8

LiDAR SLAM

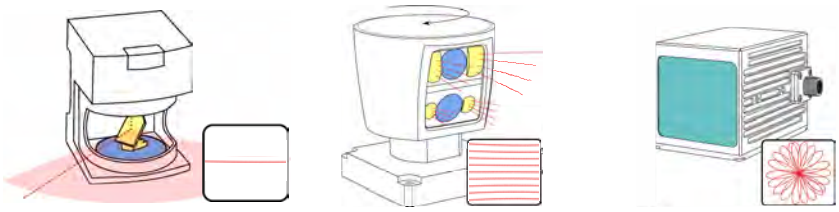
Jens Behley, Maurice Fallon, Shibo Zhao, Giseop Kim
Ji Zhang, Fu Zhang, and Ayoung Kim

Along with cameras, LiDAR sensors are one of the major sensing modalities used in robotics and computer vision. LiDAR is a technology which uses a laser to actively transmit laser light pulses and then measures the time delay in those pulses reflecting off of surfaces and returning to a detector. In doing so it directly measures the distance to those surfaces. A LiDAR sensor can be used to perceive the structure of its surrounding environment and also to estimate the motion or ego-location of the sensor.

The development of LiDAR technology began in the 1960s and 1970s with stationary systems primarily used for applications such as atmospheric research, topographic mapping, and military applications. These early LiDAR systems were bulky and expensive, making them unsuitable for mobile applications. In the 1980s, advancements in laser technology and computing power allowed for more compact and affordable LiDAR systems. These systems were still primarily stationary and used in applications like terrain mapping and environmental monitoring [257, 690]. In the 1990s, the integration of LiDAR sensors with Global Positioning System (GPS) and IMUs began to enable the first mobile LiDAR mapping systems. These systems were often mounted on vehicles or aircraft to create detailed 3D maps of large areas [507]. In the late 1990s, researchers began exploring the use of LiDAR for real-time SLAM in robotics. In the next section we will quickly review the underlying technology within different types of LiDAR sensors.

8.1 LiDAR Sensing Preliminary and Categorization

By rotating the laser emitter and detector within a LiDAR sensor in one or two axes, it is possible to build up a detailed point cloud of the environment around the sensor. The most basic principle is TOF, which employs laser pulses to infer distances using the measured time it takes for emitted light pulses to return to the detector. TOF LiDAR sensors can capture high-resolution measurements but are sensitive to external light, which reduces the signal-to-noise ratio (SNR) [606] and therefore the accuracy and frequency of measurements. Besides TOF, the techniques of Amplitude Modulated Continuous Wave (AMCW) and Frequency



(a) 2D LiDAR (b) 3D Rotating LiDAR (c) Spiral Scanning

Figure 8.1 Common LiDAR sensor types and their beam patterns. (a) A standard 2D LiDAR sensor with a rotating transmitter mirror (yellow). The encoder disk (blue) is used to measure the rotation angle of the mirror. (b) An example of a mechanical multi-beam LiDAR emitting multiple laser beams from the different emitters (yellow), which are then detected using the detectors (blue). The entire sensor head rotates to generate a 360° horizontal field of view. (c) A macroscopic steered Risley prism LiDAR with an spiral scanning pattern which only measures in the direction of the lens window (cyan). Due to the spiral pattern it can produce denser point clouds over time.

Modulated Continuous Wave (FMCW), originally developed for radar sensors have been adopted for LiDARs.

The classes of LiDAR sensors are categorized by their sensing mechanisms [941] and can be classified into mechanical LiDARs, scanning solid-state LiDARs, flash LiDARs and sensors using macroscopic scanning Risley prisms. Among these, we will examine two commonly utilized types: 2D/3D mechanical LiDARs and macroscopic scanning LiDARs.

Mechanical LiDARs are the most common category of LiDAR. They use a rotating assembly to direct the laser beam but face limitations due to mechanical wear and lower rates of data acquisition. The simplest 2D mechanical LiDAR sensors use a rotating mirror to direct a single laser beam and to measure distances, as shown in Figure 8.1a. By using an encoder disk, the angular orientation of the LiDAR beam can be measured and paired with the range measurement to produce a 2D profile measurement. By its nature, this LiDAR can only scan in an individual 2D plane.

The demand for a single sensor that can scan in full 3D was motivated by the DARPA Grand Challenges (an early self-driving car competition) in the 2000's and lead to the development of the pioneering Velodyne HDL-64E sensor. It was used by most of the teams [1119, 778, 544] in the challenge. Within a 3D mechanical LiDAR, multiple laser emitters are mounted on a single rotating mechanism to capture individual range measurements pointing in different elevation angles as the entire mechanism rotates through 360 degrees in the azimutal axis as illustrated in Figure 8.1(b). The resulting point cloud can capture a highly detailed 3D depiction of objects and the sensor's surroundings as shown in Figure 8.2, as discussed in

Figure 8.2 Example of a single multi-beam LiDAR scan and a corresponding image. The scan is from a 64-beam Hesai QT64 scanner with 104° vertical field-of-view. Note the individual scanning lines and also how dense the point cloud is close to the device (the colored axis in the 3D view) and much sparser the cloud is further away.

various studies [1175, 565, 941]. Subsequently, the technology has evolved — with the size and price of LiDAR sensors dropping significantly.

A wider variety of prototype sensors have emerged more recently drawing on a variety of physical properties including solid-state LiDAR , Risley prisms and polygonal mirrors. In contrast with traditional scanning laser, many of these sensors use Micro-electromechanical (MEMS) [475] mirror technology or optical phased arrays (OPA) [454] to avoid, or at least minimise, mechanical rotation. This is important because it can enhance the LiDAR's lifespan and reliability in environmental mapping, since fewer mechanical parts need to be actively actuated which reduce the mechanical wear.

A notable advancement is the use of Risley prisms [674] which enables rapid, controlled beam steering with much smaller amount of physical movement. This innovation results in a more compact sensor, albeit with more limited FOV currently.

All the aforementioned sensors produce sets of individual range measurements with intensity (often also called remission) value for each measurement, *i.e.,* how much of the LiDAR beam is reflected. The range measurements with associated angles of the individual beams can be then converted into a 2D or 3D point cloud. Yet, more recent advances have introduced additional measurement capabilities beyond the default range measurement. For instance, FMCW LiDAR continuously projects light with varying frequency and can measure the relative velocity of the object the beam hits using detected frequency shifts. This is similar to how FMCW is used in radar. This approach is useful in dynamic environments or challenging scenarios [1198], although it tends to be more complex and costly compared to other variants. Another innovative sensing mechanism is flash LiDAR, which can provide ambient channels resembling photometric measurements, similar to those obtained by cameras. These technologies, with their different characteristics of power consumption, weight and cost, provide a variety of options when carrying out LiDAR odometry for different applications.

8.2 LiDAR Odometry

The first building block of LiDAR SLAM is LiDAR odometry. The goal of LiDAR odometry is to estimate the incremental ego-motion of a robot or vehicle in real-time given a LiDAR scan and past observations, *i.e.,* a single scan or multiple scans aggregated into a local map. Here, the term *scan* refers to a single sweep or cycle of data collected by the LiDAR sensor. More specifically, a scan typically represents one complete rotation or one full sweep of the sensor providing a contextual snapshot of the surrounding environment at a specific time. Thus, scans are often time-stamped, allowing them to be ordered and processed as sequential observations.

At the heart of LiDAR odometry lies the technique of scan registration, also referred to as scan matching. Scan registration involves finely aligning a pair of scans to estimate the precise relative transformation between the scans. A scan is effectively a set of points or a point cloud. The well-known ICP algorithm [86, 964] is a fundamental technique for point cloud registration and it can be used to determine this relative transformation. We will discuss ICP further in the next sections.

The original development of LiDAR SLAM can be traced back to the seminal work of Lu and Milios [709], who pioneered the concept of globally consistent 2D range scan registration by introducing the idea of a network of poses—a concept closely resembling the modern pose-graph approach. This work also laid the groundwork for LiDAR odometry by defining the fundamentals of 2D scan registration. Key contributions to the probabilistic framing of scan-to-scan matching were made by works including [832, 604]. While points and lines are common choices in 2D scan matching, Olson [831] introduced an impactful approach using correlation techniques for real-time registration.

Early extensions to 3D LiDAR built on these 2D scanning techniques by actively moving the sensor in a nodding [440] or rotating manner [106], or by passively using the motion of a human carrier [108] or vehicle [106] to accumulate denser 3D point clouds. These 3D point clouds enabled 3D scan matching for LiDAR SLAM but introduced significant computational challenges due to the increased data size. Addressing these challenges, LOAM [1264] demonstrated real-time scan matching capabilities forming the basis for follow-up methods in LiDAR odometry and SLAM.

8.2.1 Foundations of Scan Registration

Scan registration is a fundamental component of LiDAR odometry and mapping systems. It involves the alignment of two scans to achieve an accurate alignment and mapping. The goal is to find the transformation, *i.e.,* rotation $\boldsymbol{R} \in \Re^{3\times3}$ and translation $\boldsymbol{t} \in \Re^3$, that can best bring one of the scans (potentially a recent scan from a sensor) into alignment with the other (*e.g.,* a scan or a local map). In doing

so, this process also yields the relative position from which the scans were taken. A large body of techniques and algorithms have been developed to perform scan registration with high accuracy, robustness and low computational cost.

Iterative Closest Point and Its Variants As introduced in Chapter 5, a point cloud is defined to be a set of points in a three-dimensional coordinate system, represented mathematically as $\mathsf{P} = \{\boldsymbol{p}_i \in \Re^3 \mid i = 1, 2, \ldots, N\}$, where each $\boldsymbol{p}_i = (x_i, y_i, z_i)$ denotes the 3D coordinates of a point. For scan registration, the ICP algorithm [86] minimizes the total registration error between two point clouds P and Q. Let us denote P as a source and Q as a target point cloud.

ICP iteratively determines transformations $\boldsymbol{R}^k, \boldsymbol{t}^k$ for an optimization iteration k that minimize the total registration error, which is measured by different distance metrics d and is given by

$$\boldsymbol{R}^k, \boldsymbol{t}^k = \arg\min_{\boldsymbol{R},\boldsymbol{t}} \sum_{(\boldsymbol{p},\boldsymbol{q})\in\mathsf{C}} d(\boldsymbol{p}_i, \boldsymbol{R}\boldsymbol{q}_i + \boldsymbol{t}), \tag{8.1}$$

where the set of correspondences C between the source point cloud P and target point cloud Q is given by

$$\mathsf{C} = \{(\boldsymbol{p}, \boldsymbol{q}) \mid \boldsymbol{p} \in \mathsf{P}, \boldsymbol{q} \in \mathsf{Q}\}. \tag{8.2}$$

In the ICP algorithm, determining the transformation between the source and target is achieved iteratively by recomputing for each iteration k a new set of correspondences C based on the last transformation at iteration $k-1$ given by rotation $\boldsymbol{R}^{k-1}$ and translation $\boldsymbol{t}^{k-1}$.

To minimize the total registration error in (8.1), two components must be specified:

1 While geometric relation is used defining the distance measure $d(\cdot)$? The aim is to align two point clouds as tightly as possible, and this *tightness* cannot be determined without defining a distance metric.
2 How is the correspondence set C, used for minimization, determined? This involves identifying the corresponding target point $q \in \mathsf{Q}$ for each source point $p \in \mathsf{P}$.

Common approaches used these two components will be discussed in the following sections.

8.2.1.1 Distance Measure in Registration Residual

The first component involves deciding which geometric elements to use for defining the residual. Typically, points, lines, and planes are the most commonly used geometric elements, as summarized in Figure 8.3.

Point-to-point ICP is the most basic approach and it minimizes the Euclidean distance between corresponding points in the two point clouds. Pioneering works

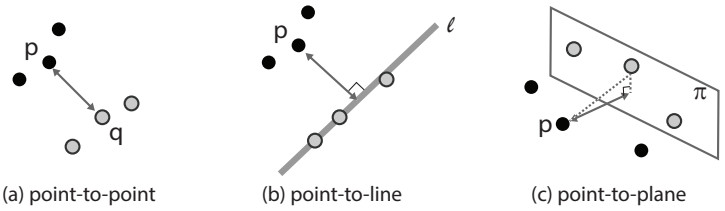


Figure 8.3 Typical distance metrics used in ICP. (a) Point-to-point distance is as straightforward as the Euclidean distance between two points. (b) and (c) The point-to-higher-level feature (*e.g.,* line or plane) is calculated as the shortest distance to the reconstructed line or plane using the target points.

in ICP by Zhang [1276] and Besl and McKay [86] formulated shape (*e.g.,* curves and surfaces) matching as a point matching problem by representing shapes as sets of points. This point-to-point cost is straightforward and simple, but can be sensitive to noise and outliers. Distances to lines are also used as an error measure. The point-to-line distance measures points in one point cloud and lines (formed by connecting points) in the other point cloud. It can provide better results in structured environments with linear features. By exploiting higher level geometric features we can go further. We can measure the distance between points in one point cloud and planes (local surfaces) in the other point cloud. This approach is more robust to noise and can achieve higher accuracy in environments with planar surfaces.

Extending from these basic geometries, other ICP variants introduce different distance metrics. Techniques which employ multi-distance metrics [845], continuous-time formulation [264], and adaptive thresholds [1137] are more some of the more recent ICP advances. Other methods [990, 599] opted to evaluate differences in a probability distribution of a local neighborhood than using Euclidean distances.

Another well-known distribution-based matching is the NDT [89]. NDT divides an input point cloud into a set of voxels and fits a normal distribution to the points in each voxel (see Chapter 5.3.2.2 for more details). Instead of incurring the cost of determining nearest neighbor associations, it takes advantage of voxelization to carry out a distribution-to-distribution matching process. This process can take advantage of a smoother and more robust registration cost surface, especially in complex environments.

8.2.1.2 Determining Correspondences

The second core component of common ICP algorithms is data association or correspondence search between the source P and the target Q.

In the most basic form, determining correspondences between P and Q can be achieved geometrically by finding the nearest neighbor of a point $\boldsymbol{p} \in \mathsf{P}$ in the

target Q, where we use the iteratively updated transformation $(\boldsymbol{R}^{k-1}, \boldsymbol{t}^{k-1})$, which is given by:

$$\mathsf{C} = \{(\boldsymbol{p}, \boldsymbol{q}) \mid \boldsymbol{p} \in \mathsf{P}, \boldsymbol{q} = \underset{\boldsymbol{q}' \in \mathsf{Q}}{\arg\min} ||\boldsymbol{p} - (\boldsymbol{R}^{k-1}\boldsymbol{q}' + \boldsymbol{t}^{k-1})||_2\} \tag{8.3}$$

However, this is typically an expensive operation when computed over a large point cloud with thousands of points.

To make real-time operation of LiDAR odometry possible, there are two common strategies used to reduce the time for correspondence search: (1) reducing the set of potential candidates for a correspondence or (2) employing a different search strategy than distance-based neighbor search to more quickly find potential candidates.

Several ICP variants used in popular LiDAR odometry systems [1264, 845, 1005] employ the first strategy to reduce the set of potential correspondences by building maps with reduced candidate sets, $\mathsf{P}' \subset \mathsf{Q}$ and $\mathsf{Q}' \subset \mathsf{Q}$ by determining points that fulfill certain geometric criteria. The criteria used include determining points lying on edges or surfaces [1264, 845], removing less descriptive points that correspond to the ground plane [1004], or downsampling of the target scan [1137, 264]. While these strategies certainly speed up the correspondence search, they have the potential drawback of removing true correspondences from the target Q.

In contrast, the second strategy employs search structures with efficient approximations which enable faster correspondence searches even though they may not always yield the exact nearest neighbor. In this direction, a common strategy is to use projective neighbor search in range images [73] or leveraging voxel grids for approximate neighbor search [1264, 264, 1137]. Furthermore, while we covered here the pure geometric correspondence search in Euclidean space, it is also possible to use alternative distance metrics, as well as to apply projections into a feature space [845] to identify correspondences.

In the following sections, we will discuss other components of a LiDAR odometry system that are integrated around the core ICP component to build out a complete scan registration system which can align a sequence of scan observations so as to estimate the relative motion of a robot.

8.2.2 Common Components for LiDAR Odometry

This section outlines the key modules involved in a LiDAR odometry system: point cloud motion compensation, identifying correspondences and pose estimation via scan registration. Point cloud motion compensation addresses the distortion caused by the motion of the LiDAR during scan acquisition. Correspondence search identifies matching points between consecutive scans that are useful for matching and scan registration. Finally, the pose estimation module estimates the sensor's motion since the previous registration. Registration can be carried out either *scan-to-scan*

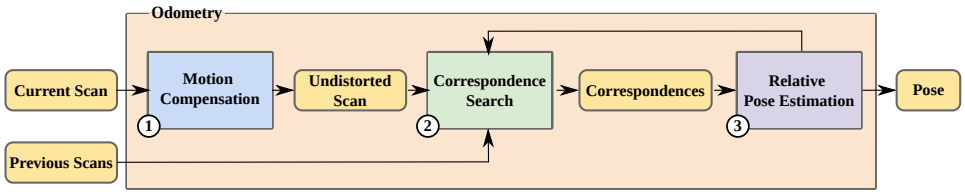


Figure 8.4 Components of a LiDAR odometry pipeline: Given a current scan, (1) motion compensation accounts for the motion of the sensor during the scan process resulting in a undistorted scan. Then, (2) correspondences between the undistorted scan and the previous scans, either a single scan or aggregated scans in a local map, are determined. Finally, (3) the relative pose estimate is determined via a scan registration to estimate the relative pose of the current scan. These steps are iterative with the correspondence set refined based on the intermediate relative pose estimates. After convergence, the final pose estimate is the output of the LiDAR odometry system.

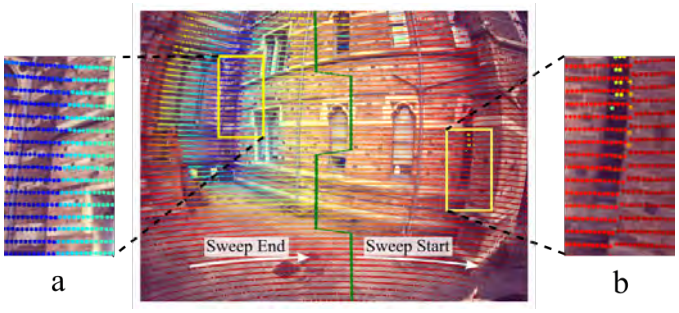
between consecutive scans or *scan-to-map* with respect to a local map. Together, these modules ensure accurate and reliable LiDAR odometry. Figure 8.4 shows the interplay between the different components in a common LiDAR odometry pipeline.

8.2.2.1 Point Cloud Motion Distortion Compensation

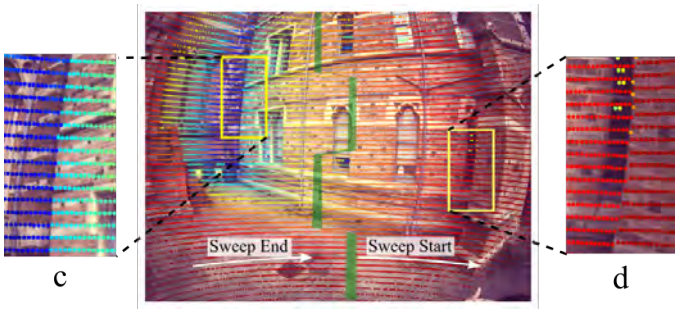
Motion distortion in LiDAR odometry occurs because a LiDAR sensor captures a scan over a period of continuous motion[1]. Due to this movement during the scan period, the sensor will emit laser pulses and receive ranging returns from slightly different times and positions. This means that a single scan does not represent a static snapshot of the surroundings at a single position but instead a set of points each captured from a slightly different scanning position. This continuous movement during the scanning process will lead to inaccuracies and in turn a distorted point cloud if left uncorrected [2]. For example, Figure 8.5 clearly illustrates a distorted point cloud sample and highlights the need for proper motion compensation. As can been seen, undistorting the point cloud to account for the motion of the LiDAR sensors is an important pre-processing step in LiDAR odometry which can improve accuracy and robustness. Compensation methods have been developed to correct this effect including a constant-velocity model, continuous-time trajectory optimization, and using IMU measurements.

Constant-velocity Model The constant velocity model assumes that the robot maintains the same translational and rotational velocities estimated during the previous time step. As this model does not require any additional sensors, it can

[1] In modern LiDAR SLAM, LiDAR sensors are often mounted on moving vehicles, robots or wearable devices. Assuming a scan rate of 10 Hz, the scan period will then be 0.1 seconds.
[2] A similar type of distortion effect during camera image capture and is known as the rolling shutter effect.



(a) LiDAR-camera overlay - without motion undistortion



(b) LiDAR-camera overlay - with motion undistortion

Figure 8.5 LiDAR point cloud overlaid on a camera image. A slight misalignment at the start of the sweep ('b') worsens significantly by the end of the LiDAR sweep ('a'). At the bottom, more consistent overlays are observed for both the start ('d') and end ('c') after motion compensation. From [1076].

be widely used in simpler LiDAR odometry systems [264, 1137], however, the constant velocity assumption is inherently less accurate when the motion contains high frequency motions.

Continuous-time Trajectory Optimization Another widely used approach for motion compensation is continuous-time trajectory optimization techniques using splines [290, 723] and the GP [57]. Continuous-time trajectories allow pose estimates to be made at any time instant without relying on linear interpolation. They can be used to remove the distortion of each individual point, however, conventional continuous-time trajectory optimization is time-consuming and often implemented offline [723].

IMU-based Motion Compensation IMU measurements directly measure high frequency motion with gyroscopes and accelerometers. They can be integrated over the scan period as an effective approach for motion compensation [1264, 1005]. IMU-based motion compensation pre-integrates the LiDAR pose using the most recent

IMU data and then uses that predicted trajectory to rectify for point distortion. Thanks to the high frequency of IMU measurements (*e.g.,* 200 Hz), IMU-based motion compensation is highly effective for jerky robot motions and is now the *defacto* standard for most robot platforms. Nonetheless, this method needs to be used carefully because IMU measurement noise, bias estimation errors and poor clock synchronization can cause this approach to underperform the other simpler methods.

Point-wise Registration Being firstly proposed in Point-LIO [449], this point-wise approach is fundamentally different from existing scan-based LiDAR odometry frameworks. In this framework, the state is updated by processing each LiDAR point when it is received, rather than accumulating a complete scan. As a result of this design, the proposed method does not suffer from intra-scan motion distortion.

8.2.2.2 Feature-based LiDAR Odometry

Once motion distortion is corrected, point correspondences can be established. Similar to visual SLAM (see Chapter 7), leveraging features for scan association is well studied. A feature-based approach allows efficient representation of the scan by processing only a small number of features extracted from the point cloud. Correspondence matching and residual computation can also be performed at the feature level, substantially reducing the overall computational cost.

Low-level Features Lines and planes are the most commonly used features in practice. In this line of study, the well-known LOAM algorithm [1264] was a significant breakthrough in LiDAR SLAM which uses a low-level detector to efficiently identify mid-level features that can contribute to scan registration. Points with high or low curvature are identified to detect edges and planes. The curvature of each point is calculated by analyzing the differences between the point and its neighbors. High curvature points are marked as edge features, while low curvature points are identified as planar features. Not all points are used as features; the algorithm selects a subset of the most significant edge and plane points to reduce computational complexity while maintaining accuracy.

Principal component analysis can also be employed to identify the principal directions of a local neighborhood of a point for effective feature detection [74, 760]. By calculating the eigenvalues and the corresponding eigenvectors of the covariance matrix of a point and its neighbors, the main axes of variation can be determined. This analysis allows us to discern the geometric properties of the points. Points with one dominant eigenvalue can then be classified as edge points, indicating sharp transitions or boundaries. In contrast, points with two similar eigenvalues can be identified as being from planar regions, representing flat surfaces within the point cloud. This enables differentiation between edge and planar features, enhancing the accuracy of LiDAR odometry by considering the geometric properties.

High-level Features In LiDAR odometry, high-level features such as semantic, surfel, and intensity features can also play an important role in enhancing the accuracy and robustness of the system. These features provide richer information about the environment compared to low-level features, facilitating better scene understanding and more accurate mapping.

- **Semantic Features** Semantic features involve the use of machine learning and deep learning techniques to classify and label the LiDAR point cloud into object categories such as vehicles, pedestrians, buildings and vegetation — typically to distinguish between dynamic and static objects. It can improve the reliability of odometry by focusing on stable landmarks [194].
- **Surfels Features** Surfels are small disk-like representations of the surface geometry of a point cloud (see Chapter 5.3.2.2). An ellipsoid disk can be fit to a set of points with the ellipsoid's principal semi-axes lengths determined by the eigenvalues of the covariance matrix of nearby points. This type of surface representation can then be used to compute point-to-plane distances during scan registration [852, 73, 910].
- **Intensity Features** Intensity features [285, 422] refer to the reflectivity or intensity values of the LiDAR returns. These values provide additional information about the material properties and surface characteristics of objects in the environment. They improve the robustness of feature matching by providing an additional dimension of information, which can be crucial in challenging scenarios such as structure-less environments.

8.2.2.3 Direct Point-wise LiDAR Odometry

A problem with this feature-based approach, is that it tends to discard subtle contributions from isolated points which do not clearly correspond to planes or edges. This is particularly a problem when mapping unstructured environment with bushes or branches in natural environments. It also requires tuning of hand-engineered feature detectors when moving from one sensor to another. Additionally, the number of points to be processed scales linearly with the LiDAR scan. The efficiency of this approach can become eroded when working with modern 64 or 128 beam sensors.

Alternatively, one can use the points directly — without extracting mid-level features. Similar to the direct methods in visual SLAM, we can directly align points in an ICP-like manner. However, due to the high computational cost during point-wise correspondence matching, this direct methods were not favored in the early development of LiDAR SLAM.

Paving the way for direct methods, Zhou et al. [1290] made it practical to use direct methods by speeding up the nearest neighbor search with a GPU-accelerated KD-tree implementation. Later the direct method Fast-LIO2 by Xu et al. [1207] demonstrated highly accurate frame-rate odometry without suffering from a correspondence search bottleneck when the map grows large using an extension, the

so-called incremental KD-tree (iKD-tree). The iKD-tree can adapt to the distribution of points by occasionally rebalancing itself to allow for efficient add, remove and query operations, which avoids rebuilding the tree for each added scan.

8.2.2.4 Local Mapping and Pose Estimation

Once reliable correspondences have been obtained, the next step is to achieve consistent registration of the consecutive scans. As mentioned before, incremental pose estimation in LiDAR odometry is achieved, in most cases, via a scan registration with a variant of ICP (see Section 8.2.1).

Initially methods focused on *scan-to-scan odometry* and sought to achieve full frequency (10 Hz) output while treating each consecutive scan registration operation as being independent. However, as individual LiDAR scans can be relatively sparse, many points will have unsuitable correspondences if the reference scan is simply the previous scan. This will result in registration errors accumulating and an inconsistent overall map.

A more modern approach is to build a detailed and accurate local map around the sensor which is known as *scan-to-map odometry*. This paradigm has been successfully used in 2D LiDAR SLAM [466], 3D LiDAR odometry [1207, 264, 1137], and 3D LiDAR SLAM systems [845, 73, 852] and has been seen to reduce overall drift rates.

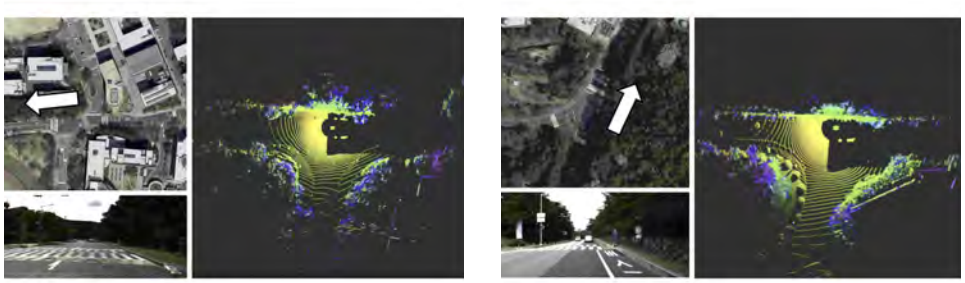
The motion prediction from either scan-to-scan odometry or IMU pre-integration can be used to pre-align the incoming scan before a fine registration to the persistent local map is carried out. The local map which becomes much dense than an individual scan results in much more suitable inlier associations. After registration, the incoming scan will be added to this local map.

Earlier LiDAR odometry approaches [1264] needed to resort to interleaving scan-to-scan odometry at a high frequency with scan-to-local-map odometry at lower frequency due to compute restrictions. More modern LiDAR odometry approaches [264, 1137] now use a single-stage scan-to-map alignment with direct point-wise correspondences enabled by a voxelized local map representation.

Finally, a quite different approach to LiDAR odometry is to use deep learning. Efforts to learn ego-motion directly from LiDAR measurements has resulted in some promising works. Early studies employed supervised learning using ground-truth labels [662], and this line of work has been extended to unsupervised learning methods [218]. While the performance of deep LiDAR methods are generally promising, concerns have been raised regarding their generalization capabilities.

8.2.3 Summary

To summarize, common 3D LiDAR odometry algorithms can produce highly accurate and robust motion estimates by iteratively aligning incoming scans to a running local map — often with the support of IMU measurements or motion models to



(a) Place A (b) Place B

Figure 8.6 Despite the large visual differences between Place A and Place B in the RGB images (aerial view and robot's front-looking view), their corresponding LiDAR scans exhibit structurally similar patterns. This causes structural perceptual aliasing, where distinct places appear similar to LiDAR-based place recognition algorithms due to shared road topology and surrounding structures. This example is captured from the `SNU Afternoon` sequence of the STheReO dataset [1257].

correct for motion distortion of the scan. Resulting systems can achieve drift rates in the order of 1 m per 1000 m traveled — but the performance is highly dependent on the environment around the robot and the level of dynamics present in the scene and the dynamics of the sensor itself. Accounting for this remaining amount of small drift is a key aspect of LiDAR SLAM with place recognition being a key component of such a system.

8.3 LiDAR Place Recognition

Place recognition systems seek to identify places that have been previously visited by a robot/sensor. It is an essential capability for several applications, including multi-session SLAM as well as global localization in a prior map. Unlike visual data, LiDAR data allows a robot to obtain consistent metric 3D information about the surrounding environment. This capability ensures that LiDAR is less affected by lighting condition changes than conventional cameras. However, despite this advantage over visual localization, the nature of LiDAR sensing presents unique challenges for LiDAR place recognition. For example:

Sparse Data In contrast to visual measurements, where pixels are dense (megapixel cameras) and organized (*i.e.,* structured into a regular 2D grid), the spacing and local density of points captured by conventional LiDAR sensors varies depending on the type of sensor (*e.g.,* the number of beams) and the sensing range (*e.g.,* points farther away are sparser). The resolution of point clouds is typically much lower than camera images. Because of these limitations, LiDAR place recognition typically does

not rely on local keypoint descriptors, where each point has its own feature descriptor. To address the lack of structure, approaches identify semantically meaningful point cloud segments [293, 1269] or compute global descriptions [576, 1208] (*i.e.,* a single representative descriptor for a scan). Recently, with the advancement of deep learning, learning to determine robust local keypoint descriptors has also been actively studied [162]. Seminal papers and paradigms in the area of LiDAR-based place recognition will be revisited in more detail in Section 8.3.2.

Structural aliasing The second difficulty that LiDAR-based place recognition systems face is structural repetition in man-made environments such as long corridors or indistinguishable structures on highways. Consider the corridors on each floor of a regular modern office building. Using a camera, visual place recognition might be able to identify unique or descriptive visual texture (e.g. pictures, posters or decorations) on otherwise identical corridor walls. However, it is very difficult to distinguish the specific floor using only a single scan obtained in the corridor. A similar challenge arises in outdoor environments, where perceptual aliasing can occur between visually distinct places that share similar structural layouts in LiDAR scans, as shown in Figure 8.6. Global LiDAR descriptors such as ScanContext [576] typically fail in such situations. Other approaches using object-level clusters, such as SegMap [293] and InstaLoc [1269], have been developed using higher level semantic features and can be more successful in such situations.

In summary, research needs to keep in mind these specific challenges when developing LiDAR place recognition methods.

8.3.1 Problem Definition

In this section, we will focus on the task of place recognition – the loop closure candidate detection problem. Given a query (*i.e.,* a scan represented as a point cloud), the objective is to retrieve corresponding entries from a database that are similar to the query. The database (*i.e.,* the previously visited places) is a set composed of disjoint place descriptors spatio-temporally acquired in an explored region.

A key consideration in LiDAR place recognition is the robustness of the retrieval method to variations in the sensor type, acquisition time, and robot pose. For example, the LiDAR type used in a query may differ from that used to create the database if different LiDAR devices were employed. Furthermore, a temporal gap between mapping (*i.e.,* building the database of visited places) and revisiting a place at a later point in time is inevitable. This temporal gap might lead to structural changes in the environment as well as differences due to dynamics caused by moving objects or people. However, the most significant variation arises due to changes in the robot's pose. Translation and/or rotation shifts between the database and the query result in different appearance of the captured sensor data of the same

environment. Consequently, LiDAR place recognition methods must be robust to pose variations and environmental changes at the same time.

If a method cannot determine pose variance but still can correctly identify a candidate, it is said to have *invariance*. If it can also estimate pose variance, it is described as having *awareness* of the revisit pose variations. Researchers have particularly focused on this awareness property for two main reasons. First, it serves as a good initial guess for fine registration, which is crucial when establishing the SE(2) or SE(3) constraints required for estimating precise loop closures (see Section 8.4.1). Secondly, working towards the more complex goal of awareness can naturally enhance the invariance capability (*e.g.,* estimating heading changes [571] or inferring the degree of overlap [195]).

8.3.2 Methods for LiDAR Place Recognition

In addition to achieving invariance and awareness, we should note that point cloud representations with different levels of granularity have been proposed to address the unstructured nature of the raw LiDAR measurements and to ensure real-time place retrieval performance for large-scale robot autonomy. Approaches for descriptor-based LiDAR place recognition can generally be categorized as using either local or global descriptors for retrieval and matching in the database. That said, there are variants that as well as using descriptor distance for place recognition also directly learn a place similarity function. In the following, we will discuss these different paradigms in more detail.

8.3.2.1 Local Descriptors

In the early days of LiDAR place recognition research, and corresponding to the evolution of visual place recognition methods (*e.g.,* SIFT [706], ORB [958], DBoW2 [370]), computing local keypoint descriptors was a natural approach both for 2D [1100] and 3D LiDAR sensors [107]. However, 3D local descriptors specifically developed for dense RGB-D point cloud registration or object recognition [965, 966, 1107] typically struggle to be adapted to the sparsity and unstructured nature of LiDAR sensing — particularly in outdoor scenarios. To mitigate this sensitivity, methods have been proposed that use the statistical distribution of local keypoints (*e.g.,* histograms) [470]. However, these descriptor remain limited to a local neighborhood, which results in poor descriptiveness due to the lack of metric structural context from across an overall scan.

8.3.2.2 Global Descriptors

In contrast to the local approaches, global descriptors leverage the higher level patterns in the entire scan rather than concentrating on low-level local keypoints. These methods aim to address the lack of structure by building a simpler and coarser representation. In turn, this often results in matching methods which are

more computationally efficient. Two coarse representations which have been widely used to generate global descriptors are as follows:

Bird's-eye-view (BEV) BEV representations transform a 3D point cloud into a structured, coarse-grained, top-down image using either a polar representation or a sparse grid representation. Scan Context++ [571, 576] and RING++ [711, 1208] are examples of approaches using this representation[3]. The former proposed a yaw alignment matching algorithm to achieve orientation invariance, and the latter theoretically proved its invariance and awareness by leveraging the Radon transform.

Range images As an alternatively to using a 3D point cloud, a range image (see Section 5.1) provides a structured, well-aligned representation of a scan. The recent advancements of dense multi-beam LiDAR technology (see Section 8.1), has made representing LiDAR data as a dense range image a much more suitable approach. The advantage here is that approaches can directly borrow well-established tools from the computer vision field such as convolutional neural networks (CNN) [616] or Vision Transformers [288] to extract a detailed feature representation. OverlapNet [195, 728] showed that yaw invariance can be achieved by applying an overlap loss to the range image, while more recently FRAME [1034] demonstrated LiDAR place recognition in mines using range images.

8.3.2.3 High-level or Combined Descriptors

To derive a descriptor for a scan (either local or global), there have been also approaches proposed that use a hybrid strategy or even learn directly a similarity function for place recognition.

Segmentation-based approaches: As discussed previously, representing a scan with a single descriptor can be vulnerable to structural aliasing. Because of this attempts which describe a place using a set of meaningful objects or segments have been proposed to increase uniqueness and descriptiveness and to avoid perceptual aliasing. These methods include SegMap [293] and InstaLoc [1269].

Descriptors which bridge between local and global: The previously introduced global descriptors are typically effective only when the point cloud projections are consistently aligned in a specific direction (*i.e.,* top-view or spherical-view). This requirement may restrict the application of the method to vehicles traveling with predictable directions along roads. To address this issue, BTC [1253, 1254] was proposed that utilizes both local and global descriptors. This approach aims

[3] As an example RING++ [1208] uses a representation of 120 by 120 pixels equally spaced over a $[-80, 80]$ meter grid.

to maintain the local geometry and the overall structure of a scan, effectively combining the strengths of each type of descriptor.

Direct 3D Data Processing More recently, data-driven approaches have been proposed that can achieve retrieval using the raw unstructured points directly without handcrafted rules. In particular, deep learning-based methods [1123, 162] have been proposed to extract robust point-wise features that are resilient to the diverse sensor sparsity and local surface distributions. In particular, Cattaneo et al. [162] showed that a pipeline designed with a triplet loss for place discrimination and differentiable relative pose estimation can achieved improved registration and overall benefits LiDAR SLAM. This can also be interpreted as a evidence that focusing on awareness can enhance both invariance and discriminative capabilities.

8.3.3 Summary

In summary, LiDAR place recognition has the same role as visual place recognition and shares common attributes and its performance is defined by the same metrics.

The sensor modalities and related techniques are often complementary as many of the locations where LiDAR place recognition fails, visual place recognition succeeds; and vice versa. In a mobile robot navigation system, both sensor modalities are often used to ensure robust and reliable results.

In the next section, we will describe how LiDAR place recognition can be used with pose-graph optimization to correct the unavoidable drift which occurs with LiDAR odometry so as to form a consistent, accurate and scalable LiDAR SLAM system.

8.4 LiDAR SLAM

The purpose of the LiDAR odometry systems described in Section 8.2 is to estimate a *locally consistent* motion of the sensor as it moves through the environment. However, this motion estimate will inevitably accumulate drift as the device travels a long distance.

To counteract this drift, a LiDAR SLAM system can maintain a *globally consistent* estimate for the entire history of the mapping operation by recognizing when the sensor returns to previously visited parts of the environment. These recognition events are known as *loop closures* and they can be used by the SLAM system to correct not just its current pose estimate but also to revise the full trajectory of previous pose estimates — as well as the corresponding map representation.

A key property that we seek for a LiDAR SLAM system is that it maintains a globally consistent map. This requires the system to achieve consistency between our current observations and past observations a single map representation. This task is particularly challenging in large-scale and potentially dynamic environments.

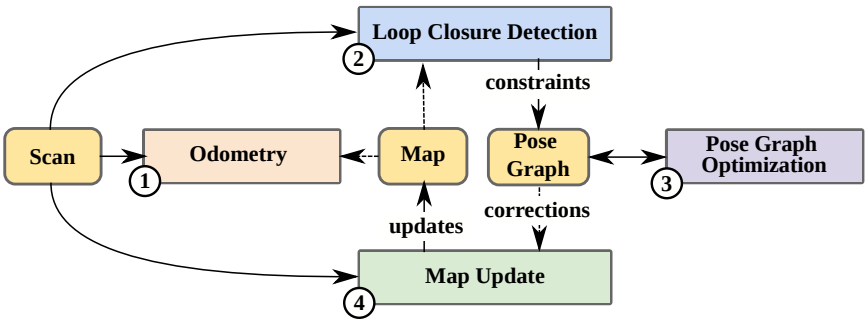


Figure 8.7 Conceptual structure of a typical LiDAR SLAM system composed of multiple components: (1) Odometry estimates the robot/sensor pose, (2) Loop Closure Detection determines if a place has been revisited, (3) Pose-graph optimization uses the loop closure constraints to correct the pose trajectory using factor graph optimization, and (4) Map Update uses the most recent pose trajectory to revise the map representation.

In Section 8.2, we discussed the development of approaches for LiDAR odometry which can achieve drift rates as low as one meter per kilometer traveled and then in Section 8.3 we reviewed different methods which carry out LiDAR place recognition to determine loop closures. LiDAR SLAM encompasses the techniques necessary to bring these components together to maintain a consistent trajectory and map representation — and to do so in real-time while running on-board a robot, vehicle or sensing system.

Most contemporary LiDAR SLAM systems [73, 852, 290, 273, 1248] are composed of the components shown in the system diagram in Figure 8.7. In the following Section 8.4.1, we will discuss this structure in more detail. We will then focus on the key steps of backend optimization and map update in Section 8.4.2 and describe some common techniques for integrating loop closures to correct the robot trajectory during backend optimization.

Advanced topics for LiDAR SLAM include multi-session and multi-robot mapping. These topics focus on how to fuse multiple mapping sessions into a common global reference frame — which can be either concurrent (running live on multiple robots and solved in real-time) or long-term (aligning multiple maps over time so as to infer environmental change). Multi-session and multi-robot SLAM will be covered in Section 8.4.3, where we will also provide an overview of common solutions and challenges.

Finally, as LiDAR SLAM has matured, it has brought into focus other advanced topics. Safety critical systems such as autonomous vehicles rely on SLAM so we have to consider the robustness and the scalability of LiDAR SLAM system. These topics will be discussed in the final part of this chapter in Section 8.5.

8.4.1 Structure of a LiDAR SLAM System

When implementing a LiDAR SLAM system [73, 852, 290, 273, 1248] it is common to decompose it into several modules, one module maintains a relative pose estimate using an odometry estimator, and other modules identify and use loop closures to re-estimate the pose trajectory and to then update the associated map representation to account for this revised pose trajectory. A LiDAR SLAM system usually consists of an odometry estimation module that runs at the sensor frame rate of the LiDAR sensors (*e.g.,* 10 Hz) with the other modules operating at a lower rate (*e.g.,* 1 Hz.).

More concretely, consider Figure 8.7. Here, (1) an odometry component estimates a pose in a frame-to-map fashion using the currently active local map. For the odometry module, the most common approach is to register the incoming laser scan to a rolling/active local map (as discussed in Section 8.2). This odometry module will typically only have access to data from the direct vicinity of the sensor — often called an active local map.

Next, (2) a LiDAR-based place recognition method identifies potential loop closure *candidates* as discussed in Section 8.3. Place recognition only identifies that two places (or more specifically observations taken at those two places) are similar. To determine a precise relative transformation estimate between those two places requires fine registration of the corresponding LiDAR scans (typically using ICP).

To initialize the registration, a sufficiently good initial guess of the relative transformation is needed. Geometric priors (taken from the existing pose-graph) can be used for small pose-graphs. Where no geometric prior can be used, modern global registration methods which do not rely on an initial guess but can robustly estimate a relative transformation have been developed [1219, 671]. These methods work well in situations with low overlap between the pair of scans.

Heuristics, such as the travel distance or time difference between two loop closure candidates or the degree of confidence in a RANSAC-based alignment for geometric verification, can be used to determine the validity of a loop closure candidate and to avoid adding false loop closures to the pose-graph.

Module (3) is the backend pose-graph optimization (PGO) step. It uses the full set of SLAM constraints to solve for an optimized pose-graph and to update the pose trajectory. We will discuss PGO in more detail in the following section.

Finally, in (4) a map update mechanism integrates the sensor measurements into a combined map representation according to this corrected trajectory. There are several potential approaches for this. The most common approach is to use the points directly [264] while approaches such as surfels [290, 852, 73] and implicit representations [1248, 273] attempt to improve the quality of the underlying map or seek to achieve a stronger probabilistic foundation. We refer the reader to Chapter 5 for technical details and discussion about the different dense map representations.

In the next section, we will discuss backend pose-graph optimization in more detail and how the full map representation is typically updated.

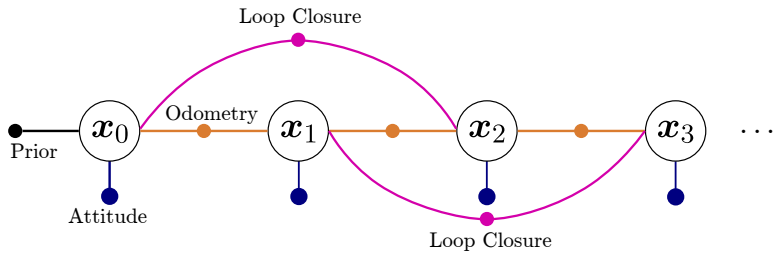


Figure 8.8 A SLAM problem represented as a pose-graph. Each node represents the pose of the sensor whereas the edges represent the constraints coming from odometry (orange) and loop closures (magenta). A Prior Factor fixes the graph origin. Optional Attitude Factors can be used to constrain the pitch and roll when inertial sensing is available. From [894].

8.4.2 Pose-graph Optimization and Map Update

The key part of a LiDAR SLAM system is updating the pose trajectory and map representation after a loop closure has been proposed and verified. As the local pose estimate will contain drift, the error in the local pose estimate needs to be accounted for in an updated pose trajectory. Additionally the existing map representation, integrating the past measurements, will also need to be updated.

Pose-graph optimization is sometimes known as the *SLAM backend*. It is the module which corrects the estimated trajectory to respect both the odometry constraints and loop closure constraints identfied when revisiting already observed places. As introduced and discussed in Part I, see Chapter 1, a factor graph can be used to represent these constraints (as shown in Figure 8.8). Because the graph is made up of only relative pose constraints, it is commonly referred to as a pose-graph. An example of a point cloud map before and after loop closure detection and pose-graph optimization is shown in Figure 8.9.

The constraint set can be optimized using general-purpose solvers such as g2o [622] and GTSAM [261]. To achieve real-time performance, with a pose-graph of increasing size, it is necessary to iteratively resolve a continually growing optimization problem. However, the constraint set is typically sparse — with few interconnected edges. Sparse matrix factorization methods which reorder and relinearize the underlying system of equations allow graphs of over 1000 nodes to be updated in a fraction of a second. For further reading please see iSAM2 [540] and HOG-Man [409].

Note that while 1000 nodes corresponds to a large pose-graph, one must consider scalability. It is common to add odometry constraints only relatively sparsely — not at sensor rate (*e.g.,* 10 Hz) but instead every few meters traveled. Another approach is to subdivide the mapped environment into submaps of fixed physical size (say 30 m traveled) each with a corresponding pose-graph node. It is then assumed that within these submaps the odometry will be locally accurate making re-adjustment

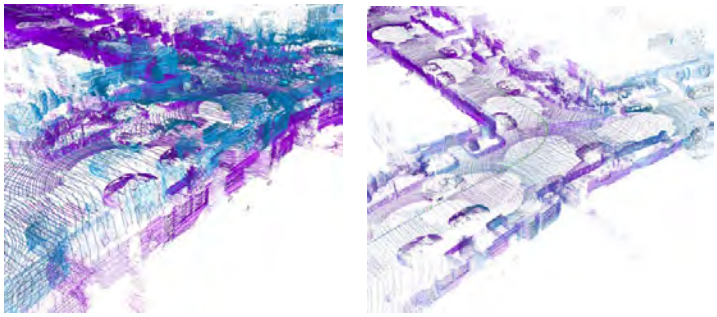


Figure 8.9 By incorporating loop closure constraints, a SLAM system can create a globally consistent map of revisited locations. The left shows the odometry-only trajectory of a revisited place with visible misalignment between the original visit (blue) and the current visit (purple) to this junction. The right shows the result after pose-graph optimization when loop closures have been integrated resulting in a consistent map of the road junction.

of the trajectory inside the submap unnecessary. This approach allows pose-graph SLAM to scale to city-sized maps.

In the backend, pose-graph optimization has to account for pose estimation errors by the odometry as well as errors in the loop closure constraints. To properly model the uncertainty in the pose estimate of the odometry poses, we can also estimate data-driven covariances to distribute the error in the pose-graph optimization sensibly [630, 165] — for example using high covariance of edge constraints where the drift rate is likely to be higher. Finally, to account for incorrect loop closure constraints and bad configurations of the pose-graph, there is a body of research into methods for robust pose-graph optimization [19, 159, 1064, 1065, 1217] which can down-weight, disable or ignore pose-graph constraints which would otherwise cause the map to degrade or diverge.

After pose-graph optimization, the full robot trajectory will now be globally consistent, but the effect of this update also needs to be reflected in the map itself. For this purpose, a common approach is simply to re-build the map using the past observations. This would require a system to store the previous observations indefinitely — which can quickly become unsuitable in large-scale environments. An alternative approach is to deform the map representation [852] or to directly link map elements (*i.e.,* such as surfels or submaps) to poses, to allow map deformation in a more scalable manner.

8.4.3 Multi-robot and Multi-session LiDAR SLAM

With the development of mature single-robot single-session LiDAR SLAM systems, there is interest in extending these systems to support multi-robot and multi-session

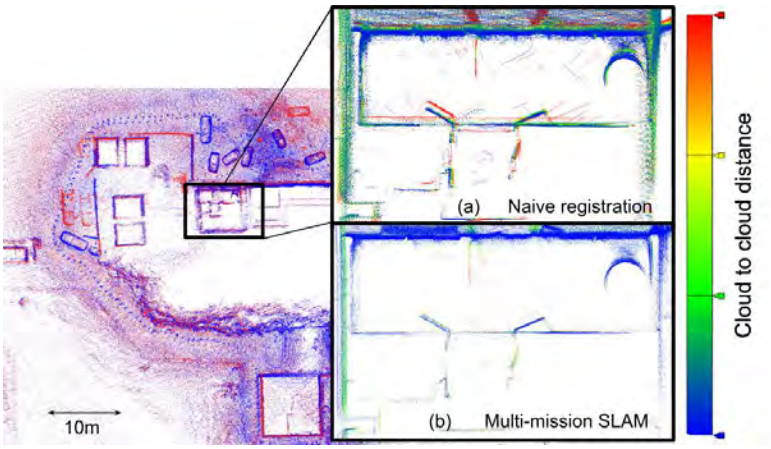


Figure 8.10 Comparison between (a) naive direct alignment of two global point clouds and (b) multi-mission pose-graph relaxation. (a) Point-to-point distances between the two global point clouds shows "double walling" causing phantom change to be hallucinated. (b) Multi-mission relaxation reduces point-to-point distances with structures being more clearly reconstructed. From [956] (©2024 IEEE).

applications. This would be useful because it would allow incomplete maps to be extended into newly scanned territory or for multiple field robots to coordinate their activities using a common map representation. Another use is to co-register maps taken in the same area over time to infer longitudinal environmental change, *e.g.,* for security or monitoring applications.

One initial point which is necessary to make is that while modern LiDAR SLAM systems are increasingly accurate — with **one meter drift per kilometer** being typical in open space — there will always remain some small error within a SLAM map. Simply taking the final point cloud map from two individual SLAM missions and co-registering them will result in locations where point cloud alignment is inconsistent as shown in Figure 8.10.

Initial work in this space focused on how to carry out joint backend optimization of multiple mapping sessions. One approach is to simply transfer the individual constraints from the set of SLAM instances into a single global map representation. An alternative approach is to build each map individually — each with their own coordinate frame, nodes and edges. To link them to one to another, Kim et al. [570] introduced an auxiliary variable called an *anchor node* which accounts for the different map coordinates of the individual SLAM missions. This node allows easy global alignment of each individual map and has been used for both multi-session visual SLAM and LiDAR SLAM [756, 573].

As described in [292], aside from the backend optimization, one must determine how loop closure constraints can be established between entirely disconnected

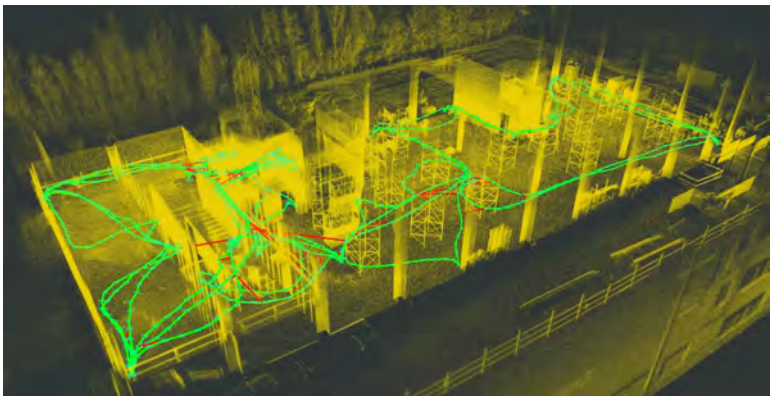


Figure 8.11 A multi-session SLAM map of a construction site. Five different mapping sessions are merged together by establishing inter-session loop closure constraints (in red) and adjusting a joint optimization of the five mapping session trajectories (in green).

SLAM missions. Unlike in the single session SLAM case, there is initially no geometric prior to form the first inter-mission constraint — with multi-session SLAM relying entirely on place recognition to relate maps to one another.

8.4.3.1 Multi-robot SLAM

Real-time multi-robot SLAM goes one step further — solving the multi-session mapping problem, but doing so with data collected in real-time by robots operating in the field. Estimating a combined map from multiple platforms in real-time allows a robot team to coordinate mission planning, optimally select frontiers of exploration, and to avoid wasted effort returning to a location mapped by another robotic team member. Achieving this capability can allow a team of robots to operate in concert — efficiently exploring territory, identifying which routes are free of obstacles and perhaps identifying people or objects of interest. This capability is relevant for search and rescue as well as military applications.

One can distinguish between systems which are centralized or decentralized. Centralized systems may transmit sensor measurements to a base station and then compute a combined map at that location. This may be so that the field robot's compute and sensing is kept as simple as possible, for example the mobile robots used by Amazon and Ocado for warehouse operations. On the other hand, decentralized (or distributed) systems would instead build a SLAM map on each individual robot with merging of the set of robot maps being achieved at a base station.

To describe the evolution of the state of the art we refer to two major international multi-robot exploration challenges. The first one is the Multi Autonomous Groundrobotic International Challenge (MAGIC), which was held in Brisbane, Australia in 2010. This challenge involved teams of wheeled robots executing a reconnaissance

Figure 8.12 Multi-robot SLAM progressed from 2D to full 3D between the 2010 MAGIC challenge to the 2021 DARPA SubT Challenge. The pictures show the winning University of Michigan and Cerberus Teams from the two challenges. Image courtesy of Edwin Olson and Cerberus team.

mission in a 500 m × 500 m challenge area to correctly locate and classify simulated threats. The winning team, Team Michigan, fielded 14 3D-printed robots equipped with 2D LiDAR scanners [834] as well as cameras (to identify loop closures). Each robot carried out 2D LiDAR odometry and transmitted its pose-graph constraints to a base station which assembled a global 2D multi-robot map.

Research progress over the last decade was evidenced by the DARPA Subterranean Challenge [608] which was held in Louisville, Kentucky in 2021. It posed a similar challenge to competing teams at MAGIC — to explore unknown environments — but with the robots operating in more complex 3D underground environments with stairs, kerbs and ramps. Here 3D multi-beam LiDAR was heavily used but also augmented with visual odometry, wheel/legged and, in some cases, thermal odometry to overcome degenerate circumstances where LiDAR odometry can fail such as in narrow environments in the underground tunnels.

The SubT teams published an overview article which provides a comparison between the fielded systems [307]. Each team used a semi-decentralized approach with individual robots building pose-graph-based SLAM maps on board with a multi-robot SLAM map created at a single central base station. Key challenges included compression and communication as the robot teams needed to maintain a dynamic wireless mesh network to transmit data back to this base station. For example, the WildCat SLAM system from CSIRO [609] was notable for using a compressed surfel representation to represent local submaps. These surfel maps took up much less space than the raw point clouds — greatly reducing the bandwidth needed to transmit the map and pose-graph constraints to the base station computer. During the finals, a complete map took only 21.5 MB per robot.

As mentioned above, the most complex problem is fully distributed SLAM system — where each robot platform is tasked with building and maintaining a representation of the overall combined map subject to communication and scaling constraints.

Some existing approaches [500, 1099] have explored how to do this and focused on the mechanisms to share the set of constraints and local submaps progressively with each robot. Issues of consistency are key in this topic.

8.5 Outlook and Futures Challenges

LiDAR SLAM has seen significant advancements over the last decades — especially since the introduction of LOAM [1264]. Improved odometry with high accuracy [1207, 1005, 1137, 264] and efficient pose-graph SLAM systems [73, 273, 852, 910] have also been developed. However, despite this progress, there are still unsolved problems and challenges to tackle.

Robust and Resilient Perception A recent robustness evaluation by Zhao *et al.* [1283] identified that LiDAR SLAM systems struggle to perform effectively in cluttered and unstructured environments. Structure-less corridors, underground mines and extreme weather conditions such as snow, fog, and dust are other challenging situations pointed out in a review by the DARPA Subterranean Challenge competitors [307].

Furthermore, the performance of current LiDAR SLAM systems is typically demonstrated experimentally and lack formal robustness evaluation. Best performance is achieved through feature engineering and manual parameter tuning, which perhaps ought to be dynamically adjusted according to the operational scenario as in KISS-ICP [1137]. Future improvements in this regard should consider actively adapting algorithm behavior to account for changes in the environment through introspection.

With regard to place recognition, there are a broad spectrum of research directions. Key survey papers such as [1013, 1244] offer a comprehensive foundation on the topic. Ongoing research includes methods for robust retrieval which generalize across the various categories of LiDAR sensors [536]. Other research looks to achieve heterogeneous place recognition between LiDAR and other modalities such as radar [1243] and OpenStreetMap [219]. Researchers have also successfully leveraged the LiDAR's intensity information [1006, 1151], alongside traditional XYZ data to improve performance. Long-term place recognition across multiple mapping sessions [574] is another promising research topic; as is change detection and lifelong map management [573, 1246].

Multi-sensor Fusion Fusing multiple sensors with complementary characteristics is a key route to more robust and resilient robotic systems. Degraded perception of a particular sensor can be ameliorated by fusing a different and complementary sensor, *e.g.,* Radar works well in rain or smoke; or using visual feature tracking in a tunnel where LiDAR fails [1188, 1282]. However, when integrating additional sensors with LiDAR, we inevitably acquire a plethora of sensor data, leading to

redundancy. There are open questions about how to achieve a balance between redundancy and lightweight computation. Furthermore, one must consider how to efficiently select the most reliable information when fusing estimates of multiple sensors. Solutions range from early fusion approaches (using a single tightly coupled estimator) and late fusion approaches (where separate individual-sensor pose estimators are combined).

Finally, another practical consideration is that multi-sensor systems may lack accurate calibration and individual sensors may not be precisely synchronized. This places a burden on the underlying estimation system making full, tight sensor fusion difficult to practically use repeatedly. There is still space for research into these intriguing research questions.

Uncertainty and Bayesian Estimation Closely related to the question of how sensors can be reliably fused is the question of uncertainty estimation in LiDAR SLAM. Properly calibrated measures of sensor uncertainty are required to probabilistically fuse multiple pose estimates. However, most successful LiDAR-based approaches rely on ICP, which cannot provide a calibrated and robust estimate of the pose uncertainty.

While some early approaches [630, 165] for approximating covariance exist, the estimated uncertainty used in LiDAR SLAM system is often unreliable. As a result, algorithms often use fixed odometry covariances during backend pose-graph optimization. A more introspective handling of uncertainties in the odometry process has the potential to address degraded pose estimates at an earlier stage. There are still many open questions regarding uncertainty estimation, where robust solutions would support the development of more resilient SLAM systems as well as multi-modal SLAM.

Deployment in Closed Loop Autonomy Systems A final consideration is how LiDAR SLAM performs in a desired final application. These applications are as varied as light-weight aerial vehicles flying through forests, handheld devices scanning construction sites and self-driving cars operating in poor weather. The traditional electronics parameters of Size, Weight and Power (SWaP) are augmented with additional parameters of accuracy, robustness, computation and latency.

The ability for a self-driving car to respond to a potential upcoming collision with as little delay soon as possible is affected by the computational latency of the LiDAR system. Thus in practical application the most accurate and computationally complex system may not always be the preferred solution.