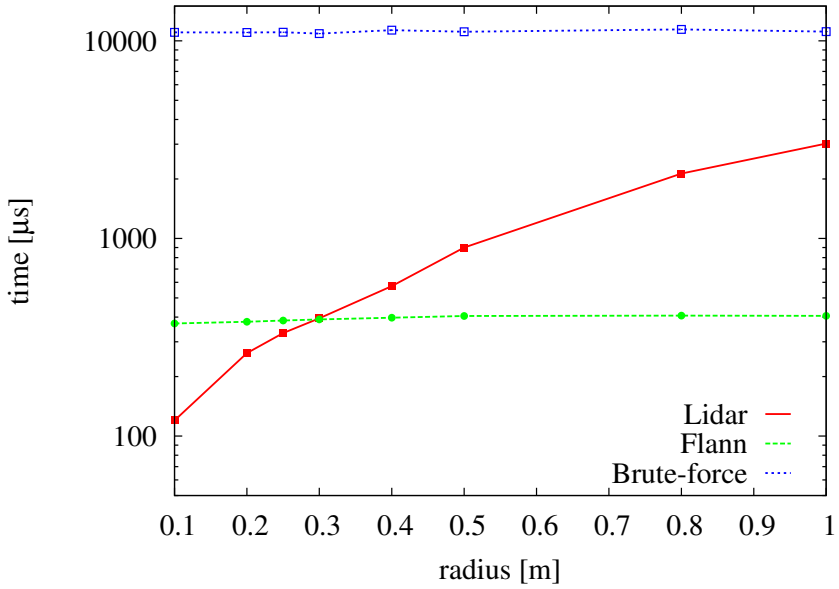


Figure 2.1: Average times of radius search performed using different methods: *Lidar*, *Flann* and *Brute-force* for different values of radius search.

nearest neighbors to find are varied. The times required for the two procedures are shown respectively in Figures 2.1 and 2.2. It can be observed that the execution time of *Lidar* is rather sensitive to the radius and for short distances it performs search more efficiently than *Flann*. The k-nearest search is less sensitive to the number of neighbors to be found, since it is essentially a spherical neighborhood search with radius equal to the candidate nearests and the candidate are found approximately at the same distance.

The proposed searching procedures are derived from bounds on the angular values intersecting a given spherical neighborhood. The points of the cloud in the neighborhood and the k-nearest of a query point are efficiently found by limiting the interval of the indices. The second contribution is the proposal a feature detector for the identification of high-curvature interest points presented in chapter 3. Also the feature extraction algorithm exploits the organization of the LIDAR measurements. The

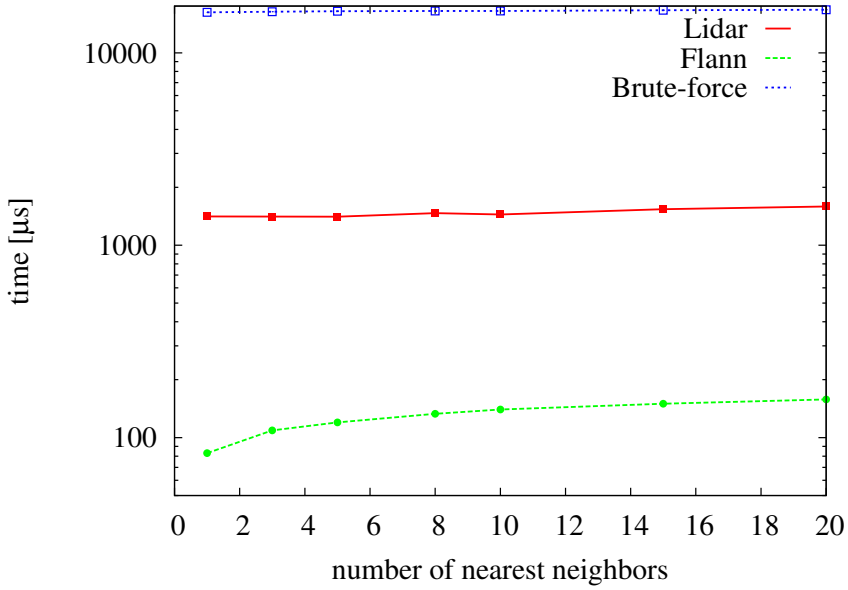


Figure 2.2: Average times of radius search performed using different methods: *Lidar*, *Flann* and *Brute-force* for different number of neighbors k.

proposed procedure operates according to the induced row-wise order of points to select and score the cornerness of each point. Experiments on datasets have assessed the efficiency of the proposed algorithms and the stability of the detected point of interests.

2.3 Place Recognition with Multi-layer Lidar

Place recognition with 3D LIDAR involves using a 3D LIDAR sensor to scan the environment and identify specific places or locations. This can be useful in a variety of applications, including autonomous vehicle navigation, building and facility management, and virtual reality. Place recognition with 3D LIDAR can be used to reduce drift in an autonomous system, such as an autonomous vehicle, by providing a means of accurately determining the location of the system in the environment.

Drift refers to the accumulation of errors in the system's position and orientation over time, which can result in the system becoming misaligned with its surroundings. By using 3D LIDAR to scan the environment and compare the resulting point cloud to a database of known locations, the system can determine its position and orientation with a high degree of accuracy. This can help to reduce drift by providing a more accurate estimate of the system's position and orientation, which can be used to correct for any errors that may have accumulated over time. However, that place recognition with 3D LIDAR is not a complete solution to drift. Other factors, such as sensor noise and errors, can also contribute to drift, and it may be necessary to use additional techniques, such as inertial measurement units (IMUs) or odometry, to further reduce drift.

Techniques for place recognition with 3D LIDAR include using visual features such as texture, color, or shape, or using machine learning algorithms to classify the scanned point clouds. Not only outdoor but the case of indoor place recognition with 3D LIDAR involves using a 3D LIDAR sensor to scan the interior of a building or facility and identify specific locations or places within the structure. This can be useful for a variety of applications, such as facility management, indoor navigation, and virtual reality. Some of the solutions for place recognition with 3D LIDAR, including:

- Database comparison: One common approach to place recognition with 3D LIDAR is to use a database of pre-scanned 3D LIDAR point clouds for comparison. The point cloud captured by the LIDAR sensor in the current environment is compared to the point clouds in the database, and the location is identified based on the closest match.

- Visual features: Another approach is to use visual features such as texture, color, or shape to identify specific locations. These features can be extracted from the 3D point cloud and used to classify the scanned data.

- Machine learning: Machine learning algorithms can be used to classify the 3D point clouds and identify specific locations. This can be effective, particularly when combined with other techniques such as database comparison or visual feature extraction.

- Fusion of multiple sensors: Place recognition with 3D LIDAR can be improved by using multiple sensors, such as cameras or IMUs, to provide additional data about the environment. This data can be fused with the 3D LIDAR data to improve the accuracy and robustness of the place recognition system.

It is important to carefully evaluate any place recognition system using 3D LIDAR data to ensure that it is accurate and robust enough for the intended application.

Place recognition with 3D LIDAR can be challenging due to the complexity of the 3D data, the variability of the environment, and the need for robust algorithms to handle noise and errors in the data. Therefore, it is important to carefully design and evaluate any place recognition system using 3D LIDAR data. There are several challenges involved in place recognition with 3D LIDAR, including:

- Complexity of 3D data: 3D LIDAR sensors capture a large amount of data in the form of a point cloud, which can be difficult to process and analyze.
- Variability of the environment: The appearance of a location can change significantly over time due to various factors such as lighting conditions, weather, and the presence or absence of objects or people.
- Noise and errors in the data: LIDAR sensors can be affected by noise and errors due to various factors such as reflections, occlusions, and hardware limitations.
- Lack of standardization: There is no standard format for storing and sharing 3D LIDAR point clouds, making it difficult to compare data from different sensors or systems.

To address these challenges, it is important to carefully design and evaluate any place recognition system using 3D LIDAR data. This may involve using techniques such as data filtering and noise reduction, using machine learning algorithms to classify the data, or developing robust algorithms to handle the complexity and variability of the 3D data.

There has been a significant amount of research on place recognition using LIDAR sensors. Here are a few examples of research papers that address place recog-

nition using LIDAR data. Place recognition is a crucial problem in robotics that involves identifying the location of a robot based on sensory data. This capability is essential for the robot to navigate and complete tasks. Additionally, place recognition can be used in simultaneous localization and mapping (SLAM) [94], [95] [96] systems to detect when a robot has returned to a previously visited location, which is known as loop closure. This allows the robot to correct errors in its estimated pose and improve the accuracy of its map. In large-scale environments, long-term simultaneous localization and mapping (SLAM) can suffer from error accumulation, which leads to inconsistencies in the mapping results. To address this issue, the use of a place recognition method to detect loop closures can be helpful. Loop closure detection is the process of identifying when a robot has returned to a location it has previously visited, and it is essential for eliminating accumulated errors in the SLAM process.

As computer vision technology has advanced in recent years, image-based place recognition has become a highly effective method for identifying locations. [97], [98], [99], [100]. While image-based place recognition can be effective in many situations, it can be challenged by changes in illumination and camera viewpoint. This can make it difficult to use in challenging environments such as low light conditions. On the other hand, 3D LIDAR sensors can directly capture geometric information about the environment with high precision, and are less sensitive to changes in illumination. They also have a wider field of view compared to cameras. As a result, there has been a significant amount of research on using 3D LIDAR [101], [102], [103], [104] for place recognition in various environments.

2.3.1 Scan Context

Inspired by the method Scan Context [105], the aim is to develop a method for place recognition using multi-layer LIDAR. In this work, they introduced a new approach for representing the location of a 3D scan using a matrix-based place descriptor called Scan Context. This method builds upon the Shape Context technique [106] and is specifically designed for use with 3D LIDAR scan data in place recognition tasks. In urban areas, it is common for vehicles to make frequent revisits and lane changes

due to the high traffic and busy nature of the environment or a robot moving in an industrial setup. This can be a challenge for robots, as they need to be able to navigate the complex and dynamic traffic patterns in these areas in order to operate safely and efficiently. Scan Context is a place descriptor that has been proposed for use in outdoor place recognition tasks. It is designed to encode a point cloud of a 3D scan into a matrix, which can be used to represent the location of the scan. This method for encoding the shape of a point cloud into an image differs from previous approaches in that it uses the maximum height of points in each bin, rather than simply counting the number of points. This allows us to more accurately capture the geometrical shape of the point cloud around a local keypoint.

To begin the process of encoding a 3D scan into an image using this method, first divide the scan into equally spaced bins in the sensor coordinate system. These bins are separated in the azimuthal and radial dimensions. This allows us to evenly sample the 3D scan and capture its shape in a more detailed and accurate manner. A 3D scan using an image-based descriptor is based on the concept of a global keypoint, which is located at the center of the scan. Because the descriptor is centered on this keypoint, we refer to it as an egocentric place descriptor. This means that it encodes the shape of the point cloud relative to the position of the keypoint, rather than using a fixed reference frame. The scan is divided into a number of sectors $\mathbf{R}_i$ and rings $\mathbf{R}_J$. These divisions allow us to evenly sample the point cloud and capture its shape in a detailed and accurate manner. The specific number of sectors and rings used will depend on the characteristics of the scan and the desired level of detail in the resulting descriptor. divide the scan into a grid of bins, each of which represents a specific region of the point cloud. The points belonging to a particular bin $\mathbf{P}_{ij}$ are those that fall within the region where the ith ring and jth sector overlap.

The Scan Context procedure takes two inputs: a point cloud $\mathscr{P}$ and a map $\mathscr{M}$ as in Algorithm 3. The point cloud $\mathscr{P}$ is a set of 3D points, and the map $\mathscr{M}$ is a set of scans, each of which is a subset of points from the point cloud. The procedure returns a list $\mathscr{F}$ of features for the comparison between all scans in the map. The Scan Context procedure first initializes an empty list $\mathscr{L}_p$ to store local features for each point, and an empty list $\mathscr{L}_s$ to store scan context features for each scan. It then iterates through

Algorithm 3 Scan Context for Place Recognition

```
1: function SCAN CONTEXT(𝒫, ℳ)
2:     // Initialize empty list to store comparison features
3:     ℱ ← empty list
4:     // Initialize empty list to store local features for each point
5:     ℒ_p ← empty list
6:     // Initialize empty list to store scan context features for each scan
7:     ℒ_s ← empty list
8:     // Compute local features for each point
9:     for each point p ∈ P do
10:        l_p ← COMPUTELOCALFEATURES(p)
11:        // Append local features to list
12:        ℒ_p.append(l_p)
13:    end for
14:    // Compute scan context for each scan
15:    for each scan s ∈ M do
16:        l_s ← COMPUTESCANCONTEXT(s, ℒ_p)
17:        ℒ_s.append(l_s)
18:    end for
19:    // Compare each scan to all other scans in the map
20:    for each scan s ∈ ℳ do
21:        f ← COMPARESCANS(s, ℒ_s)
22:        ℱ.append(f)
23:    end for
24:    return ℱ
25: end function
```

each point p in the point cloud $\mathscr{P}$, and calls the `computeLocalFeatures()` procedure (Algorithm 4) to compute the local features for that point. The local features may include, for example, the normal vector of the point, the range (distance from the origin) of the point, and the angle between the point and its normal vector. The local features are appended to the list L_p. Next, the Scan Context procedure iterates through each scan s in the map $\mathscr{M}$, and calls the `computeScanContext()` procedure to compute the scan context for that scan. The scan context may include,

Algorithm 4 Scan Context subroutines

function COMPUTELOCALFEATURES(p)
    // Initialize empty list to store local features for point
    $l_p \leftarrow$ empty list
    $n \leftarrow$ COMPUTENORMAL(p)
    // Compute range (distance from origin) for point
    $r \leftarrow$ COMPUTERANGE(p)
    $a \leftarrow$ COMPUTEANGLE(p, n)
    $l_p.append(n)$
    $l_p.append(r)$
    $l_p.append(a)$
    return l_p
end function
function COMPUTESCANCONTEXT($s, \mathscr{L}_p$)
    // Initialize empty list to store scan context features
    $l_s \leftarrow$ empty list
    $c \leftarrow$ COMPUTECENTROID(s)
    $l_s.append(c)$
    // Append all local features for points in scan to list
    for each local feature $l_p \in L_p$ **do**
        $l_s.append(l_p)$
    end for
    return l_s
end function

for example, the centroid (center) of the scan, the local features of all points in the scan, and a histogram of the local features. The scan context features are appended to the list $\mathscr{L}_s$. Finally, the Scan Context procedure iterates through each scan s in the map $\mathscr{M}$ again, and calls the compare scans procedure to compare the scan to all other scans in the map. The comparison features may include, for example, the distance between the scans. The comparison features are appended to the list $\mathscr{F}$.The `computeLocalFeatures()` procedure takes a point p as input, and returns a list l_p of local features for that point. It first initializes an empty list l_p. It then com-

putes the normal vector n of the point using the `computeNormal()` function, the range r of the point using the `computeRange()` function, and the angle a between the point and its normal vector using the `computeAngle()` function. It appends the normal vector, range, and angle to the list l_p, and returns the list. The `computeScanContext()` procedure takes a scan s and a list L_p of local features for each point in the scan as input, and returns a list l_s of scan context features for the scan. It first initializes an empty list l_s. It then computes the centroid c of the scan using the `computeCentroid()` function, and appends the centroid to the list l_s. It then iterates through each local feature l_p in the list L_p and appends it to the list l_s. Finally, it computes a histogram h of the local features using the `computeHistogram()` function.

2.4 Discussion

In this chapter, we have discussed algorithms for efficiently finding points and neighbors within a LIDAR point cloud that is organized in a specific way. These algorithms are designed to work effectively with point clouds that are sparse and organized, and they take advantage of the angular values and indices of the points in the cloud to perform efficient searches. One key aspect of the proposed algorithms is the use of bounds on the angular values intersecting a given spherical neighborhood. By limiting the interval of the indices within this neighborhood, it is possible to efficiently locate points within the neighborhood and find the k-nearest points to a query point. These algorithms have been designed specifically for use with LIDAR point clouds, and they have the potential to significantly improve the efficiency of tasks such as feature extraction and localization when applied to these types of point clouds. Overall, the algorithms presented in this chapter provide a useful tool for working with organized and sparse LIDAR point clouds.

In this chapter, we have also discussed the potential of using scan context in place recognition tasks involving 3D point cloud maps. One of the main contributions of this chapter is the suggestion that scan context can improve the efficiency and robustness of place recognition. Specifically, using scan context can help to reduce

the number of false positive matches, which can improve the efficiency of the place recognition process by reducing the need for additional computations. In addition, scan context can provide valuable temporal information about the dynamics of the environment, which can be used to match scans that were collected at different times, even if the environment has changed significantly between those scans. This can improve the robustness of place recognition to changes in the environment, making it more resistant to errors due to changes in lighting, moving objects, or other factors.

There are still many open questions and challenges related to the use of scan context for place recognition, such as how to effectively incorporate this information into place recognition algorithms, and how to handle scenarios where the scan context is noisy or incomplete. However, the results of the studies presented in this chapter suggest that scan context has the potential to be a powerful tool for improving the accuracy and robustness of place recognition in 3D point cloud maps.

Future research in this area could focus on developing more advanced algorithms for incorporating scan context into place recognition, as well as exploring new applications and scenarios in which scan context could be useful. Overall, the use of scan context as an egocentric spatial descriptor for place recognition within 3D point cloud maps represents a promising direction for future research and development.

3 Novel Geometric Features for Multi-layer LIDAR

3.1 LIDAR Features: Curvature-based Keypoints and SKIP

The main contribution of this thesis lies in the SKIP-3D features extracted from the point clouds acquired through 3D LIDARs. Multi-layer LIDARs acquire range measurements by almost simultaneously firing laser beams at different vertical angles, while the sensor head rotates on its axis. The vertical angle of each beam is usually referred to as altitude, whereas the horizontal angle of the head is the azimuth. The altitude resolution is fixed so that the sampling resolution and the FoV are rather limited (usually about $1^\circ - 2^\circ$ and $30^\circ - 50^\circ$ respectively). Conversely, the azimuth resolution is higher and the FoV covers completely the whole 360° turn angle. Thus, the point clouds acquired by 3D LIDARs are sparse and organized in vertical layers. This means that their points are organized in a matrix-like structure. In particular, the rows of said matrix, vertically covering the FoV of the sensor, are called rings. Thus, the point cloud collected by a 3D LIDAR is partitioned into layers that can be processed independently.

This structure of LIDAR data strongly influences the feature algorithms proposed to detect salient regions in sensor data. The underlying assumption is that the points of a layer are samples of a continuous curve. Inside each layer the points are sufficiently dense for shape description and, moreover, can be radially sorted. If the point cloud

is properly sorted and all the measurements including the invalid ones are kept, then the partitioning into layers is implicit.

F-LOAM exploits layer-based processing to detect points in high curvature and in flat regions, which are called respectively *edges* and *surfaces*. The former are found by thresholding a smoothness parameter computed on a sliding window interval of $2\,w+1$ beams (with fixed $w=5$). Let $\mathscr{P}=\{\mathbf{p}_i\}_{i=0,\ldots,n_l-1}$ be the points of one layer (the layer index is omitted for simplicity). The smoothness of a point $\mathbf{p}_i$ is defined as

$$\sigma_i = \frac{1}{2w}\sum_{j=-w}^{w} \|\mathbf{p}_{i+j}-\mathbf{p}_i\| \tag{3.1}$$

The classification of points according to their σ_i is performed on section of $n_l/6$ consecutive measurements. The points with σ_i greater than a threshold (set to 0.1) are marked, but only the first 20 with largest curvature in the sector are labeled as edges. The unmarked points are classified as surfaces. Although such features are not used to directly build feature-based maps, this simple criterion enables classification of points that is effectively exploited by the registration algorithm in point-to-point association, depending on the label.

This smoothness parameter used to discriminate features has several disadvantages. First, its value depends only on the relative distance between the point and its neighbors without taking into account the global shape of the layer. Second, it overlooks that high value of σ_i may be caused by occlusions or other sensor limitations rather than high curvature. Occlusion produces discontinuity in range values or gaps that violate the implicit assumption that points are sampled from a continuous curve.

3.2 Edge and Planar Feature Point Extraction

The proposed SKIP-3D algorithm [107] addresses these issues. SKIP-3D algorithm exploits the multi-layer structure to extract interest point. SKIP operates on layers and searches for salient points, but it implements additional criteria. First, the sequence of points in a layer are split in correspondence to gaps which are due to occlusion or in a region of strong discontinuity and limitation of the FoV. The splitting criterion

Algorithm 5 Removal of gaps in a point cloud ring

function REMOVEGAP($\mathscr{R} = \{\mathbf{p}_i\}_{i=1\ldots n}$, q_{gap}, m_{gap})
  $\mathscr{Q} \leftarrow \emptyset$
  $i_{first} \leftarrow -1$, $i_{last} \leftarrow -1$
  for $\mathbf{p}_i \in 1 \ldots n$ **do**
    if $\mathbf{p}_i = nan$ **then**
      continue
    end if
    $prev(i) \leftarrow i_{last}$
    if $0 \leqslant i_{last} \leqslant n$ **then**
      $next(i_{last}) \leftarrow i$
      $r_{curr} \leftarrow \|\mathbf{p}_i\|$, $r_{last} \leftarrow \|\mathbf{p}_{i_{last}}\|$
      $r_{mid} \leftarrow (r_{curr} + r_{last})/2$
      if $|r_{curr} - r_{last}| < q_{gap} + m_{gap}\ r_{mid}$ **then**
        $score(\mathbf{p}_{i_{last}}) \leftarrow dist(\mathbf{p}_{prev(i_{last})}, \mathbf{p}_{i_{last}}, \mathbf{p}_i)$
        $\mathscr{Q} \leftarrow push(\mathscr{Q}, \mathbf{p}_{i_{last}})$
      else
        $\mathscr{G} \leftarrow \mathscr{G} \cup \{\mathbf{p}_{i_{last}}\}$
      end if
    else
      $i_{first} \leftarrow i$
    end if
    $i_{last} \leftarrow i$
  end for
  if $0 \leqslant i_{last}, i_{first} \leqslant n$ **then**
    $prev(i_{first}) \leftarrow i_{last}$
    $next(i_{last}) \leftarrow i_{first}$
    $r_{curr} \leftarrow \|\mathbf{p}_{i_{last}}\|$, $r_{last} \leftarrow \|\mathbf{p}_{i_{first}}\|$
    if $|r_{curr} - r_{last}| < q_{gap} + m_{gap}\ r_{mid}$ **then**
      $score(\mathbf{p}_{i_{last}}) \leftarrow dist(\mathbf{p}_{prev(i_{last})}, \mathbf{p}_{i_{last}}, \mathbf{p}_{i_{first}})$
      $\mathscr{Q} \leftarrow push(\mathscr{Q}, \mathbf{p}_{i_{last}})$
    else
      $\mathscr{G} \leftarrow \mathscr{G} \cup \{\mathbf{p}_{i_{last}}\}$
    end if
  end if
  return priority queue $\mathscr{Q}$, gap points $\mathscr{G}$
end function

Algorithm 6 Detection of SKIP-3D features

function DETECTSKIP($\mathscr{R} = \{\mathbf{p}_i\}_{i=1\ldots n}$, d_{th}, q_{gap}, m_{gap})
  $(\mathscr{Q}, \mathscr{G}) \leftarrow RemoveGap(\mathscr{R}, (q_{gap}, m_{gap}))$
  while not $empty(\mathscr{Q})$ and $score(top(\mathscr{Q}) < d_{th}$ **do**
    $\mathbf{p}_i \leftarrow pop(\mathscr{Q})$
    // Removal of i: scores of its next and prev are changed
    if $changed(\mathbf{p}_i)$ **then**
      $score(\mathbf{p}_i) \leftarrow dist(\mathbf{p}_{prev(i)}, \mathbf{p}_i, \mathbf{p}_{next(i)})$
      $changed(\mathbf{p}_i) \leftarrow false$, $\mathscr{Q} \leftarrow push(\mathscr{Q}, \mathbf{p}_i)$
    else
      $next(prev(i)) \leftarrow next(i)$
      $prev(next(i)) \leftarrow prev(i)$
      $changed(prev(i)) \leftarrow true$
      $changed(next(i)) \leftarrow true$
    end if
  end while
  // Points still in queue are edges $\mathscr{E}$
  $\mathscr{E} \leftarrow copy(\mathscr{Q})$
  // Surface points $\mathscr{S}$ *as flat points between edge or gap points*
  $\mathscr{U} \leftarrow \mathscr{E} \cup \mathscr{G}$
  sort $\mathscr{U}$ by point index
  for $\mathbf{p}_{i_j} \in \mathscr{U}$ **do**
    define segment $\overline{\mathbf{p}_{i_j}\mathbf{p}_{i_{j+1}}}$
    for $k = i_j + 1 \ldots i_{j+1} - 1$ **do**
      if $dist(\mathbf{p}_k, \overline{\mathbf{p}_{i_j}\mathbf{p}_{i_{j+1}}}) < s_{th}$ **then** $\mathscr{S} \leftarrow \mathscr{S} \cup \{\mathbf{p}_k\}$
      end if
    end for
  end for
  return edges $\mathscr{E}$, surfaces $\mathscr{S}$
end function

is rather simple and based on a threshold of difference between consecutive ranges. The threshold increases with the magnitude of ranges to adapt its value.

The salient points are obtained through bottom-up simplification of the polyline according to the procedure inspired by [108]. The input data consists of a single laser scan with field-of-view of 360 *deg* represented as a curve connecting adjacent points. Each point $\mathbf{p}_i$, except for invalid ranges or gap points, has a previous and next point, respectively $\mathbf{p}_{p_i}$ and $\mathbf{p}_{n_i}$, according to the radial order holding in each LIDAR layer. The procedure to split the ring into intervals at gap points is illustrated by Algorithm 5. The exceptions are the gap points, i.e points in strong range discontinuities due to occlusion and limitation of the FoV. A layer is not represented by a closed curve and the points laying on the gaps are removed from the list of potential points. The saliency of a measurement/point is given by a score. The *cornerness score* of each point is defined as

$$\kappa_i = \|\mathbf{p}_{n_i} - \mathbf{p}_i\| + \|\mathbf{p}_i - \mathbf{p}_{p_i}\| - \|\mathbf{p}_{n_i} - \mathbf{p}_{p_i}\| \tag{3.2}$$

This score increases with the sharpness and saliency of points. SKIP points are estimated by iteratively removing the points with lowest values of cornerness κ_i in order to further refine and obtain more meaningful measurements. Every time a point $\mathbf{p}_i$ is removed from a particular ring, its previous and next points $\mathbf{p}_{p_i}$ and $\mathbf{p}_{n_i}$ respectively changes. At this stage the score is re-calculated in order to put back the consecutive points in relation. The cornerness values evolve during the procedure and are less and less dependent from local neighborhood. Thus, the remaining points at the end of the procedure provide a faithful representation of global shape. Lines 3-15 of Algorithm 6 illustrates the procedure for computation of SKIP points from the ring intervals previously computed. The procedure extracts intervals from a priority queue according to cornerness score. The main data structure is the priority queue $\mathscr{Q}$ containing the points $\mathbf{p}_i$ ordered by increasing score.

The edge points correspond to the SKIP points computed as described above. The points $\mathbf{p}_i$ are classified as surface if the following conditions hold:

1. the radial index i of $\mathbf{p}_i$ belongs to an interval $f \leqslant i \leqslant l$ where $\mathbf{p}_f, \mathbf{p}_l$ are either edge points or gaps;

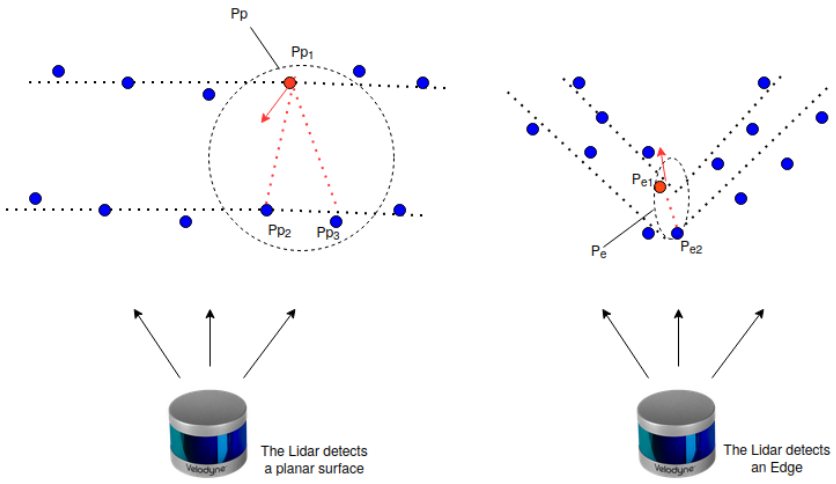


Figure 3.1: SKIP based feature points. The left side shows planar feature point and on the right edge feature is shown

2. the distance between $\mathbf{p}_i$ and segment $\overline{\mathbf{p}_f\mathbf{p}_l}$ is less than a threshold s_{th};
3. the number of points $\mathbf{p}_i$ satisfying the previous conditions on interval $[f,l]$ is greater than a threshold s_{num}.

This very concept of edge and planar feature extraction handles point cloud data which contain limited number of feature points with most of the required information instead of considering a whole point cloud and thus reduce the computation load. This idea is explained below with the help of Figure 3.1.

Lets consider the simplified expression of smoothness c of point $\mathbf{p}_i$ on a single scan ring as below

$$c = \sum_{\mathbf{p}_j \mathscr{P}, j \neq i} \|\mathbf{p}_i - \mathbf{p}_j\| \tag{3.3}$$

Where $\mathscr{P} = \{\mathbf{p}_j\}_{j=0,\ldots,n_l-1}$ a collection of j points and $\mathbf{p}_i$ is current point in a layer. After calculating the score value for each point the points are sorted with regards to c value. An Edge point is chosen with the highest value of c and planar points with

lower c value. In order to find a smooth or planar surface $\mathbf{P}_p$ formed by three planar points on two nearest single scan ring like $\mathbf{P}_p = \{\mathbf{P}_{p_1}, \mathbf{P}_{p_2}, \mathbf{P}_{p_3}\}$. Among the three points, along with the surface normal one of the point is refereed as a planar feature. The edge feature $\mathbf{P}_e$ can be extracted by considering two nearest edge points on two close by rings for example $\mathbf{P}_e = \{\mathbf{P}_{e_1}, \mathbf{P}_{e_2}\}$. This process is repeated for all scan rings to collect features.

In summary, surface patches consist of a sufficiently numerous sequence of smooth (aligned) points between two salient points. The classification achieved by F-LOAM original features and SKIP is homogeneous, but the criteria to compute edges and surfaces are significantly different.

3.3 LIDAR Odometry and Mapping

Zhang [74] proposed a real-time solution to LOAM by using LIDAR odometry to estimate velocity at a higher frequency, while the mapping performs fine processing to create maps at a lower frequency, also using feature matching to ensure fast computation in the odometry algorithm, and to enforce accuracy in the mapping algorithm.

F-LOAM [77] is presented as an extension of the original LOAM, as it aims to be a lightweight system for estimation of sensor odometry and mapping, willing to use faster computational approaches, specifically while compensating the distortion of previously extracted features. These features can belong to edge or surface sets, according to their computed local curvature smoothness. The improvements regarding computational complexity become particularly interesting when noting that modern LIDARs can produce point clouds with frequencies close to 20 Hz.

Both LOAM and F-LOAM are used to estimate motion and position of the sensor, which are then used to estimate mutual poses. They also produce the complete map through registration, accumulation and refinement of previous LIDAR measurements. Figure 3.2 shows a diagram illustrating the different steps of F-LOAM, also including steps that will be analyzed in the following section. The algorithm can be split into separate blocks. The LIDAR yields a raw point cloud $\mathscr{P}$. The point cloud is then organized by a property called sweep. A sweep is 360 degrees rotation produces a

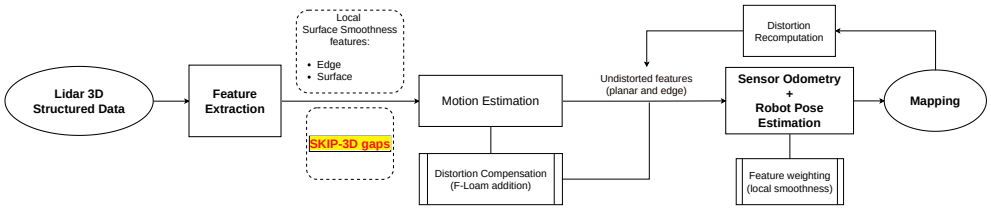


Figure 3.2: (F-)LOAM algorithm steps, including SKIP-3D addition

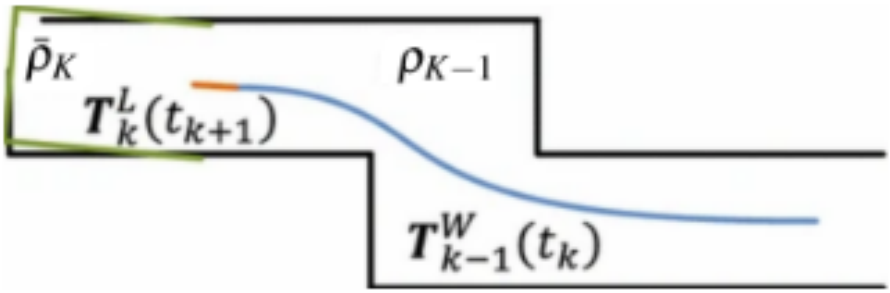


Figure 3.3: The pose on the map, $T_w^{K-1}(t_k)$, is depicted by the blue curve belonging to the sweep $k-1$. In the same way, the LiDAR motion estimated by the odometry algorithm is represented with the orange curve, equivalent to an entire sweep k, $T_L^K(t_{k+1})$. Having the pose on the map $T_w^{K-1}(t_k)$, the odometry motion $T_L^K(t_{k+1})$, the undistorted point cloud $\bar{P}_k$ can be published on the map, represented by $\bar{\rho}^K$ and the green segments. Finally, this cloud is matched to the existing cloud from the map, ρ^{K-1} depicted by the black segments. adapted from [Zhang, J. and Singh, S.] [74].

point cloud. As the Velodyne LIDARs are collection of different independent lasers (VLP-16, HDL-32, HDL-64), a single scan is collection of points returned by a single laser. Subsequently, the features belonging to successive 3D scans are compared in order to find the optimal transform between two consecutive point clouds. This is the core task of the system and its accuracy depends on stability and number of the estimated features. The LIDAR Mapping block repeatedly receives data at 1 Hz from the LIDAR odometry block. This frequency indicates how fast the map is generated.

The mapping process is shown in Figure 3.3. The LIDAR pose on the map is

characterized by the curve. In comparison to LIDAR odometry module the mapping module is called once per sweep and runs at a frequency 10 times lower. After a complete sweep an un-distorted point cloud is generated

3.4 Pose Estimation and Distortion Compensation

This section addresses the estimation of poses between the current LIDAR scan and the map built accumulating previous scans based on features. LIDAR points are labelled as edge or surface features as discussed in the next section. The distortion in LOAM [74] is corrected from one point cloud to another point cloud by comparing and estimating the transformation among them. To find the transformation between successive scans a frequent computation is needed which is not so efficient. In opposition to LOAM the distortion compensation in F-LOAM is double stage distortion compensation which enables the algorithm to minimize the computational cost. The scanning time of 3D LIDARs are very short as they are capable to scan the surrounding at higher frequency (10 Hz) resulting often short time between two consecutive scans. considering constant velocities at the first stage between two scans. Estimate the motion and correct the distortion. During the second stage after the pose estimation the distortion is re-calculated and the corrected features are added to the final map. The un-distort features are used for pose estimation. The pose estimation takes the obtained corrected features after finding the transform between two successive scans aligns them with global feature map. It is important to minimize the searching computational cost, for this reason the features map are stored in KD-trees.

For each cluster of neighbor edge points the algorithm computes a covariance matrix and the edge direction $\mathbf{n}_e$ is equal to the largest eigenvalue of that covariance matrix. Each edge point $\mathbf{p}_e$ is associated to an edge point $\mathbf{p}_e^g$ with its edge direction $\mathbf{n}_e^g$ in the global map (superscript ^{g} refers to the reference frame of the global map). Hence, the distance between edge point $\mathbf{p}_e$ and its corresponding global map feature $\mathbf{p}_e^g$ is defined as

$$\mathscr{F}_e(\mathbf{p}_e) = \mathbf{p}_n^\top \cdot ((\mathbf{T}_k\, \mathbf{p}_e - \mathbf{p}_e^g) \times \mathbf{n}_e^g) \tag{3.4}$$

where $\mathbf{T}_k$ is the transformation matrix representing the k-th robot pose with respect

to the global frame and the unit vector $\mathbf{p}_n$ is given by

$$\mathbf{p}_n = \frac{(\mathbf{T}_k\ \mathbf{p}_e - \mathbf{p}_e^g) \times \mathbf{n}_e^g}{\| (\mathbf{T}_k\ \mathbf{p}_e - \mathbf{p}_e^g) \times \mathbf{n}_e^g \|} \tag{3.5}$$

Similarly, each cluster of surface points $\mathbf{p}_s$ has a covariance matrix and plane norm $\mathbf{n}_s$ corresponding to the eigenvector associated to its smallest eigenvalue. The surface points and plane norm in the global map are labeled respectively as $\mathbf{p}_s^g$ and $\mathbf{n}_s^g$. The distance between a planar feature $\mathbf{p}_s$ and its associated map feature $\mathbf{p}_s^g$ is equal to

$$\mathscr{F}_s(\mathbf{p}_s) = (\mathbf{T}_k \mathbf{p}_s - \mathbf{p}_s^g)^\top \cdot \mathbf{n}_s^g \tag{3.6}$$

The transformation $\mathbf{T}_k$ is found by minimizing the sum of the distances of edge and surface features between the current scan and the global map.

When the transnational or rotational change is greater than already set threshold a new keyframe is obtained. Each new keyframe is used to update the global map which consist of edge global map and planar global map. As mentioned in the beginning of this section the key improvement regarding the computational cost is reduced with the help of keyframed map rather than comparing frame by frame update.

The motion estimation part of the system is kept less complicated by assuming that the LIDAR sensor is moving with constant velocity. Transformation matrix is obtained for the end point of a data frame relative to the start point. Each point in the data frame can be obtained by time interpolation which allows to estimate the pose transform for points that are being received at different time. The interpolation can be found by using the following equation;

$$\mathbf{T}^R_{(j+1,i)} = \frac{t_i - t_j + 1}{t - t_j + 1} \mathbf{T}^R_j + 1 \tag{3.7}$$

Where the $\mathbf{T}^R_{j+1}$ is the pose transform between $\mathbf{T}_{j+1}$ and t, it also carries the rigid motion in 6-DOF. A time stamp with a given point is denoted by t_i. Since the LIDAR pose is in continuous motion the interpolation method take into account the current time t transform.

Hence, the correspondence between points in current data frame and previous data frame is obtained by equation

$$\mathbf{X}^R_{(j+1,i)} = \mathbf{R}\tilde{X}^R_{(j+1,i)} + \mathbf{T}^R_{(j+1,i)} \tag{3.8}$$

R represents rotation matrix and a translation matrix is represented by **T**. Rotation matrix is expanded with the help of Rodrigues formula as shown

$$\mathbf{R} = e^{\hat{\omega}\theta} = P + \hat{\omega}\ \sin\theta + \hat{\omega}^2(1 -\ \cos\theta) \tag{3.9}$$

where, **R** is the rotation matrix and θ is the magnitude of rotation. The sensor rotation direction is given by ω, which is a unit vector and $\hat{\omega}$ is the skew symmetric matrix of ω. Once the rotation matrix is derived, the follow up step is to calculate the relevant distances. In the data frame distance from any specific point-to-line and distance from point-to-surface is obtained.

As a complete sweep of 3D LIDAR collects data points in a point cloud and feature points are extracted from the acquired point cloud. For this reason in equation 3.8 $\tilde{X}^R_{(j+1,_i)}$ represent the coordinates of a point P in a collection of feature points (edge and plane points). In the collection of points $\tilde{X}^R_{(j+1,_i)}$ are the re-projected corresponding points to the beginning of a sweep. In the translation part this $\mathbf{T}^R_{(j+1,i)}$ corresponds to the coordinates of feature points.

For optimization a non linear error function is obtained taking into consideration LIDAR motion and Levenberg Marquardt (LM) method.

$$f(\mathbf{T}^R_{(l+1)}) = d \tag{3.10}$$

Here, every rows of the function f contains a feature point. Each row in f represents a feature point, and the Jacobian matrix is calculated as

$$\mathbf{T}^R_{(l+1)} \rightarrow \mathbf{T}^R_{(l+1)} - (J^T J + \lambda diag(J^T J)^{-1} J^T d) \tag{3.11}$$

where, J is the jacobian matrix and can be calculated as $J = \frac{\partial f}{\partial T^R_{j+1}}$ and λ is LM factor adjusted at each iteration.

3.5 Experiments

The experiments presented in this section has been designed to assess the performance of the registration algorithm with the proposed features. Tests have been performed on both datasets either indoor or outdoor.

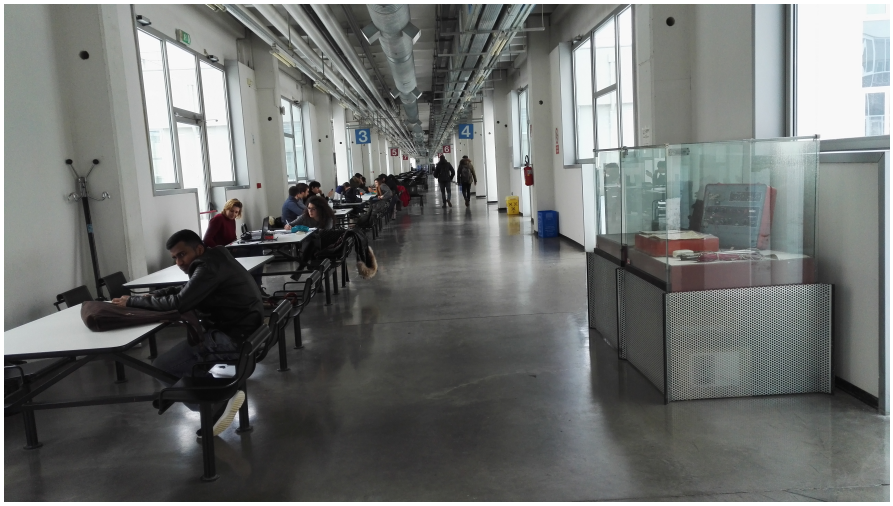


Figure 3.4: Environment where the indoor dataset DIA used in the experiments have been obtained.

The indoor dataset UNIPR-DIA has been acquired by the authors in the main hallway of the Dipartimento di Ingegneria e Architettura of the University of Parma, which consists of a long corridor with branches and tables (see Figure 3.4).

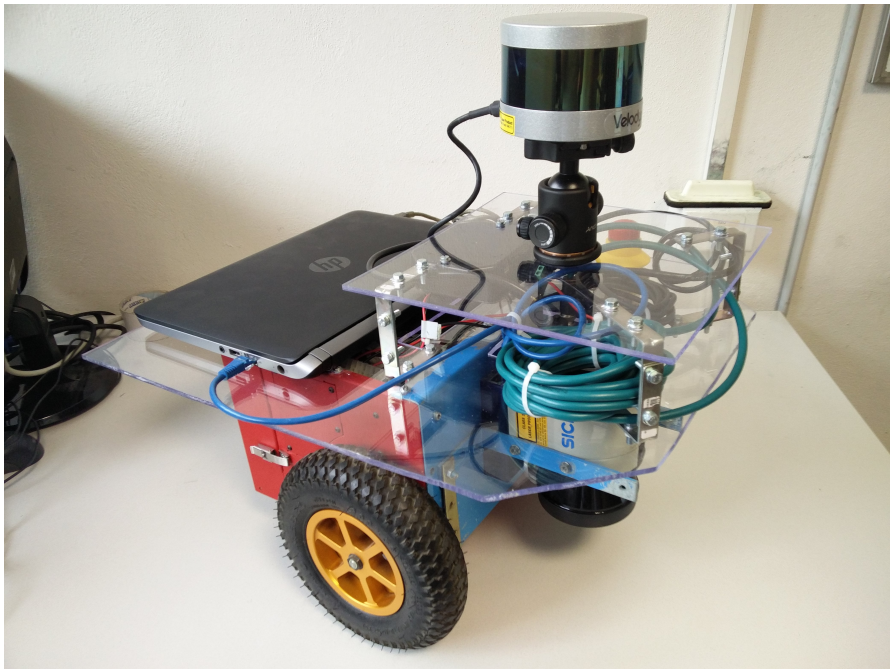


Figure 3.5: Pioneer 3DX robot equipped with the Velodyne VLP-16.

The Pioneer 3DX robot equipped with the Velodyne VLP-16 sensor can be seen in Figure 3.5 has been teleoperated and collected a dataset of consecutive LIDAR point clouds. During the teleoperation the operator and other moving people have been captured in the measurements.

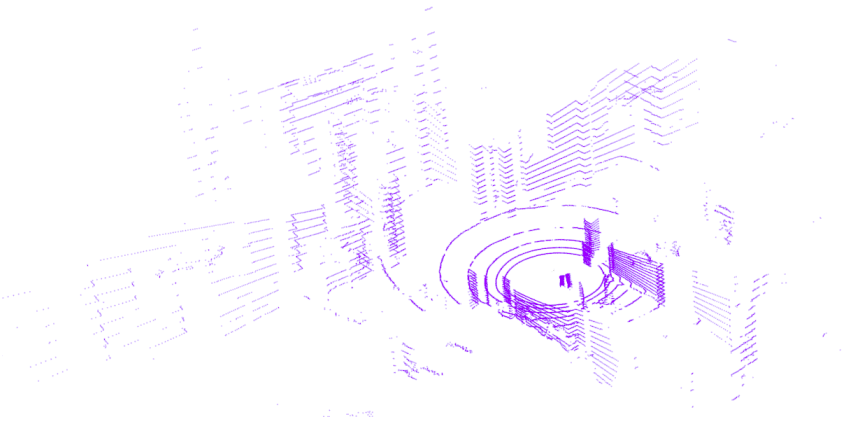


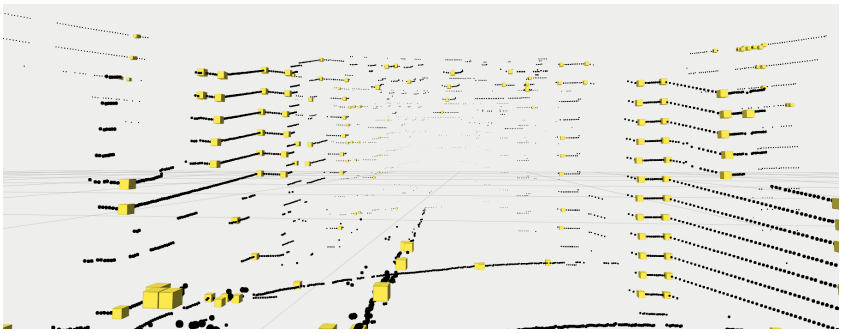
Figure 3.6: Point Cloud obtained with Pioneer 3DX robot equipped with Velodyne VLP-16.

The acquired dataset consists of 384 scan clouds, each containing 29184 points (including the invalid measurements) organized into 16×1824 matrix. Figure 3.6 shows a single scan of VLP-16 (sixteen layers) collected with Pioneer 3DX.

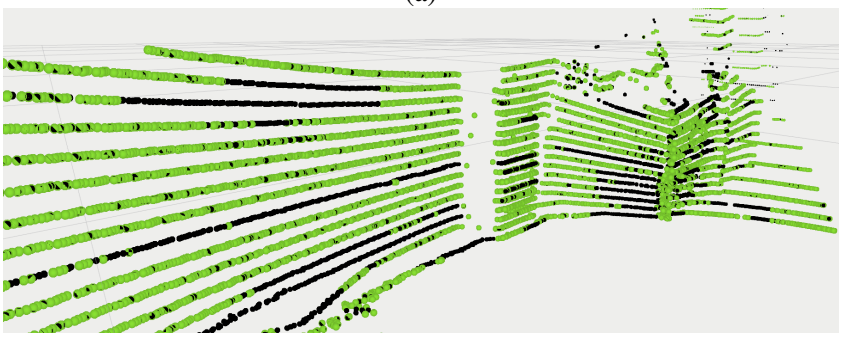
KITTI dataset [109], along with several other datasets that will be mentioned, have been utilized for testing in outdoor settings. We only used the data collected by Velodyne HDL-64, a multi-layer LIDAR with 64 layers. The dataset also provides accurate groundtruth. Henceafter, the F-LOAM algorithm using its original feature extractor will be referred to as *orig* algorithm and the F-LOAM algorithm using the SKIP feature extractor as *skip*. The experiments have used the implementation of F-LOAM provided by the authors* and the implementation of SKIP-3D feature detector.

*https://github.com/wh200720041/floam.

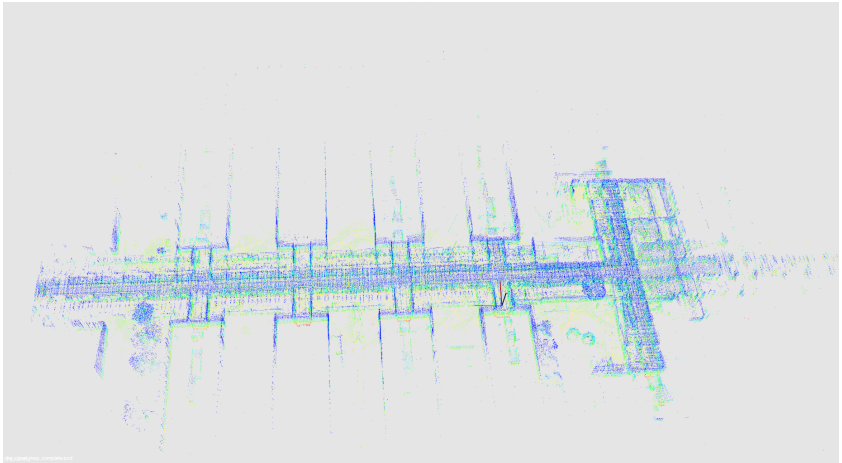
3.5.1 Indoor datasets



(a)



(b)



(c)

Figure 3.7: Overview of F-LOAM with SKIP-3D features: (a) Yellow points represents an example of SKIP-3D edge points; (b) example of SKIP-3D surface points shown with green points; (c) the complete map of indoor DIA dataset estimated using F-LOAM and SKIP-3D.

These experiments qualitatively compare the trajectory obtained with *orig* and *skip* in indoor environments, where we expect to achieve effective registration due to regularity of building structures. Figures 3.7(a) and 3.7(b) show example of respectively SKIP-3D edge and surface points obtained in UNIPR-DIA. Edges are often detected on pillar borders or other sharp structures whereas surfaces lies on concrete and glass walls. The complete map of UNIPR-DIA obtained with algorithm *skip* is displayed in Figure 3.7(c).

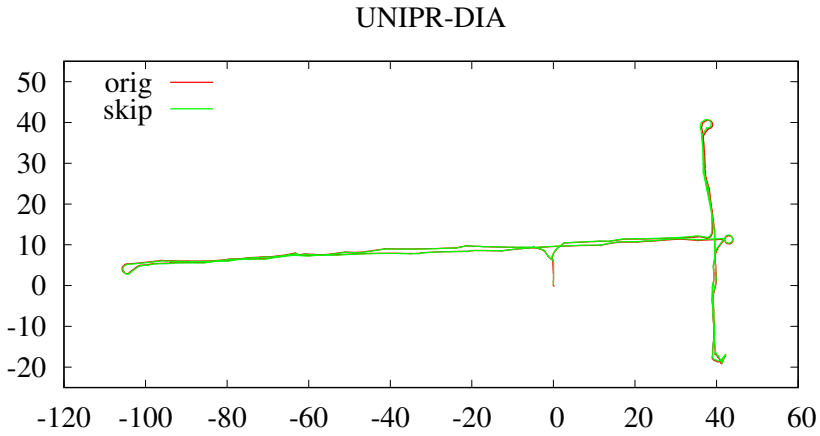


Figure 3.8: Estimated trajectories obtained with F-LOAM original and FLOAM-SKIP are plotted in red and green color respectively

There is no groundtruth to compare the path estimated by *orig* and *skip*, but the two outcomes can be qualitatively compared as shown in Figure 3.8. The paths of *orig* (red line) and *skip* (green line) largely overlap and are almost indistinguishable.

3.5.2 Outdoor datasets

The performance of F-LOAM with *orig* and *skip* has also been assessed on several outdoor datasets.

1. **KITTI** is a benchmark largely used by robotic and computer vision community. We selected the sequences 01, 02, 05 and 07 and used only the data related to point clouds acquired by the LIDAR sensor. The point clouds have been published with rate 10 Hz comparable to the acquisition rate to simulate the online execution of simultaneous odometry and mapping tasks. KITTI dataset also provide the groundtruth of the trajectories.

2. **ICSENS**. The *i.c.sens Visual-Inertial-LIDAR* [110] [†] is provided by a group from the University of Hannover. It has been acquired using Velodyne HDL-64 LIDAR (64 layers, vertical resolution 0.5°) at 10 Hz rate. The sensor is mounted on a car vehicle with axis x facing forward driving direction, axis y toward left and axis z upward. The eight sequences labeled as ICENS-01/08 are the result of a 15 minutes drive.

3. **ITU**. The ITU dataset [111] [‡] provided by Istanbul Technical University is collected using a Husky A200 mobile robot equipped with a Velodyne VLP-16 Puck (16 layers, vertical resolution 2°). The ground vehicle traveled in between trees about 174 meters to record the data. We used the first 3 sequences.

4. **STEVENS**. The Stevens-VLP16 dataset [76] provided by Stenvens Institute of Technology is collected using a Clearpath Jackal robot equipped with a Velodyne VLP-16 Puck. The robot has been manually guided on a setting consisting of buildings, trees, roads and sidewalks. We used 3 sequences acquired on June 15th 2017 [§].

As for indoor datasets, we compared F-LOAM *orig* and *skip* in two consecutive trials for each sequence.

[†] `https://data.uni-hannover.de/dataset/i-c-sens-visual-inertial-LIDAR-dataset`
[‡] `https://doi.org/10.25835/0026408`
[§] `https://tinyurl.com/2p87t9w2`

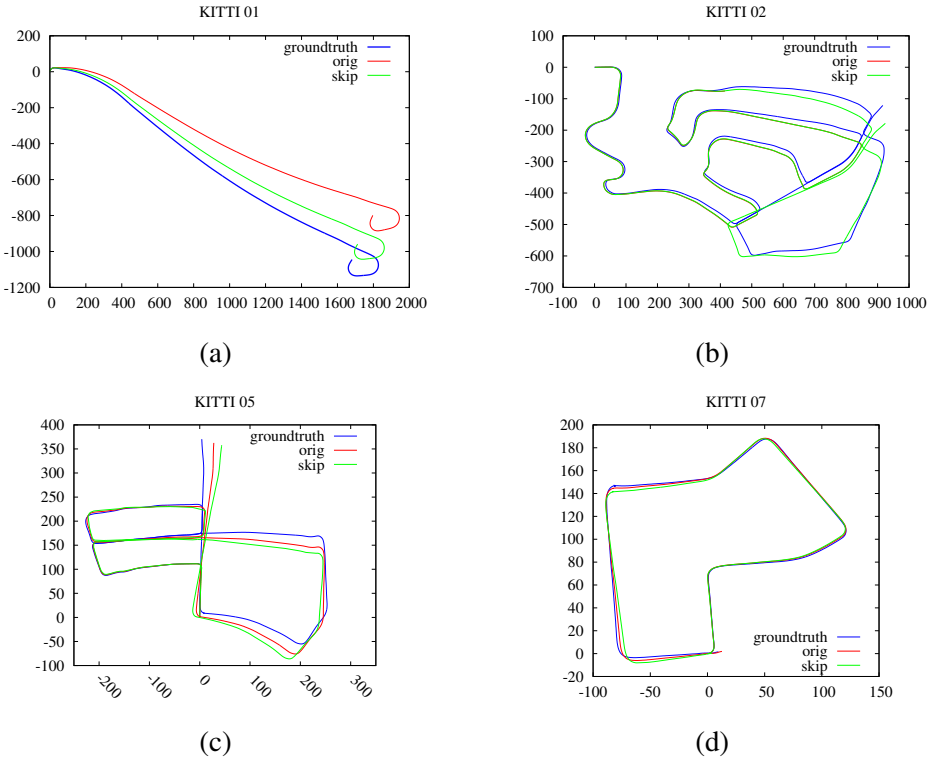


Figure 3.9: Robot trajectories estimated using F-LOAM with Original (red), SKIP features (green) and Groundtruth (blue) for the following KITTI sequences: (a) KITTI-01, (b) KITTI-02, (c) KITTI-05 and (d) KITTI-07. The distances are measured in meters.

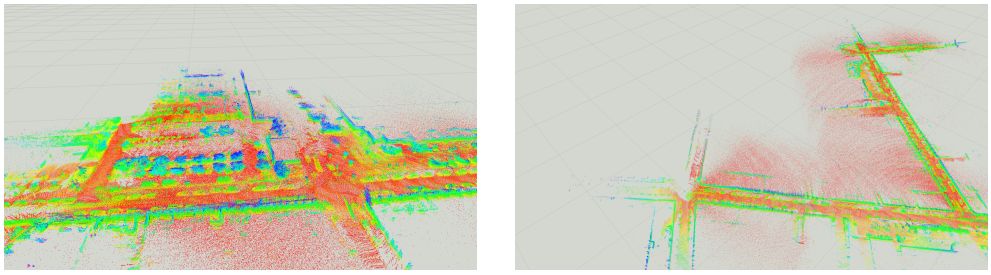
KITTI dataset is distributed with the information about groundtruth and allows more significant comparison. The paths obtained on KITTI sequences are illustrated in Figure 3.9. In particular, each subfigure displays the path estimated with *orig* (red line) and *skip* (green line) as well as the *groundtruth* (blue line), which is provided with KITTI dataset. In these experiments, the robot travels longer paths (magnitude order of unit kilometers) than indoor datasets and F-LOAM only estimates using registration without loop closure. A slight rotation error at some point of the trajectory results in irretrievable propagation of the error to all the next poses. Hence, the drift among *orig*, *skip* and *groundtruth* can be readily observed in the latter segments of

each path, but they are still rather consistent for most part of the robot path. In sequence 07 (Figure 3.9(d)) the path estimated with *orig* prematurely interrupted.

Dataset	F-LOAM *orig*		F-LOAM *skip*	
	ATE [%]	ARE [$10^{-2}{}^{\circ}/m$]	ATE [%]	ARE [$10^{-2}{}^{\circ}/m$]
KITTI-01	2.58	0.67	2.34	0.59
KITTI-02	8.56	4.11	5.12	1.97
KITTI-05	9.89	4.11	63.72	27.20
KITTI-07	2.68	1.74	9.99	5.32

Table 3.1: Average Translational Error (ATE) and Average Rotational Error (ARE) obtained by F-LOAM with features *orig* and *skip* on the given sequences of KITTI dataset.

Table 3.1 reports the ATE (average transnational error) and ARE (average rotational error) [77] obtained from *orig* and *skip*. The groundtruth and estimated paths are aligned according to the travelled distance from initial frame instead of the unavailable sampling time. ATE and ARE are computed on path slices of lengths $100, 200, \ldots, 800\,m$ sampled with steps of $10\,m$. We observe that ATE and ARE are singificantly smaller with *skip* than with *orig*. The only exception refers to sequence 05. We have investigated the reasons for such large ATE and ARE, which seem inconsistent with the paths in Figure 3.9(c). It appears that the reason for potential deviation is the improper alignment of sub-paths based on distances (path slices), but further analysis is required.



(a) (b)

Figure 3.10: Complete maps of (a) ICSENS-01 and (b) ICSENS-02 datasets estimated using F-LOAM and SKIP-3D

The tests on datasets ICSENS, ITU and STEVENS further confirmed the performance of SKIP-3D over the original features. Figure 3.10(a)-(b) displays the maps in the form of point clouds obtained with F-LOAM + SKIP-3D respectively from datasets ICSENS-01 and ICSENS-02. We have computed the paths estimated by F-LOAM with either its original features (hence labelled as *orig* and plotted in red color) and the proposed SKIP-3D (hence labelled as *skip* and plotted in green color).

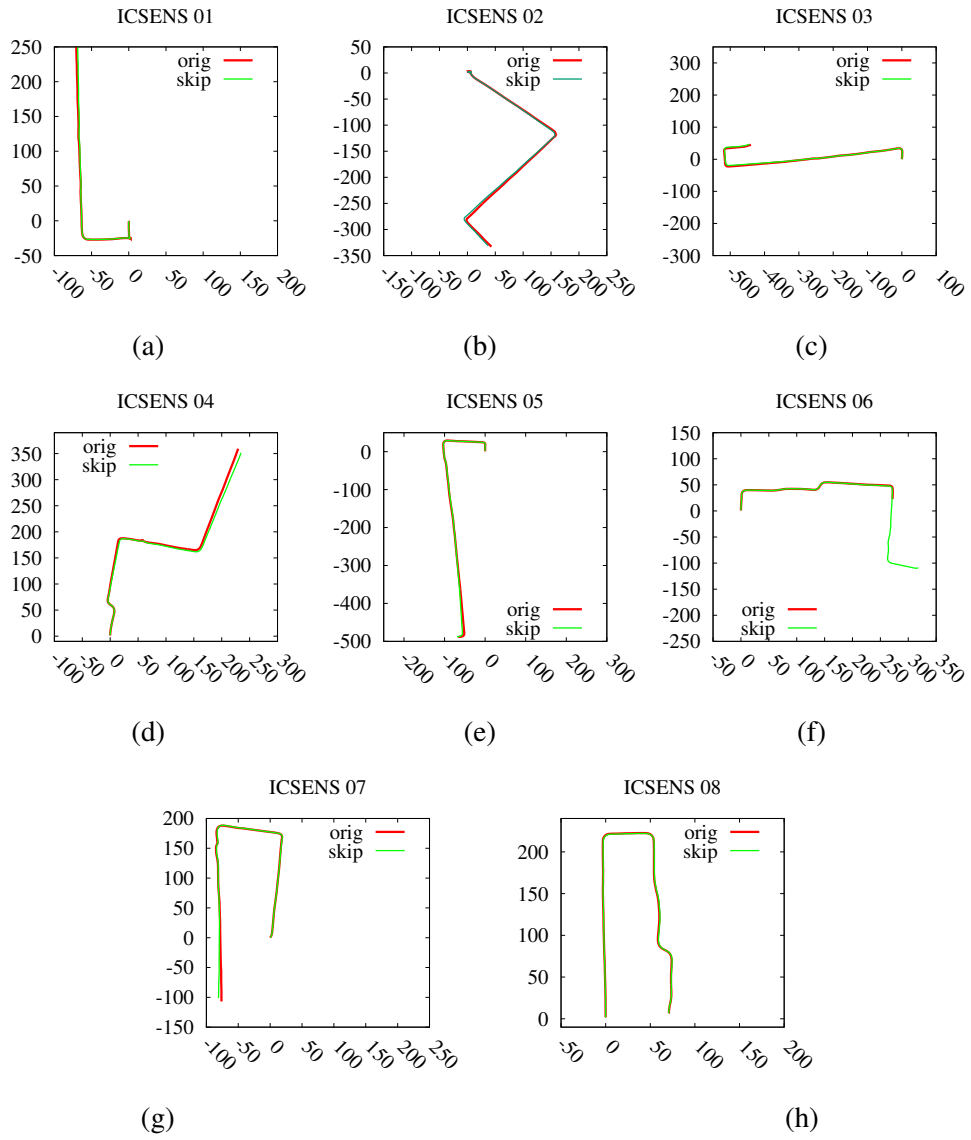
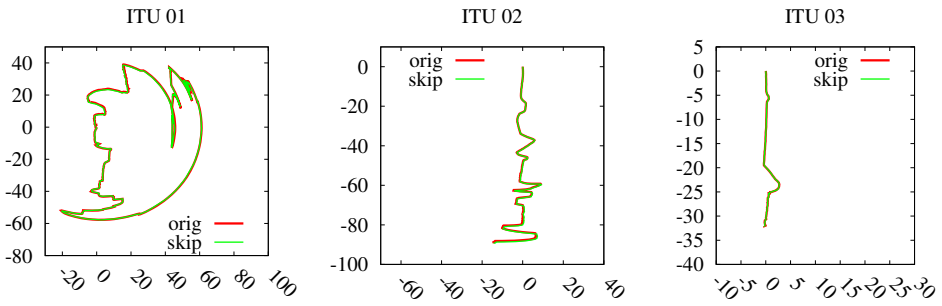
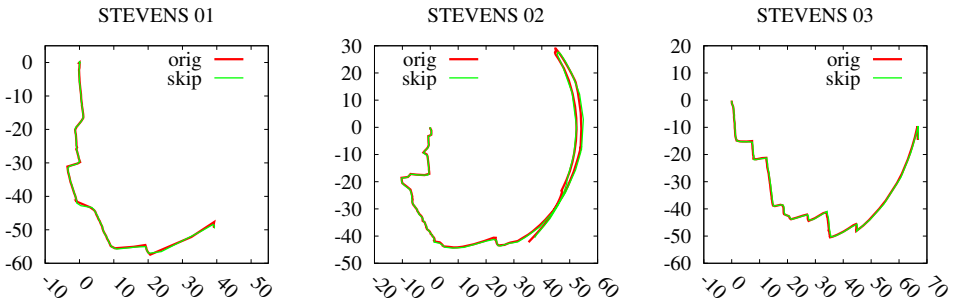


Figure 3.11: Robot trajectories estimated using F-LOAM with Original (red), SKIP features (green) for the following i.c.sens Visual-Inertial-LIDAR Dataset sequences: (a) icsens-01, (b) icsens-02, (c) icsens-03 and (d) icsens-04 (e) icsens-05 (f) icsens-06 (g) icsens-07 (h) icsens-08. Distances are measured in meters.



(a) (b) (c)

Figure 3.12: Robot trajectories estimated using F-LOAM with Original (red), SKIP features (green) for the following ITU sequences: (a) ITU-01, (b) ITU-02, (c) ITU-03. Distances are measured in meters.



(a) (b) (c)

Figure 3.13: Robot trajectories estimated using F-LOAM with Original (red), SKIP features (green) for the following STEVENS sequences: (a) STEVENS-01, (b) STEVENS-02, (c) STEVENS-03. Distances are measured in meters.

Figures 3.11, 3.12 and 3.13 show the outcomes on datasets ICSENS, ITU and STEVENS. In most of cases the paths estimated with *orig* and *skip* largely overlap. An exception is obtained for ICSENS-06 (Figure 3.11(f)) where the path estimated with *orig* breaks up ahead of time and a failure of *orig* is observed. In the STEVENS dataset the environment varies to better assess the performance of features extraction

algorithm namely Edges and Planar surfaces. In these sequences, the points are returned from tree leaves, which are not stable features in the edge features. However, as the robot moves to the building area features become more clear and consistent in the *skip* by adjusting the threshold.

Table 3.2: Average Translational Difference (ATD) and Average Rotational Difference (ARD) obtained by F-LOAM with features *skip* w.r.t. features *orig*.

Dataset	ATD Translation [%]	ARD Rotation [10^{-2}°/m]	Step [m]
ICSENS 01	1.691	1.025	100
ICSENS 02	0.974	1.133	100
ICSENS 03	3.563	1.984	100
ICSENS 04	1.548	0.9971	100
ICSENS 05	1.681	1.064	100
ICSENS 06	1.600	1.369	100
ICSENS 07	1.099	0.698	100
ICSENS 08	1.226	0.7573	100
ITU 01	11.600	15.288	100
ITU 02	9.662	8.863	60
ITU 03	4.963	8.586	20
STEVENS 01	4.783	6.694	40
STEVENS 02	21.178	25.276	40
STEVENS 03	4.159	4.770	40

Table 3.2 reports the ATD (average translational difference) and ARD (average rotational difference) obtained from *orig* and *skip* which indicates to discrepancies

in the estimated paths. In particular, higher ATD and ARD have been noticed with ICSENS-03, ITU-01 and STEVENS-02 which are evident from the Figures 3.11(c), 3.12(a), 3.13(b) as the *orig* and *skip* slightly diverge.

In (STEVENS, ICSENS and ITU) cases, ground truth data are not available and difficult to obtain. This is particularly true for these datasets that involve real-world environments, such as LIDAR point clouds. In many outdoor environments, it can be challenging to accurately label all of the objects in the scene due to factors such as occlusions, lighting variations, and complex object geometries.

3.6 Discussion

In this chapter, I have proposed the novel feature detector SKIP-3D integrated in sensor odometry system F-LOAM for LIDARs. The proposed features effectively estimates points belonging to sharp items and planar patches in the scene, which can substitute the original F-LOAM feature extractor. The input point cloud is processed layer by layer exploiting the organization of multi-layer LIDARs. Salient point detection is obtained by removing less significant points so that at each iteration the general shape of the layer is maintained. SKIP-3D features are processed online and have been integrated with F-LOAM. The proposed and the original features have been compared in robot odometry and mapping tasks performed in indoor and outdoor environments. F-LOAM with SKIP performs similarly or better than the version with the original features and achieves generally lower position and rotation errors.

4 Conclusion

Odometry and mapping are critical elements in all modern SLAM systems, and LIDAR is likely to be a key component in SLAM systems for self-driving vehicles. LIDAR sensors are reliable in a wide range of lighting conditions, making them well-suited for self-driving systems that need to ensure safety. Additionally, competition among LIDAR manufacturers has made these sensors more affordable for this application. This thesis presents feature extraction algorithm for lidar odometry and mapping algorithm and exploits the organizaion of multi-layer LIDAR scan for better key points association to achieve a consistent global map. A novel geometric feature detector is presented SKIP-3D and integrated with sensor odometry system F-LOAM for LIDARs.

Feature extraction through 3D LIDAR is a powerful technique that allows for the identification and characterization of important features in a given environment. By utilizing the high-resolution point cloud data provided by a 3D LIDAR system, it is possible to accurately detect and analyze various types of features, such as edges and planes. This information can then be used in a variety of applications, such as autonomous navigation, object recognition, and scene understanding. Overall, the use of 3D LIDAR for feature extraction has proven to be a reliable and effective method for gaining a detailed understanding of the surroundings. This demonstrates the general-purpose applicability of the method on benchmark indoor and outdoor

dataset.

To summarize, the use of geometric feature extraction from LIDAR data has the potential to significantly improve the accuracy and reliability of odometry and mapping in a wide range of applications, including self-driving cars, drones, and robotic assistants. By extracting and using these features, it is possible to build more detailed and accurate maps of the environment and to enable more sophisticated and intelligent navigation through the environment. This thesis covered the following points:

- Research aimed to improve understanding and exploitation of 3D LIDAR point clouds for efficient motion estimation and mapping.
- Focused on reducing data and verifying effectiveness of salient points in improving environment understanding.
- Maximizing the use of the LIDAR point clouds organizational structure and indices to efficiently locate neighboring points and perform place recognition using pre-existing signatures.
- Proposed feature extraction algorithm, SKIP-3D, demonstrated improved performance in odometry and mapping tasks with F-LOAM.
- The proposed geometric features effectively estimate points belonging to sharp items and planar patches in the scene.

4.1 Future Work

Feature extraction from LIDAR data has been a popular research topic in the field of robotics for odometry, mapping, and place recognition. In this section, I will discuss some of the current challenges and potential future directions in this area. One current challenge in feature extraction from LIDAR data is to design robust and discriminative features that can be extracted from the noisy and sparse point clouds. Traditional feature extraction methods, such as SIFT or SURF, are designed for image data and may not be directly applicable to LIDAR data. Some researchers have proposed to use

geometric features, such as normals and curvatures, as the basis for feature extraction, which I addressed in this thesis in the form of SKIP-3D algorithm. while others have proposed the use of non-geometric features, such as Deep Learning-based features, which can be trained on a large dataset of LIDAR scans.

Another challenge is to efficiently extract and match features across different LIDAR scans. This is important for odometry, as it requires matching features between consecutive scans to estimate the robot's motion. It is also important for mapping and place recognition, as it requires matching features between different scans taken at different times or locations to build a consistent map of the environment or to recognize previously visited places. Efficient feature extraction and matching methods are essential to enable real-time performance and to reduce the computational cost of these tasks.

One potential future direction in feature extraction from LIDAR data is to incorporate additional information, such as the robot's motion or the structure of the environment, to design more robust and discriminative features. For example, the robot's motion can be used to filter out moving objects or to align the scans to a common reference frame, which can improve the accuracy and robustness of the features. Similarly, the structure of the environment can be used to guide the selection and extraction of features, for example, by focusing on features that are more informative or more discriminative for the task at hand.

Another potential future direction is to explore the use of more advanced machine learning techniques, such as deep learning, to extract and match features from LIDAR data. These techniques have shown great promise in a variety of applications, and they may also be useful for feature extraction from LIDAR data. However, there are also challenges to be addressed, such as the need for large and diverse datasets to train the models and the need for efficient and effective ways to incorporate the learned features into the odometry, mapping, and place recognition algorithms.

In conclusion, feature extraction from LIDAR data is a field of active research with ongoing challenges and opportunities. Future work may involve the development of more robust features, the use of additional information and advanced machine learning techniques, and efficient extraction and matching of features across

LIDAR scans. These efforts will help to enable more accurate and robust odometry, mapping, and place recognition using LIDAR data.

Bibliography

[1] Chaiyapon Thongchaisuratkrul, Surachai Suksakulchai, D.M. Wilkes, and Nabin Sarkar. Sonar behavior-based fuzzy control for a mobile robot. volume 5, pages 3532 – 3537 vol.5, 02 2000.

[2] Cesar Cadena, Luca Carlone, Henry Carrillo, Yasir Latif, Davide Scaramuzza, José Neira, Ian Reid, and John J. Leonard. Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age. *IEEE Transactions on Robotics*, 32(6):1309–1332, 2016.

[3] Simon J. Julier and Jeffrey K. Uhlmann. New extension of the kalman filter to nonlinear systems. In *Defense, Security, and Sensing*, 1997.

[4] Yonghui Liu, Weimin Zhang, Fangxing Li, Zhengqing Zuo, and Qiang Huang. Real-time lidar odometry and mapping with loop closure. *Sensors*, 22(12), 2022.

[5] Stéphane Bazeille and David Filliat. Combining odometry and visual loop-closure detection for consistent topo-metrical mapping. *RAIRO - Operations Research*, 44:365–377, 10 2010.

[6] Randall C. Smith, Matthew Self, and Peter C. Cheeseman. A stochastic map for uncertain spatial relationships. 1988.

[7] Randall Smith, Matthew Self, and Peter C. Cheeseman. Estimating uncertain spatial relationships in robotics. In Ingemar J. Cox and Gordon T. Wilfong, editors, *Autonomous Robot Vehicles*, pages 167–193. Springer, 1990.

[8] Philippe Moutarlier and Raja Chatila. An experimental system for incremental environment modelling by an autonomous mobile robot. In Vincent Hayward and Oussama Khatib, editors, *Experimental Robotics I*, pages 327–346, Berlin, Heidelberg, 1990. Springer Berlin Heidelberg.

[9] J.J. Leonard and H.F. Durrant-Whyte. Mobile robot localization by tracking geometric beacons. *IEEE Transactions on Robotics and Automation*, 7(3):376–382, 1991.

[10] Joseph Djugash and Sanjiv Singh. A robust method of localization and mapping using only range. volume 54, pages 341–351, 01 2008.

[11] Gerhard Lakemeyer and Bernhard Nebel, editors. *Exploring Artificial Intelligence in the New Millennium*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2003.

[12] Feng Lu and Evangelos E. Milios. Globally consistent range scan alignment for environment mapping. *Autonomous Robots*, 4:333–349, 1997.

[13] Jens-Steffen Gutmann and Kurt Konolige. Incremental mapping of large cyclic environments. *Proceedings 1999 IEEE International Symposium on Computational Intelligence in Robotics and Automation. CIRA'99 (Cat. No.99EX375)*, pages 318–325, 1999.

[14] T. Duckett, S. Marsland, and J. Shapiro. Fast, On-line Learning of Globally Consistent Maps. *Journal of Autonomous Robots*, 12(3):287 – 300, 2002.

[15] Udo Frese. A proof for the approximate sparsity of slam information matrices. *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, pages 329–335, 2005.

[16] Dirk Ahnel, Sebastian Thrun, Ben Wegbreit, and Wolfram Burgard. Towards lazy data association in slam. 03 2004.

[17] Christian Forster, Zichao Zhang, Michael Gassner, Manuel Werlberger, and Davide Scaramuzza. Svo: Semidirect visual odometry for monocular and multicamera systems. *IEEE Transactions on Robotics*, 33(2):249–265, 2017.

[18] Christian Forster, Luca Carlone, Frank Dellaert, and Davide Scaramuzza. On-manifold preintegration for real-time visual–inertial odometry. *IEEE: Transactions on Robotics*, 33(1):1–21, feb 2017.

[19] Tong Qin, Peiliang Li, and Shaojie Shen. Vins-mono: A robust and versatile monocular visual-inertial state estimator. *IEEE Transactions on Robotics*, 34(4):1004–1020, 2018.

[20] Ryan Wolcott and Ryan Eustice. Robust lidar localization using multiresolution gaussian mixture maps for autonomous driving. *The International Journal of Robotics Research*, 36:027836491769656, 04 2017.

[21] Xiangyu Yue, Bichen Wu, Sanjit Seshia, Kurt Keutzer, and Alberto Vincentelli. A lidar point cloud generator: from a virtual world to autonomous driving. pages 458–464, 06 2018.

[22] Weixin Lu, Yao Zhou, Guowei Wan, Shenhua Hou, and Shiyu Song. L 3 -net: Towards learning based lidar localization for autonomous driving. 09 2019.

[23] Rendong Wang, Youchun Xu, Miguel-Angel Sotelo, Yulin Ma, Thompson Sarkodie-Gyan, Z. Li, and Weihua Li. A robust registration method for autonomous driving pose estimation in urban dynamic environment using lidar. *Electronics*, 8:43, 01 2019.

[24] Matea Galić, Tomislav Petković, and Tomislav Pribanic. On tablet 3d structured light reconstruction and registration. pages 2462–2471, 10 2017.

[25] Bo Fang and Chaoli Wang. A 3d reconstruction method based on the combination of the icp and artificial potential field algorithm. pages 483–493, 10 2018.

[26] Peng Wu, Wei Li, and Ming Yan. 3d scene reconstruction based on improved icp algorithm. *Microprocessors and Microsystems*, 75:103064, 02 2020.

[27] Shahram Izadi, David Kim, Otmar Hilliges, David Molyneaux, Richard Newcombe, Pushmeet Kohli, Jamie Shotton, Steve Hodges, Dustin Freeman, Andrew Davison, and Andrew Fitzgibbon. Kinectfusion: Real-time 3d reconstruction and interaction using a moving depth camera. pages 559–568, 10 2011.

[28] Ellon Mendes, Pierrick Koch, and Simon Lacroix. Icp-based pose-graph slam. pages 195–200, 10 2016.

[29] Hyunhak Cho, Eun Kim, and Sungshin Kim. Indoor slam application using geometric and icp matching methods based on line features. *Robotics and Autonomous Systems*, 100, 12 2017.

[30] Pileun Kim, Jingdao Chen, and Yong Cho. Slam-driven robotic mapping and registration of 3d point clouds. *Automation in Construction*, 89:38–48, 05 2018.

[31] Lei Han, Lan Xu, Dmytro Bobkov, Eckehard Steinbach, and Lu Fang. Real-time global registration for globally consistent rgb-d slam. *IEEE Transactions on Robotics*, PP:1–11, 01 2019.

[32] Dejing Ni, Aiguo Song, Xiaonong Xu, Huijun Li, Zhu Chengcheng, and Hong Zeng. 3d-point-cloud registration and real-world dynamic modelling-based virtual environment building method for teleoperation. *Robotica*, 35:1–17, 09 2016.

[33] P.J. Besl and H.D. Mckay. A method for registration of 3-d shapes. *IEEE Trans. Pat. Anal. Mach. Intel*, 14(2):239–256, 1992.

[34] S. Gold, A. Rangarajan, C.-P. Lu, S. Pappu, and E. Mjolsness. New algorithms for 2D and 3D point matching: Pose estimation and correspondence. *Pattern recognition*, 31(8):1019–1031, 1998.

[35] Y. Liu. Improving icp with easy implementation for free-form surface matching. *Pattern Recognition*, 37(2):211–226, 2004.

[36] S. Du, N. Zheng, S. Ying, and J. Liu. Affine iterative closest point algorithm for point set registration. *Pattern Recognition Letters*, 31(9):791–799, 2010.

[37] 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. *CoRR*, abs/2209.15397, 2022.

[38] R. Toldo, A. Beinat, and F. Crosilla. Global registration of multiple point clouds embedding the Generalized Procrustes Analysis into an ICP framework. In *5th Int. Symposium 3D Data Processing, Visualization and Transmission (3DPVT2010)*, 2010.

[39] J. Aleotti, D. Lodi Rizzini, R. Monica, and S. Caselli. Global Registration of Mid-Range 3D Observations and Short Range Next Best Views. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, pages 3668–3675, 2014.

[40] P. Biber and W. Straßer. The normal distributions transform: A new approach to laser scan matching. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, pages 2743–2748, 2003.

[41] M. Magnusson, A. Lilienthal, and T. Duckett. Scan registration for autonomous mining vehicles using 3d-ndt. *Journal of Field Robotics*, 24(10):803–827, 2007.

[42] D. Holz, A.E. Ichim, F. Tombari, R.B. Rusu, and S. Behnke. Registration with the point cloud library: A modular framework for aligning in 3-d. *IEEE Robotics Automation Magazine*, 22(4):110–124, Dec 2015.

[43] Y. Yang, S.H. Ong, and K.W.C. Foong. A robust global and local mixture distance based non-rigid point set registration. *Pattern Recognition*, 48(1):156–173, 2015.

[44] J. Fan, J Yang, D. Ai, L. Xia, Y. Zhao, X. Gao, and Y. Wang. Convex hull indexed Gaussian mixture model (CH-GMM) for 3D point set registration. *Pattern Recognition*, 59:126–141, 2016.

[45] M. Grogan and R. Dahyot. Shape registration with directional data. *Pattern Recognition*, 79:452–466, 2018.

[46] M.A. Fischler and R.C. Bolles. Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. *Commun. ACM*, 24(6):381–395, June 1981.

[47] P.H.S. Torr and A. Zisserman. MLESAC: A New Robust Estimator with Application to Estimating Image Geometry. *Computer Vision and Image Understanding*, 78(1):138–156, 2000.

[48] A. Myronenko and X. Song. Point set registration: Coherent point drift. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 32(12):2262–2275, Dec 2010.

[49] J. Ma, J. Zhao, J. Tian, A.L. Yuille, and Z. Tu. Robust Point Matching via Vector Field Consensus. *IEEE Transactions on Image Processing*, 23(4):1706–1721, April 2014.

[50] J. Ma, W. Qiu, J. Zhao, Y. Ma, A.L. Yuille, and Z. Tu. Robust L2E Estimation of Transformation for Non-Rigid Registration. *IEEE Transactions on Signal Processing*, 63(5):1115–1129, March 2015.

[51] J. Ma, J. Zhao, and A.L. Yuille. Non-rigid point set registration by preserving global and local structures. *IEEE Transactions on Image Processing*, 25(1):53–64, Jan 2016.

[52] M.K. Khan and I. Nystrom. A modified particle swarm optimization applied in image registration. In *2010 20th International Conference on Pattern Recognition*, pages 2302–2305, 2010.

[53] J. Sandhu, J. Dambreville, and A. Tannenbaum. Point Set Registration via Particle Filtering and Stochastic Dynamics. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 32(8):1459–1473, Aug 2010.

[54] J. Luck, C. Little, and W. Hoff. Registration of range data using a hybrid simulated annealing and iterative closest point algorithm. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, pages 3739–3744, 2000.

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

[56] R.B. Rusu, N. Blodow, and M. Beetz. Fast Point Feature Histograms (FPFH) for 3D registration. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, pages 3212–3217, 2009.

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

[58] S. Belongie, J. Malik, and J. Puzicha. Shape matching and object recognition using shape contexts. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 24, 2002.

[59] C. Brechbuhler, G. Gerig, and O. Kubler. Parametrization of closed surfaces for 3-d shape description. *Computer Vision and Image Understanding*, 61(2):154–170, 1995.

[60] A. Makadia, A. Patterson, and K. Daniilidis. Fully automatic registration of 3d point clouds. In *Proc. of IEEE Computer Vision and Pattern Recognition (CVPR)*, volume 1, pages 1297–1304, 2006.

[61] A. Makadia and K. Daniilidis. Rotation recovery from spherical images without correspondences. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 28(7):1170–1175, 2006.

[62] M. Kazhdan. An approximate and efficient method for optimal rotation alignment of 3D models. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 29(7):1221–1229, 2007.

[63] S. Althloothi, M.H. Mahoor, and R.M. Voyles. A robust method for rotation estimation using spherical harmonics representation. *IEEE Trans. on Image Processing*, 22(6):2306–2316, June 2013.

[64] A. Censi, L. Iocchi, and G. Grisetti. Scan Matching in the Hough Domain. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2005.

[65] A. Censi and S. Carpin. HSM3D: feature-less global 6DOF scan-matching in the Hough/Radon domain. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, pages 3899–3906, 2009.

[66] L. Reyes, G. Medioni, and E. Bayro. Registration of 3D Points Using Geometric Algebra and Tensor Voting. *International Journal of Computer Vision*, 75(3):351–369, Dec 2007.

[67] D. Lodi Rizzini. Angular Radon Spectrum for Rotation Estimation. *Pattern Recognition*, 84:182–196, dec 2018.

[68] Dario Lodi Rizzini and Ernesto Fontana. Rotation Estimation Based on Anisotropic Angular Radon Spectrum. *IEEE Robotics and Automation Letters*, 7(3):7279–7286, 2022.

[69] T.M. Breuel. Implementation Techniques for Geometric Branch-and-bound Matching Methods. *Comput. Vis. Image Underst.*, 90(3):258–294, June 2003.

[70] C. Olsson, O. Enqvist, and F. Kahl. A polynomial-time bound for matching and registration with outliers. In *IEEE Conf. on Computer Vision and Pattern Recognition*, pages 1–8, 2008.

[71] C. Olsson, F. Kahl, and M. Oskarsson. Branch-and-Bound Methods for Euclidean Registration Problems. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 31(5):783–794, May 2009.

[72] J. Yang, H. Li, D. Campbell, and Y. Jia. Go-ICP: A Globally Optimal Solution to 3D ICP Point-Set Registration. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 38(11):2241–2254, Nov 2016.

[73] H. Yang, J. Shi, and L. Carlone. Teaser: Fast and certifiable point cloud registration. *IEEE Transactions on Robotics*, 37(2):314–333, 2020.

[74] J. Zhang and S. Singh. LOAM: Lidar Odometry and Mapping in Real-time. In *Proc. of Robotics: Science and Systems (RSS)*, pages 834–849, Jul 2014.

[75] J. Zhang and S. Singh. Low-drift and Real-time Lidar Odometry and Mapping. *Autonomous Robots*, 41(2):401–416, Feb. 2017.

[76] T. Shan and B. Englot. LeGO-LOAM: Lightweight and Ground-Optimized Lidar Odometry and Mapping on Variable Terrain. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, pages 4758–4765, 2018.

[77] H. Wang, C. Wang, C. Chen, and L. Xie. F-loam : Fast lidar odometry and mapping. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, number 3, 2021. arXiv:2107.00822.

[78] H. Ye, Y. Chen, and M. Liu. Tightly Coupled 3D Lidar Inertial Odometry and Mapping. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, pages 3144–3150, 2019.

[79] 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 Int. Conf. on Intelligent Robots and Systems (IROS), pages 5135–5142, 2020.

[80] C. L. Gentil, T. Vidal-Calleja, and S. Huang. In2lama: Inertial lidar localisation and mapping. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, pages 6388–.6394, May 2019.

[81] Raúl Mur-Artal, J. M. M. Montiel, and Juan D. Tardós. Orb-slam: A versatile and accurate monocular slam system. *IEEE Transactions on Robotics*, 31(5):1147–1163, 2015.

[82] Jianke Zhu. Image gradient-based joint direct visual odometry for stereo camera. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17*, pages 4558–4564, 2017.

[83] Pedro Núñez Trujillo, R. Vázquez-Martín, José Carlos del Toro Lasanta, Antonio Bandera, and Francisco Sandoval Hernández. Feature extraction from laser scan data based on curvature estimation for mobile robotics. *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.*, pages 1167–1172, 2006.

[84] A. Diosi, L. Kleeman, and Albert Diosi. Uncertainty of line segments extracted from static sick pls laser scans. 01 2003.

[85] V.T. Nguyen, A. Martinelli, Nicola Tomatis, and Roland Siegwart. A comparison of line extraction algorithms using 2d laser rangefinder for indoor mobile robotics. pages 1929 – 1934, 09 2005.

[86] Z. Man, W. Ye, H. Xiao, and X. Qian. Method for corner feature extraction from laser scan data. *Nanjing Hangkong Hangtian Daxue Xuebao/Journal of Nanjing University of Aeronautics and Astronautics*, 44:379–383, 06 2012.

[87] Y. Li and M.Q.-H Meng. A general-purpose method to extract features from lidar data. 32:812–821, 11 2010.

[88] Kevin Peterson, Jason Ziglar, and Paul Rybski. Fast feature detection and stochastic parameter estimation of road shape using multiple lidar. pages 612 – 619, 10 2008.

[89] Michael Bosse and Robert Zlot. Continuous 3d scan-matching with a spinning 2d laser. pages 4312 – 4319, 06 2009.

[90] Z.J. Chong, Baixue Qin, Tirthankar Bandyopadhyay, Marcelo Jr, Emilio Frazzoli, and Daniela Rus. Synthetic 2d lidar for precise vehicle localization in 3d urban environment. pages 1554–1559, 05 2013.

[91] Yulan Guo, Hanyun Wang, Qingyong Hu, Hao Liu, Li Liu, and Mohammed Bennamoun. Deep learning for 3d point clouds: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PP:1–1, 06 2020.

[92] J. Serafin, E. Olson, and G. Grisetti. Fast and robust 3D feature extraction from sparse point clouds. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, pages 4105–4112, 2016.

[93] M. Muja and D.G. Lowe. Scalable Nearest Neighbor Algorithms for High Dimensional Data. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 36(11):2227–2240, 2014.

[94] Jingqi Jiang, Jing Yuan, Xuetao Zhang, and Xuebo Zhang. Dvio: An optimization-based tightly coupled direct visual-inertial odometry. *IEEE Transactions on Industrial Electronics*, 68(11):11212–11222, 2021.

[95] Qinxuan Sun, Jing Yuan, Xuebo Zhang, and Feng Duan. Plane-edge-slam: Seamless fusion of planes and edges for slam in indoor environments. *IEEE Transactions on Automation Science and Engineering*, 18(4):2061–2075, 2021.

[96] Rafael Muñoz-Salinas, Manuel J. Marín-Jimenez, and R. Medina-Carnicer. Spm-slam: Simultaneous localization and mapping with squared planar markers. *Pattern Recognition*, 86:156–171, 2019.

[97] Xiwu Zhang, Lei Wang, and Yan Su. Visual place recognition: A survey from deep learning perspective. *Pattern Recognition*, 113:107760, 2021.

[98] Stephanie Lowry, Niko Sünderhauf, Paul Newman, John J. Leonard, David Cox, Peter Corke, and Michael J. Milford. Visual place recognition: A survey. *IEEE Transactions on Robotics*, 32(1):1–19, 2016.

[99] Jing Yuan, Wenbin Zhu, Xingliang Dong, Fengchi Sun, Xuebo Zhang, Qinxuan Sun, and Yalou Huang. A novel approach to image-sequence-based mobile robot place recognition. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 51(9):5377–5391, 2021.

[100] Nathan Piasco, Désiré Sidibé, Cédric Demonceaux, and Valérie Gouet-Brunet. A survey on visual-based localization: On the benefit of heterogeneous data. *Pattern Recognition*, 74:90–109, 2018.

[101] Bastian Steder, Michael Ruhnke, Slawomir Grzonka, and Wolfram Burgard. Place recognition in 3d scans using a combination of bag of words and point feature based relative pose estimation. In *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1249–1255, 2011.

[102] Mikaela Angelina Uy and Gim Hee Lee. Pointnetvlad: Deep point cloud based retrieval for large-scale place recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[103] Zhe Liu, Chuanzhe Suo, Shunbo Zhou, Fan Xu, Huanshu Wei, Wen Chen, Hesheng Wang, Xinwu Liang, and Yun-Hui Liu. Seqlpd: Sequence matching enhanced loop-closure detection based on large-scale point cloud description for self-driving vehicles. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1218–1223, 2019.

[104] Lukas Schaupp, Mathias Bürki, Renaud Dubé, Roland Siegwart, and Cesar Cadena. Oreos: Oriented recognition of 3d point clouds in outdoor scenarios. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3255–3261, 2019.

[105] Giseop Kim and Ayoung Kim. Scan context: Egocentric spatial descriptor for place recognition within 3d point cloud map. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4802–4809, 2018.

[106] S. Belongie, J. Malik, and J. Puzicha. Shape matching and object recognition using shape contexts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(4):509–522, 2002.

[107] A.U. Khan and D. Lodi Rizzini. Novel SKIP Features for LIDAR Odometry and Mappings. In *Proc. of the Int. Conf. on Intelligent Computer Communication and Processing (ICCP)*, pages 1–6, Oct. 2021.

[108] L.J. Latecki and R. Lakamper. Shape similarity measure based on correspondence of visual parts. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 22(10):1185–1190, Oct 2000.

[109] A. Geiger, P. Lenz, and R. Urtasun. Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[110] R. Voges and B. Wagner. Timestamp offset calibration for an imu-camera system under interval uncertainty. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, pages 377–384, 2018.

[111] A. Aybakan, G. Haddeler, M. Caner Akay, O. Ervan, and H. Temeltas. A 3d lidar dataset of itu heterogeneous robot team. In *Proc. of Int. Conf. on Robotics and Artificial Intelligence (ICRAI)*, pages 12–17, Nov 2019.