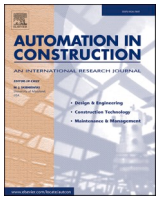
Automation in Construction 162 (2024) 105344

Contents lists available at ScienceDirect

Automation in Construction

journal homepage: www.elsevier.com/locate/autcon



Review of simultaneous localization and mapping (SLAM) for construction robotics applications

Andrew Yarovoi [a], Yong Kwon Cho [b,*]

[a] George W. Woodruff School of Mechanical Engineering, Georgia Institute of Technology, 790 Atlantic Dr., Atlanta, GA 30332, USA
[b] School of Civil and Environmental Engineering, Georgia Institute of Technology, 790 Atlantic Dr., Atlanta, GA 30332, USA

ARTICLE INFO

Keywords:
Simultaneous Localization and Mapping (SLAM)
LiDAR
Robot
Construction

ABSTRACT

With the increasing affordability of robot technologies and the reduction in size and weight of autonomous systems, the deployment of autonomous robotic systems on construction sites has gained significant attention. One major challenge faced by these systems is the accurate mapping and localization within an environment that constantly evolves on a daily basis. This computational problem, known as simultaneous localization and mapping (SLAM), has garnered substantial interest, leading to the proposal of various algorithms in recent years. This paper aims to provide a comprehensive overview of the SLAM algorithms by presenting the necessary background information, discussing the challenges involved in general, examining common approaches, and highlighting recent developments. Furthermore, the specific considerations related to deploying SLAM in construction environments are addressed. In order to provide practical insights into the advantages and use cases of state-of-the-art SLAM algorithms, a rigorous evaluation is conducted within a construction environment

1. Introduction

In the last few decades, the field of robotics has undergone tremendous growth. Fueled by the greater availability of hardware and the decreasing size and cost of computing resources, autonomous robotic systems promise to revolutionize how people work by automating repetitive and dangerous tasks, boosting productivity, and combating labor shortages. Many sectors, such as warehousing, have already demonstrated that robotic automation can boost productivity and safety [1]. However, construction has been slow to adopt robotic automation. One of the challenges preventing adoption is the dynamic nature of construction sites, making it difficult to set up infrastructure. Thus, mobile systems are necessary to provide greater flexibility.

An important requirement of mobile and aerial robotic systems is generating an internal representation of the robot's environment (mapping) and tracking the location of the robot relative to points of interest and nearby obstacles (localization). Unlike in warehousing, setting up external localization systems such as motion capture systems and visual markers is difficult due to a construction site changing daily. Furthermore, operation on construction sites often requires the robot to navigate indoor environments and requires high precision locational data, making GPS-based systems unreliable. Thus, a solution is necessary for precisely localizing the robot within an unexplored environment without the use of GPS, maps, or any sort of infrastructure.

Simultaneous localization and mapping (SLAM) techniques attempt to address these challenges by using onboard sensors to iteratively build a map and estimate the pose of the robot within the map. Since combining the readings into a unified coordinate frame requires knowledge of the robot's position and finding the robot's pose requires a map of the environment, SLAM algorithms must use probabilistic and iterative methods to model the sensor and motion noise to generate cohesive map and trajectory estimates. These estimates can then be used for navigation, or a plethora of other tasks such as 3D model reconstruction, geometry quality inspection, construction progress tracking, and safety management [2].

Most SLAM algorithms follow the same general steps (outlined in Fig. 1. At each timestep, features are extracted from data generated by onboard sensors. These features are then matched to previously seen ones stored in the robot's internal map, using the previous predicted pose to improve matching. The identified correspondences are then used to fuse the features with the map, jointly refining the features' poses and the robot's current pose. The poses of previously unseen features are also added to the map based on the estimated current pose of the robot, thereby iteratively growing the map. This process is repeated to localize

* Corresponding author.
E-mail addresses: ayarovoi3@gatech.edu (A. Yarovoi), yong.cho@ce.gatech.edu (Y.K. Cho).

https://doi.org/10.1016/j.autcon.2024.105344
Received 12 June 2023; Received in revised form 19 January 2024; Accepted 24 February 2024
Available online 11 March 2024

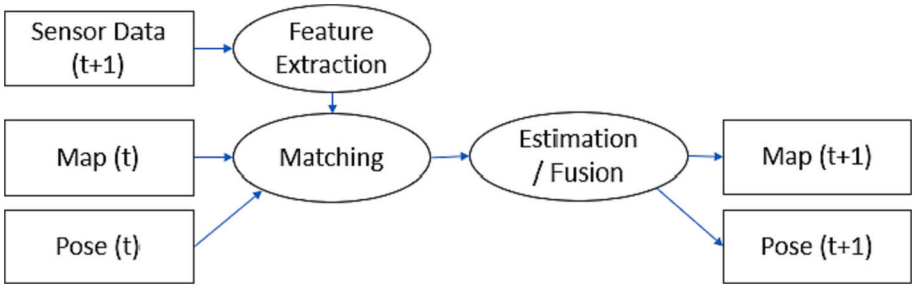


Fig. 1. Block diagram of iterative SLAM process.

the robot and simultaneously generate a map of the environment.

In this paper, we present an overview of SLAM, discuss its application for autonomous robots in construction environments, review its limitations, and demonstrate its application on a construction dataset. The rest of the paper is organized as follows. Section 2 reviews some background information on the SLAM process. Section 3 discusses some limitations of SLAM and factors of consideration for deployment on construction sites. Section 4 gives a brief overview of some popular SLAM algorithms. Section 5 compares several open-source algorithms on a construction dataset and discusses their observed limitations.

2. SLAM background

The main purpose of SLAM is to generate a geometrically consistent map of the environment along with the trajectory that a robot follows within that map. There are many different approaches for performing SLAM, but most SLAM algorithms can be classified using three distinct attributes: computation time, sensor fusion approach, and sensor type.

Most SLAM algorithms can be categorized as either online or offline approaches. Online SLAM algorithms are primarily used by autonomous robots for localization, a critical component of real-time navigation and obstacle avoidance. These algorithms attempt to incrementally recover the current robot pose based on the previous predicted trajectory and accumulated map [3]. As most applications of online SLAM aim to recover real-time location estimates, these algorithms tend to prioritize computational efficiency over the quality of the map. Computational efficiency is crucial because robots often have limited computational resources due to size and weight constraints. Online SLAM algorithms assume real-time data input and typically optimize over recent or nearby poses rather than the entire trajectory. On the other hand, offline SLAM algorithms are employed after all the data has been collected and typically achieve improved map quality. They are not bound by real-time constraints and can utilize the processing power of more powerful desktop computers, enabling the use of computationally expensive techniques. Additionally, these algorithms have access to the full trajectory, allowing for batch optimization over the entire path, resulting in improved map and trajectory estimates [3]. Therefore, online SLAM algorithms are suitable for autonomous navigation, while offline SLAM is more appropriate for generating survey maps or creating as-built Building Information Models (BIMs).

Regardless of whether a SLAM algorithm is designed to run online or offline, most SLAM algorithms follow a similar architecture. They assume that the starting position is fixed at the map's origin and that the robot starts off static in the inertial frame. Using these assumptions, they initialize a map specific to the type of sensor being used. Commonly, this map consists of a set of easily detectable features, though some algorithms use discretized maps that track occupancy instead [4]. Features must be chosen carefully, as larger sets of features incur longer computation times. Features should also be easily recognizable from different viewpoints and unique enough to enable matching between time steps [5]. Some algorithms also track the uncertainty of their feature's positions so their locations can be updated in future iterations. These features are often referred to as landmarks [6]. Many other algorithms tie the features to a specific sensor pose to reduce the complexity of the problem [7–9]. In the next step, the algorithm uses odometry and control inputs to create an initial guess for the current pose. Feature-based algorithms then extract features and match them to existing features in the map using proximity or similarity constraints. Point-based methods often skip the feature extraction step and instead directly match points to nearest neighbors [10]. These correspondences are subsequently incorporated either as sensor measurements and optimized through sensor fusion techniques to jointly optimize both the features' positions and the robot's pose, or employed iteratively to converge on the robot's pose in relation to the constructed map, which is then added as an odometry measurement and optimized using sensor fusion techniques [11]. For online-SLAM, this process is repeated iteratively to track the robot's pose and grow the map of the environment.

There are many different sensor fusion techniques, but three popular approaches are particle filters (sometimes referred to as sequential Monte Carlo methods) [12], Kalman filters [13], and graphical

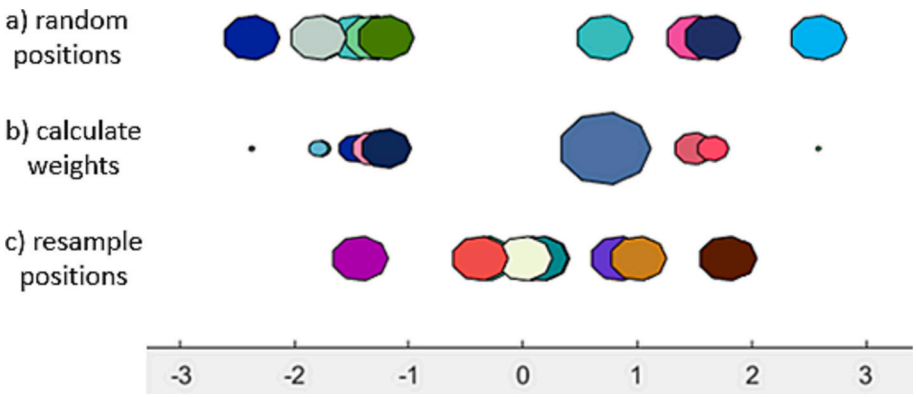


Fig. 2. Particle filter example: a) randomly sample positions b) given sensor readings, weigh particles based on likelihood (correct position is 0 in this example) c) resample positions based on weights.

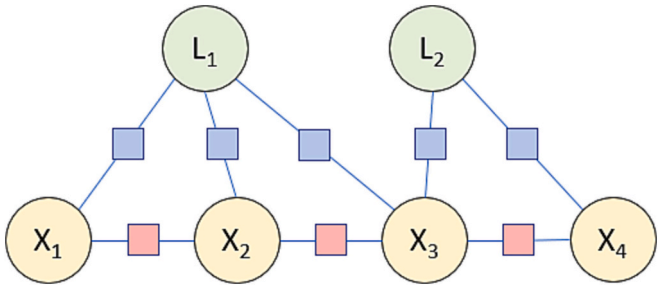


Fig. 3. Example factor graph, with yellow circles representing robot positions, green circles representing landmark positions, red squares representing odometry factors, and blue squares representing landmark observation factors. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

representations combined with least-squares optimizers [14]. Particle filters attempt to model the uncertainty in the robot's pose by generating many hypothetical positions of the robot (Fig. 2a), and then using the sensor readings to weigh the poses based on their likelihood (Fig. 2b). Particle filters then update the hypothetical positions by resampling them based on their weights (Fig. 2c). The last two steps are repeated, causing the particles to converge around the robot's position. While particle filters perform well in 2D environments and can directly model non-linearity [15,16], they often become intractable when used with 3D maps with many features.

Kalman filters provide a popular alternative to particle filters. Two varieties of Kalman filters, the Extended Kalman filter (EKF) [17] and the Unscented Kalman filter (UKF) [18] are particularly popular options for SLAM. Kalman-based approaches use Gaussians to model the uncertainty of the robot's and features' poses and provide optimal solutions for linear systems with Gaussian noise. EKFs and UKFs are built based on standard Kalman filters to address the nonlinear nature of orientations. While EKFs are more common, UKFs are slightly more robust to nonlinear systems [18]. The main disadvantage of Kalman Filter approaches is that they rely on storing and inverting a covariance matrix, which scales quadratically in memory usage and cubically in computation time with respect to the number of elements in the tracked state (robot pose plus number of tracked landmarks) [3]. Thus, additional modifications and simplifying assumptions are required to achieve real-time performance in large-scale environments [19].

Recently, factor graphs optimized via nonlinear least-squares optimizers have become a popular alternative to Kalman methods. Factor graphs use variables to represent unknown quantities in the problem and factors to represent functions on subsets of the variables. For example, the robot's pose at each timestep and the landmarks' positions may be modeled as variables, and the error functions associated with each odometry and sensor measurement may be modeled as factors (Fig. 3). In a graph representation, factors are connected to the variables they depend on. For example, the landmark observation factor between X_1 and L_1 may represent a distance measurement error from X_1 to L_1, which depends on the unknown poses those two variables represent. This representation of the problem generalizes it, enabling factor graphs to directly model nonlinear, non-Gaussian processes. Factor graphs also tend to scale better to higher-dimensional maps due to memory usage scaling linearly with the number of elements in the graph, allowing exploitation of the sparse nature of most SLAM problems [20]. Another advantage of graph representations is that they can be optimized by a variety of optimizers, such as DCS [21] or iSAM2 [22], enabling greater flexibility in the desired computational complexity, accuracy, and solution properties. Their main disadvantages are they can sometimes be computationally expensive to optimize and can be complex to implement from scratch, though popular open-source frameworks such as g2o [23] and GTSAM [24] have dramatically simplified the implementation process.

The final and perhaps the most impactful attribute of a SLAM algorithm is the type of sensors that are used. All SLAM approaches require sensor information to build the map. Common sensors include ultrasonic sensors, cameras, LiDARs, and Inertial Measurement Units (IMUs). Ultrasonic sensors are commonly used for underwater exploration and are out of the scope of this paper.

Cameras are a popular choice for SLAM as they are low-cost, provide abundant sensor information to extract features, and are lightweight [25–28]. A popular type of camera for SLAM is RGB-D, which in addition to regular colour information, uses infrared projections or a stereo setup to estimate the depth of each pixel [29]. This gives an estimate of depth, simplifying the estimation of the 3D positions of features. Unfortunately, the depth information is usually short-ranged and may be unreliable in varying lighting conditions. Additionally, cameras that use infrared projections are typically impacted by direct sunlight, limiting their use to indoor environments. Most RGB-D cameras also have smaller fields of view (FOV) than LiDAR sensors [30]. Typically, SLAM algorithms designed for cameras (visual SLAM) will extract many key points, which are point features such as corners that can be reliably tracked between frames. These key points will then be matched between frames to estimate the camera's movement. A large number of key points make particle filters intractable, and thus, EKF or least-squares approaches are commonly used. This reliance on key points makes Visual SLAM algorithms sensitive to lighting conditions and unreliable in textureless environments such as empty hallways and rooms with many plain walls

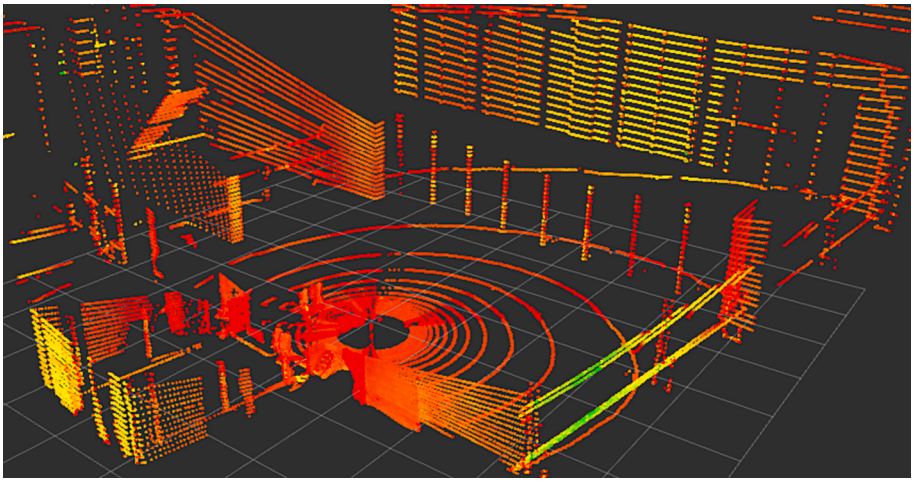


Fig. 4. Example point cloud from the Hilti SLAM Challenge Dataset, Exp 04 [38].

[31]. RGB-D cameras' smaller FOV (typically 60–90°) can also cause SLAM algorithms to lose tracking features and, by extension, the robot's pose. These issues make visual SLAM less desirable in complex environments [32]. Construction sites often mix indoor and outdoor environments, have harsh lighting contrasts (especially during night shift work), and include many unfinished textureless environments (e.g., empty rooms with drywall, unfurnished hallways, etc.), which make them challenging environments for Visual SLAM.

Due to these limitations, LiDAR is often the preferred sensor for SLAM systems deployed on construction sites. LiDAR is an active sensor that measures the time-of-flight of a laser to measure depth. By measuring thousands of returns a second, the sensor creates a point cloud of the environment (Fig. 4). As LiDAR sensors emit their own laser light, they are more robust to lighting conditions and perform similarly in both outdoor and indoor environments. Most LiDAR sensors for navigation also have a greater range (20–200 m) and a larger horizontal FOV (usually 360°) than depth cameras, enabling them to observe more of the environment at once [33]. In addition, their depth data tends to be more consistent and accurate than that generated from depth cameras, with fewer outliers and less sensor noise at moderate distances. This makes them ideal for construction sites and large-scale environments, where they have been used for tasks such as pavement 3D reconstruction [34], automatic evaluation of rebar spacing [35], and automatic BIM component extraction [36,37]. LiDARs do have their drawbacks, as they are generally larger, heavier, and more expensive than cameras [33]. They are also sensitive to highly reflective surfaces and glass, but their large FOV often offsets this issue.

Inertial measurement units (IMUs) measure the robot's linear accelerations and angular velocities and provide a magnetic heading. This data is often used in combination with cameras or LiDARs in SLAM to improve pose estimation. For LiDAR-based SLAM methods, IMUs can also provide an initial trajectory estimate which can be used to deskew the point clouds, improving the quality of the data [9,39]. Deskewing is the process of unwarping point clouds to remove distortions caused by the motion of the sensor during the scan period. Additionally, they can provide an initial position estimate, which can be useful for speeding up the convergence of iterative matching methods commonly used in LiDAR-based and camera-based SLAM algorithms [40,41]. IMU data can also be integrated to provide an additional odometry measurement source for visual-inertial SLAM algorithms [42].

3. SLAM limitations

3.1. Accumulation of errors

Most SLAM systems rely on relative sensor measurements, meaning that the sensor measurements are relative to the robot's position and not the global inertial frame. These relative sensors, such as cameras, LiDARs, and gyroscopes, include sensor noise, which causes uncertainty in the trajectory estimate. As the robot moves, the robot pose is estimated relative to a recent past position. This chaining of relative transformations, each contributing a small amount of uncertainty, leads to an accumulation of errors commonly referred to as drift. This drift results in SLAM producing highly accurate relative poses (and maps) over short trajectories. However, it yields significant inaccuracies when comparing the absolute position of poses far away from each other in time [43]. This can be viewed as the uncertainty of the pose growing with time in the inertial frame, as visualized and explained in [44]. To address this, a technique called loop closure is often used [45,46]. By detecting when the robot has revisited a previously explored space, we can improve the accuracy and reduce the uncertainty of the current robot pose by constraining it to a previous robot pose with less accumulated drift. This shortens the effective length of the trajectory, reducing pose errors. The improved estimate of the current pose can then be propagated back through the past trajectory, reducing the uncertainty of past positions [20]. Thus, loop closure improves the global consistency of the generated map and robot trajectory by correcting the large drift errors at loop junctions.

Another means of reducing drift is by providing some absolute measurements which are relative to the inertial frame. Common absolute measurements are GPS readings, magnetometer headings, or visual markers with known accurate global positions [47]. As the noise of these measurements is relative to the inertial frame, these measurements do not suffer from incremental drift [44]. A combination of relative and absolute measurements is ideal for accurate long-distance mapping. However, absolute measurements can be difficult to obtain in many real-world environments, including construction, so reducing drift is a crucial objective of many SLAM algorithms.

3.2. Degenerate environments and sensor degradation

Degenerate environments refer to environments in which the data received by the sensor cannot adequately constrain the robot's location due to a lack of useful information for feature extraction, symmetric or underdefined environment geometry, or excessive sensor noise. When SLAM algorithms encounter degenerate environments, they tend to experience large drifts in the estimated location [48]. Due to the iterative nature of most SLAM algorithms, this can result in a loss of tracking. This means that even after the robot exits the degenerate environment, it may not be able to relocalize with respect to the global map as its pose estimate error is too large or there are insufficient shared features between the global and current map for matching (see Experimental section for example).

What constitutes a degenerate environment heavily depends on the specific sensor and SLAM algorithm being used. For visual sensors using features, textureless environments or repetitive patterns are a common degradation source [32,48,49]. Even when operating in rooms with sufficient unique features, loss of tracking can occur if the camera's field of view temporarily experiences a textureless environment (e.g., the camera looking down at a textureless floor). Another common cause for degradation is rapidly changing lighting conditions. For example, when transitioning between indoor and outdoor environments, the camera may experience overexposure, leading to a lack of features and loss of tracking [50]. It can then be challenging to regain tracking as limited features connect the two environments. While the loss of tracking is a common problem in visual-SLAM, some visual-SLAM algorithms include explicit modules that attempt to detect the problem and initialize relocalization [51–53]. Despite this, the loss of tracking can still lead to erroneous position estimates for a navigating robot, which can lead to collisions. It can also result in permanent errors in the map [54].

LiDARs are more robust and have fewer sources of degradation. However, rooms with many highly reflective or glass surfaces can still pose issues for a robot. Some LiDAR-SLAM algorithms also rely on explicit floor detection to constrain the robot vertically. This can lead to the loss of tracking in environments where the floor is not captured by the sensor's field of view or in tight spaces where the floor cannot be seen from the sensor's position (see Experiments section). Highly symmetric environments, such as the inside of a pipe or shaft, can also pose issues as the change in position cannot be constrained from the observed geometry.

A common approach for reducing the influence of degenerate environments is to combine multiple sensors for SLAM, such as using a combination of a camera and a LiDAR [55,56] or combining LiDARs with IMUs [9]. Using multiple sensors can help in degenerate environments by allowing the system to rely on other sensors when one sensor is compromised. For example, in symmetric environments, IMUs can provide semi-accurate tracking for short intervals, preventing the loss of tracking [9]. Camera data could also help in geometrically symmetric environments by utilizing texture information, while LiDARs can help maintain tracking while the camera is compromised due to lighting changes [55]. While adding sensors can improve reliability, it can also introduce additional errors to estimates if sensor degradation is not

accurately tracked and accounted for during sensor fusion. For example, IMUs themselves can be degraded by vibrations and external magnetic fields, which if naively combined with the LiDAR data, can result in additional errors in position estimates (see LeGO-LOAM results for exp04 in experiments section). Magnetic interference is particularly pertinent to construction as ferrous materials such as steel and equipment's running engines are common sources of external magnetic fields. Thus, it is often recommended to either disable the magnetometer portion of an IMU or to explicitly take into account these magnetic disturbances [57] in SLAM systems operating in such construction environments. Despite these limitations, having multiple sensor types typically leads to improved reliability and accuracy [9,55,56].

3.3. Robot's fast motions

Fast and abrupt motions are more difficult for SLAM algorithms to track than slower and smoother motions (see Exp06 in the experiments section). In addition to the motion blur or skewing, which can make feature extraction and feature matching more difficult, fast motions cause reduced overlap between sequential sensor inputs. As SLAM relies on an iterative approach that finds transformations between frames, less overlap means fewer constraints between time steps, reducing the accuracy of the positional estimate. Running sensors at higher frequencies or including a high-frequency IMU can help mitigate the effects of fast motions. For LiDAR data, IMUs are particularly helpful as they can be used to deskew the LiDAR scan, reducing distortions in the map and improving localization [39].

3.4. Implications for construction

As stated previously, construction environments present many challenges for accurate SLAM implementation. For navigation and localization, LiDAR-based SLAM is generally preferred due to its robustness to common environments found in construction, such as mixed lighting conditions and low-texture environments. LiDAR also provides lower noise measurements primarily owing to their direct measurement of depth [58], leading to more accurate depth reconstruction and scale. Furthermore, LiDAR paired with IMU can provide more accurate mapping by deskewing scans before applying SLAM and attenuating the effects of fast motions and loss of tracking [9]. If using an IMU, disabling the magnetometer data or explicit handling of magnetic interference is often recommended in highly metallic environments or where large engines or motors are running, as it is easily corrupted [57]. For longer scans, adding sparse absolute measurements such as GPS or fixed markers is essential if large distance measurements will be computed from the map [43]. If absolute measurements are unavailable, then it becomes important for the user to understand the implications of drift and to take them into account when reporting measurements. The amount of drift is highly dependent on the environment and algorithm used, but two current state-of-the-art LiDAR-based methods [54,59], report 2–3% translational drift in an urban environment dataset.

Another consideration is that construction environments usually include rough and uneven terrain. This may cause 2D SLAM algorithms to be inaccurate as slopes and significant elevation changes can occur. 3D SLAM is thus often preferred. It also becomes important to consider the sensor's specifications. Rotating mobile LiDARs typically have a 360° horizontal FOV and a smaller vertical FOV (e.g., 30–45°) [33]. The smaller vertical FOV can make it difficult to detect the ground and accurately track elevation changes, especially in confined indoor spaces (see experiments section). Some common solutions include mounting the sensor at an angle and rotating it [59] or simply using a sensor with a larger vertical FOV (e.g., 90°) [60]. However, these solutions come with drawbacks such as reduced effective framerate, reduced vertical resolution, reduced ranges, or increased sensor cost. It can also be important to consider the range of a sensor, as short-range LiDARs (about 20 m) are more suited for indoor environments and may struggle to find features in outdoor construction environments. The number of channels or vertical resolution can also be an important consideration. Higher resolution, which is typically achieved by choosing mobile LiDAR sensors with more channels, can improve point cloud density and enable finer feature extraction. Mobile LiDARs commonly have much lower vertical resolutions than horizontal due to them spinning in only one axis. Choosing a LiDAR with more channels can result in more uniform densities and provide better coverage for down-stream tasks such as segmentation. However, LiDARs with more channels tend to cost more and provide more points per second, which can lead to increased processing times [33].

Finally, downstream tasks may influence the choice of sensors and SLAM algorithms used for construction. While LiDAR's are incredibly useful for localization, high-resolution images, thermal camera data, or even colorized point clouds may be desired for monitoring and inspection tasks. Additionally, while many approaches for semantic segmentation of point clouds have been proposed [61–64], point cloud segmentation is notoriously challenging due to the sparse and unorganized nature of point clouds. Thus, navigating robots often augment their maps with camera images to aid in their downstream automated tasks. In these cases, it is common to either use both LiDARs and cameras to create a tightly coupled SLAM system [55,56], or use only LiDAR-inertial SLAM system for localization and then use the predicted robot poses to project the camera data to the map [65]. The first approach has the advantage of enabling the use of texture information during localization but has the disadvantage of introducing the potential for additional inaccuracies due to poor visual features in degenerate environments. The second approach ignores the camera data during localization and thus has the same limitations discussed in prior sections. Thus, it is up to the developer which is preferable for their specific application. Both approaches can be used to create colorized point clouds, textured 3D reconstructions, and maps with geolocated images, which can be helpful for monitoring, inspection, and semantic segmentation tasks.

4. SLAM algorithms

4.1. Visual SLAM

There are many visual SLAM approaches, but arguably the most influential and widely adopted approach is ORB-SLAM [52] and its successors [54,66]. The open-source online algorithm works with monocular, stereo, and RGB-D cameras and tracks the robot's position by detecting and matching custom-designed ORB features and performing bundle adjustment using g2o's nonlinear optimizers [23]. The algorithm can also generate dense 3D reconstruction for smaller environments and sparse point cloud reconstruction for large-scale scenes but is primarily focused on generating online accurate robot trajectories. ORB-SLAM2 includes tracking, loop closure, and relocalization modules, enabling it to operate on large-scale scenes. The algorithm is compatible with almost any type of camera, although, like all visual SLAM algorithms, it suffers from significant scale drift when used with monocular RGB cameras. The latest version, ORB-SLAM3, also allows for multimap representations, improving loop corrections, and merging maps across multiple sessions.

Another popular alternative is Direct Sparse Odometry (DSO) [67]. Unlike ORB-SLAM and most visual SLAM approaches, it does not extract a sparse set of key points, match them using descriptors, and optimize their geometric error. Instead, DSO considers the entire image, selects a sparse set of points, and then optimizes the photometric error (difference in pixel values). This makes the algorithm more robust to weakly textured surfaces and enables it to utilize smooth gradient information such as smooth lighting changes across plain walls to localize the camera. The tradeoff is that direct methods are more sensitive to photometric and geometric distortions, such as large lighting changes, white balance changes, and non-modeled camera artifacts (e.g., rolling shutter

effects). Thus, while DSO can perform better in weakly textured or controlled environments, it is more sensitive to environmental factors and is less suited for off-the-shelf commodity cameras. The initial implementation of DSO was also missing loop closure detection, though a future extension, LDSO [68], added this functionality. [69] provides a more in-depth overview of other visual SLAM approaches.

4.2. LOAM derivatives

LOAM [8] has been a heavily influential work in the field of online LiDAR-based SLAM. It introduced a simple yet effective approach for detecting feature points based on the smoothness of the local surface in the range image. LOAM also introduced a popular architecture that splits lidar odometry (scan-to-scan matching for high-frequency pose estimation) and lidar mapping (low-frequency matching between a sweep of scans and a global map). The result was a 3D LiDAR-based SLAM capable of running in real-time on lower-end devices. The original work only accepted LiDAR as input and did not perform loop closure.

However, since its publication, many extensions of the work have been proposed. V-LOAM [55] combined LOAM with visual odometry from cameras to improve performance and allow for multi-modal SLAM. Loam-Livox [70] is another extension of LOAM which is optimized for solid-state LiDAR sensors. Unlike the popular rotating LiDARs (e.g., Velodyne Puck, Ouster OS product line), solid-state LiDARs are much smaller and cheaper but have a more limited horizontal FOV.

LeGO-LOAM [7] built upon LOAM's success and added explicit ground segmentation, improving feature matching and allowing for more efficient two-stage pose estimation. These changes make it faster and more accurate than LOAM. LeGO-LOAM also integrates a factor graph and adds loop closures based on Iterative Closest Point (ICP) [71] matching. Additionally, it can incorporate IMU and GPS data into its pose estimation, though neither is required. While it supports IMU data, LeGO-LOAM does not perform bias estimation, point cloud deskewing or add the IMU data to the factor graph. Instead, it combines the IMU orientation with the pose estimate through a complementary filter and uses the IMU data to generate an initial guess for aligning the point clouds. This means that small sensor biases or imperfect calibration can significantly degrade performance. LeGO-LOAM's reliance on explicit ground segmentation to constrain the roll, pitch, and z-height of the robot pose is also a limitation. While their implementation allows for slopes, the explicit ground segmentation causes the method to struggle in confined spaces and orientations where the ground is outside of the sensor's FOV.

LIO-SAM [9] is another extension of LOAM, which tightly couples LiDAR data with an IMU, allowing for the deskewing of the scans before processing and improving robustness. LIO-SAM offers the same basic functionalities as LeGO-LOAM but is generally more accurate when paired with a high-quality IMU. Improved performance is achieved by modeling the IMU odometry as factors in the factor graph and performing explicit bias estimation. However, this restricts its use to robots with IMUs as IMU data is required.

All of these approaches are open source and rely on the same features first introduced in LOAM.

4.3. Others

While LOAM-derived approaches offer fast online SLAM for navigation, they are not the only option. Several alternative methods to LiDAR-based SLAM exist. One recent open-source algorithm is ART-SLAM [40], which uses a modular architecture to perform SLAM. The algorithm relies on ICP approaches to match point clouds and heavy down-sampling to maintain real-time performance. It also optionally accepts GPS and IMU data and includes a loop-detection module for accurate large-scale mapping. Like LeGO-LOAM, ART-SLAM makes ground plane assumptions but also assumes a fixed sensor height above the ground. This limits it to ground vehicles. ART-SLAM is capable of running online but is more computationally expensive than the LOAM-derived approaches, requiring a GPU to perform real-time scan matching using parallelized versions of ICP such as generalized ICP [72] and voxelized GICP [73].

There is also a multitude of closed-source solutions which have demonstrated superior results. One such solution is CSIRO's Wildcat [59], which won the Hilti SLAM Challenge 2022 [38]. The Hilti SLAM challenge is a competition specifically aimed at testing SLAM algorithms on real construction environments. Each SLAM algorithm was evaluated on a set of sequences collected from real construction sites. The algorithms were provided with recordings of LiDAR and IMU data, along with camera images which many teams excluded, and were evaluated based on the absolute trajectory error (ATE) between the estimated trajectory and the mm-accurate ground truth trajectory. The ground truth trajectory was collected using an external motion capture system and the winner was chosen based on a scoring system that utilizes the ATE metric. Wildcat also reports superior performance to LIO-SAM on multiple datasets. However, the code is not publicly accessible. Another closed-source algorithm is TVL-SLAM [56], which achieves superior performance to LeGO-LOAM on the KITTI dataset [74] by using additional camera data and tightly coupling visual and LiDAR odometry.

Table 1
Dataset statistics calculated based on the ground truth poses.

Dataset	Max Pitch / Roll (°)	Mean Ang Vel (°/s)	Mean Lin Vel (m/s)	Max Ang Vel (°/s)	Max Lin Vel (m/s)
Exp04	10.968	19.970	0.606	142.194	1.886
Exp05	17.324	17.110	0.561	146.696	1.616
Exp06	58.033	38.971	0.617	263.522	2.408

5. Experiments and results

In this section, we compare three open-source LiDAR-inertial SLAM algorithms to demonstrate some of the SLAM limitations discussed in section 3 and evaluate their performance in a real construction setting. To keep the tests consistent, only LiDAR-inertial algorithms designed to work with 360 mobile LiDAR scanners and IMUs are evaluated in this section. These algorithms were primarily designed around ground-based robots. However, many of the discussed limitations are found across numerous other visual and LiDAR SLAM algorithms. Visual-SLAM algorithms were not evaluated due to the dataset only containing monocular images, which would lead to significant scale ambiguity and would not be representative of stereo systems.

In our experiments, we evaluated LeGO-LOAM, LIO-SAM, and ART-SLAM on the Hilti Challenge Dataset 2022 [38]. Tests were conducted on Exp04, Exp05, and Exp06 sequences provided by Hilti. These three sequences are released as additional sequences and were not used for the competition. They were chosen as they provide high-frequency millimeter-accurate ground truth poses for the LiDAR sensor. The sequences were captured from a handheld unit and include relatively fast motions and rotations (**Table 1**). Exp04 and Exp05 each capture one loop of a single floor of a construction site in Schaan, Liechtenstein. The sequences are from different floors, but the building layout is very similar. Exp06 captures one loop of another floor with a similar layout, but includes much faster speeds and rotations, as well as close encounters with walls and corners. The construction site includes bare concrete walls and narrow corridors.

The sequences are of a live construction environment and include elevation changes, tilting in both pitch and roll, and both open and confined spaces. To provide compatibility with all 3 algorithms, the Hesai PandarXT-32 LiDAR points were translated into the Velodyne point format (preserving all information). The IMU data was also processed using the open-source Madgwick IMU filter [75,76], to generate IMU orientation estimates from the provided raw accelerations and

Table 2
Translational and rotational errors on the Hilti SLAM Challenge dataset. *Indicates loss of tracking occurred.

Algorithm	Exp04 Error				Exp05 Error				Exp06 Error			
	Translation (m)		Rotation (°)		Translation (m)		Rotation (°)		Translation (m)		Rotation (°)	
	RMSE	STD	RMSE	STD	RMSE	STD	RMSE	STD	RMSE	STD	RMSE	STD
Lego-LOAM (IMU)	3.713*	2.664*	58.7*	49.0*	0.982	0.667	7.1	3.3	9.683*	5.618*	102.1*	45.0*
Lego-LOAM (no IMU)	2.550	1.211	19.4	7.1	5.199*	3.404*	96.2*	59.0*	14.417*	6.184*	110.9*	43.7*
LIO-SAM	**0.167**	**0.088**	**1.5**	**0.4**	**0.095**	**0.045**	**0.9**	**0.9**	**0.311**	**0.196**	**1.9**	**0.8**
ART-SLAM odom	2.755	1.416	16.0	7.6	1.210	0.673	7.8	3.7	19.611*	8.375*	101.0*	32.5*
ART-SLAM final	1.032	0.597	6.5	3.2	1.120	0.586	11.6	7.2	26.413*	11.109*	134.1*	43.8*

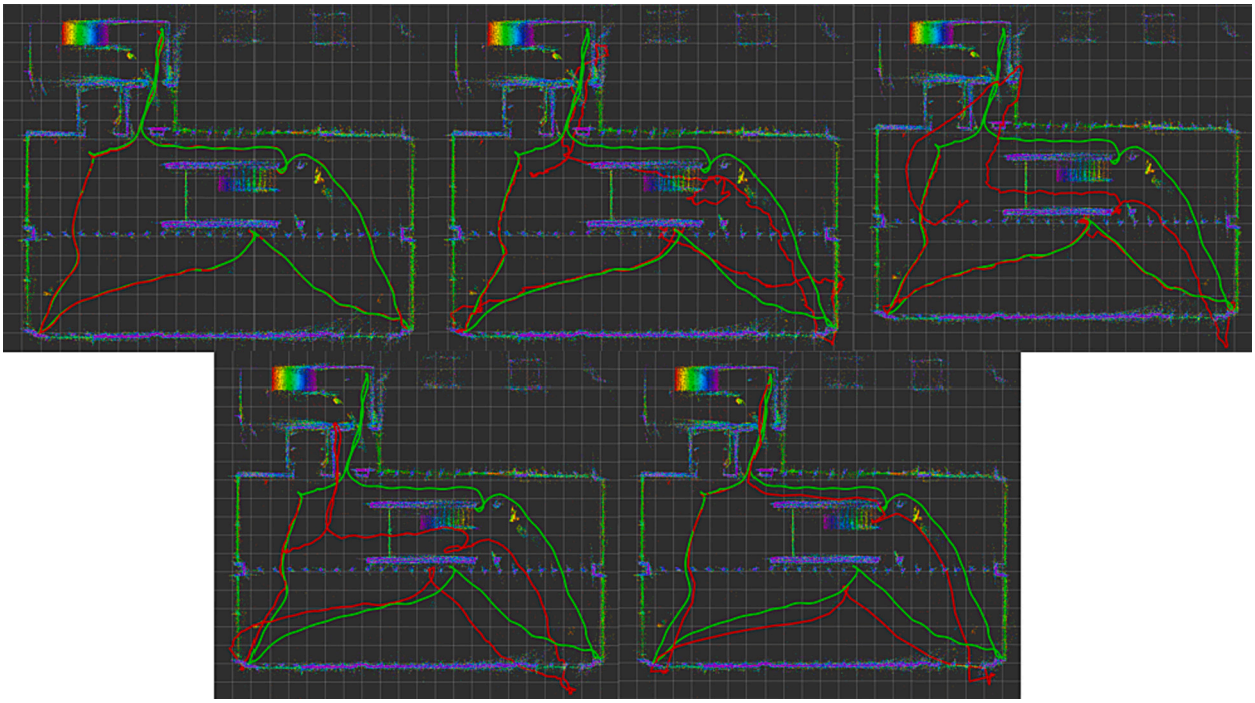


Fig. 5. Visualized ground truth (green) and estimated (red) trajectories for Exp04 (point cloud generated from ground truth poses for visualization); top left: LIO-SAM, top middle: LeGO-LOAM, top right: LeGO-LOAM + IMU, bottom left: ART-SLAM odom, bottom right: ART-SLAM final. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

angular velocities. Errors were calculated relative to the provided mm-accurate ground truth trajectories, which were generated using a motion capture system. For evaluation, translation errors were calculated using the Euclidian distance between the estimated and ground truth trajectories. The rotation errors were calculated as the smallest angle between the estimated and ground truth orientations. The root-mean-square (RMSE) and the standard deviation (STD) of the translational and rotational errors are reported to evaluate the accuracy and consistency of the estimated trajectories. Algorithm parameters were mostly kept at their default values, though sensor parameters such as IMU noise and LiDAR-dependent parameters were updated based on the device specification sheets. The distance between keyframes was tuned for ART-SLAM as the algorithm was very sensitive to this parameter. Relative transformations of the sensors were also provided to the algorithms based on the calibration files.

The evaluation results are shown in Table 2 and visualized for Exp04 in Fig. 5. LeGO-LOAM was tested both with and without IMU inputs. As ART-SLAM only provides global optimization over keyframes and does not fuse local real-time odometry with the globally optimized trajectory, both the incrementally predicted trajectory (ART-SLAM odom) and the final predicted trajectory with loop closures (ART-SLAM final) were included for reference. While ART-SLAM claims to accept IMU input, the open-source implementation quickly loses tracking on all tested sequences if IMU data is included. Therefore, IMU data was not utilized for ART-SLAM.

Table 3
The average nearest neighbor distance between the accumulated point clouds for each algorithm and the ground truth generated reference point cloud.

Avg Nearest Neighbor Distance (m)	Exp04 Error	Exp05 Error	Exp06 Error
Lego-LOAM (IMU)	0.8620	0.3092	3.0958
Lego-LOAM (no IMU)	1.3521	3.0702	4.9838
LIO-SAM	0.0667	0.0584	0.0922
ART-SLAM odom	0.6300	0.2924	1.8556
ART-SLAM final	0.6251	1.2258	2.9432

In addition to comparing the predicted trajectories, point clouds were accumulated using the LiDAR poses, representing the maps generated by each SLAM algorithm. Ground truth poses were then used to generate reference point clouds for each of the sequences. The two point clouds were then compared using the average nearest neighbor distance. The results are shown in Table 3 and are an indication of the map quality of the generate point clouds. As the generation process for the reference point cloud does not use deskewing, the unprocessed cloud was used for LIO-SAM. In practical scenarios, the deskewed point cloud would probably be employed, offering potentially greater accuracy in

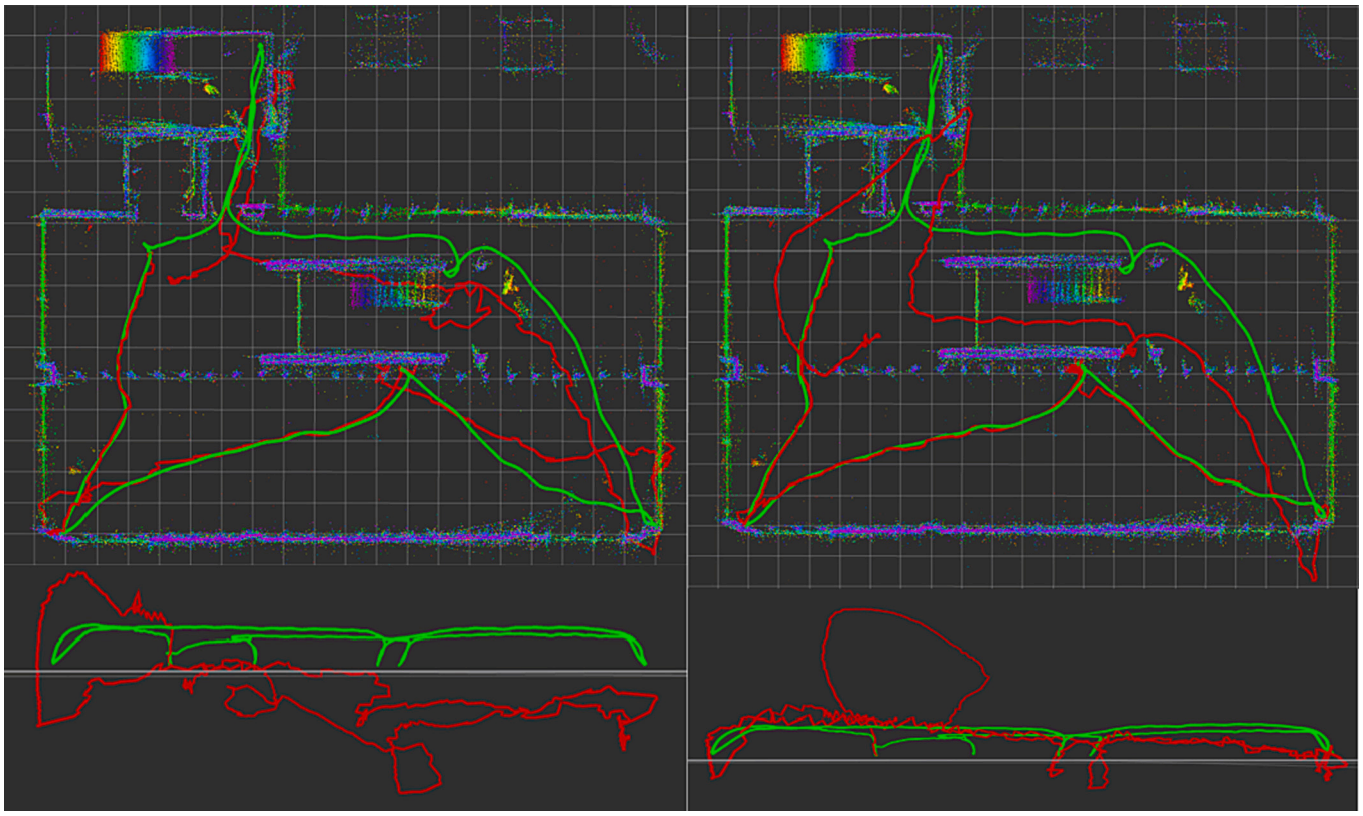


Fig. 6. Visualization of LeGO-LOAM's estimated trajectory without an IMU (left) and with an IMU (right). The red line is the estimated trajectory, and the green line is the ground truth. The trajectories are visualized from two different viewpoints. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

representing real-world geometry. Nevertheless, the deskewing process introduces corrections that aren't reflected in the reference cloud, and thus, it wasn't utilized to ensure consistency.

LIO-SAM performed the best out of the tested methods in terms of both the predicted trajectory and the accuracy of the generated map and was the only configuration that maintained tracking throughout all 3 sequences. Exp06 was particularly difficult for the other SLAM approaches as it included very fast rotations and more extreme roll and pitch angles, demonstrating how fast motions can negatively impact SLAM algorithms. This made localization difficult without an IMU to provide orientation estimates. While both LIO-SAM and LeGO-LOAM (imu) utilized IMU readings, LIO-SAM performed significantly better as LeGO-LOAM does not estimate IMU bias or directly consider integrated IMU odometry during map optimization. In fact, adding the IMU in Exp04 caused LeGO-LOAM to diverge further from the ground truth trajectory, as the IMU biases caused the algorithm to incorrectly constrain the z-height and led to a loss of tracking midway through the sequence (Fig. 6). Prior to loss of tracking, the IMU improved the accuracy of the trajectory estimate by constraining the roll and pitch (reflected in Table 3 results). Without IMU readings, both LeGO-LOAM and ART-SLAM struggled to maintain tracking through fast motions and lost tracking when they encountered confined spaces. Note that while ART-SLAM supports the use of an IMU, the algorithm lost tracking in all of our tests with IMU readings (omitted from Table 2 and 3). This demonstrates how poor integration of additional sensor data can negatively impact performance. When LeGO-LOAM and ART-SLAM managed to maintain tracking, they often suffered from drift in the pitch and roll axes, leading to significant translational errors in the z-direction.

ART-SLAM particularly struggled in this environment due to the elevation changes. As part of the floor detection module, ART-SLAM attempts to find the floor by segmenting the cloud based on static z-height thresholds. Detecting the floor helps the algorithm constrain the roll, pitch and z-height of the estimated pose. However, all three of the sequences contained significant changes in the sensor's z-height (> 1 m), which caused the floor to sometimes fall outside of the thresholds, leading to it being undetected and causing additional drift (Fig. 7). Increasing the z-height thresholds does not address the issue as this results in too many wall points being kept and leads to incorrect floor estimation. Additionally, ART-SLAM's ICP-based scan matching sometimes returned erroneous alignments, leading to additional yaw errors.

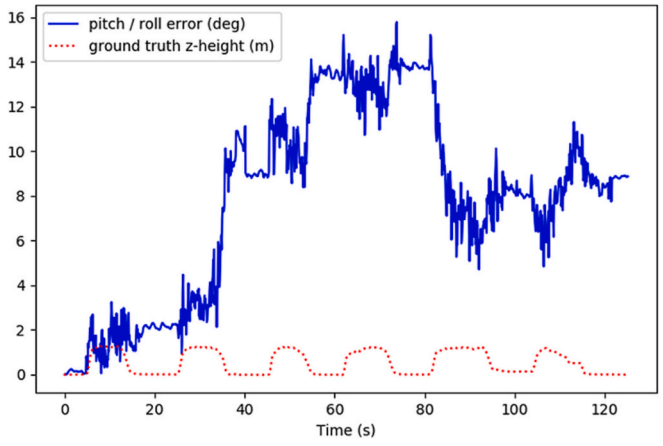


Fig. 7. Combined pitch and roll errors for ART-SLAM odom on Exp04. The large spike in error around 37 s aligns with a significant change in the sensor's z-height.

To demonstrate how running sensors at higher frequencies can help mitigate the effects of fast motions, an additional experiment was conducted using only Exp06 and LIO-SAM. The Exp06 sequence was

Table 4
Translation and rotation errors for LIO-SAM on sequence Exp 06 using different LiDAR scan frequencies. *Indicates loss of tracking occurred.

	10 Hz	5 Hz	2 Hz
Translation RMSE (m)	0.311	0.395	57.075*
Translation STD (m)	0.196	0.245	54.474*
Rotation RMSE (°)	1.92	2.14	35.32*
Rotation STD (°)	0.76	0.71	30.87*

replayed with LiDAR scans provided at lower frequencies (at 2 and 5 Hz). This was achieved by skipping over some of the LiDAR data. The results shown in Table 4 demonstrate that as the sensor frequency decreases, errors in the predicted pose increase, leading to the eventual loss of tracking. As discussed in Section 3, higher sensor frequencies lead to greater overlap between frames, improving pose estimation and resiliency to fast motions.

The configurations were also tested on the Exp14 sequence, which was labeled as medium difficulty, but all methods lost track of the robot's pose within the first 5 s and could not recover due to starting in a confined space, highlighting the need for better open-source SLAM algorithms for construction environments.

6. Conclusions

This paper provides a comprehensive overview of the challenges and approaches associated with localizing autonomous robotic systems on construction sites. The significance of simultaneous localization and mapping (SLAM) in this context is emphasized, and recent developments in SLAM algorithms are presented. As mobile robots become ubiquitous in construction environments, it becomes important for operators of these systems to understand the limitations of SLAM technologies and the sensors they rely on. Understanding these limitations can help users of mobile robots plan safe trajectories and anticipate hazardous situations due to a lack of tracking.

For the developers of autonomous robotic systems in construction sites, understanding the strengths of various SLAM approaches can help them choose the right sensors and algorithms for their use case. While it is tempting to declare a single best SLAM solution for all construction tasks, the best choice of algorithm heavily depends on the constraints of the tasks, the environment of the operation, and the engineering constraints of the robot platform. Additionally, new research developments, such as SLAM techniques based on learned convolutional features and novel SLAM architectures, continue to push the boundaries of the state-of-the-art. Thus, gaining a basic understanding of SLAM and the advantages of various SLAM algorithms is critical for choosing the best solution.

This overview of SLAM, its limitations, and common algorithms provides a high-level synopsis for operators and robot system developers working in construction and other unstructured environments. The experiments highlighted the limitations of current state-of-the-art open-source SLAM algorithms and provided insight into their failure cases. By addressing the specific considerations related to SLAM deployment in construction environments, this research contributes to the advancement of autonomous robotic systems in the construction industry. Future work will address these limitations by developing a novel LiDAR SLAM algorithm, including removing the ground plane assumptions and developing methods for efficiently extracting more meaningful point cloud features.

CRediT authorship contribution statement

Andrew Yarovoi: Data curation, Formal analysis, Investigation, Methodology, Software, Validation, Visualization, Writing – original draft. **Yong Kwon Cho:** Conceptualization, Funding acquisition, Investigation, Methodology, Project administration, Resources, Supervision, Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

Acknowledgements

This material is based upon work supported by the National Science Foundation under Grant No. 2222723. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

References

[1] S. Kohlbrecher, O. Von Stryk, J. Meyer, U. Klingauf, A flexible and scalable SLAM system with full 3D motion estimation, in: 9th IEEE International Symposium on Safety, Security, and Rescue Robotics, SSRR 2011, 2011, pp. 155–160, https://doi.org/10.1109/SSRR.2011.6106777.
[2] Q. Wang, M.K. Kim, Applications of 3D point cloud data in the construction industry: a fifteen-year review from 2004 to 2018, Adv. Eng. Inform. 39 (Jan. 2019) 306–319, https://doi.org/10.1016/J.AEI.2019.02.007.
[3] J.J. Thrun Sebastian, Leonard, Simultaneous localization and mapping, in: O. Siciliano Bruno, Khatib (Eds.), Springer Handbook of Robotics, Springer Berlin Heidelberg, Berlin, Heidelberg, 2008, pp. 871–889, https://doi.org/10.1007/978-3-540-30301-5_38.
[4] W. Hess, D. Kohler, H. Rapp, D. Andor, Real-time loop closure in 2D LIDAR SLAM, in: Proceedings - IEEE International Conference on Robotics and Automation, Institute of Electrical and Electronics Engineers Inc., Jun. 2016, pp. 1271–1278, https://doi.org/10.1109/ICRA.2016.7487258.
[5] Y. Zhao, P.A. Vela, Good feature selection for least squares pose optimization in VO/VSLAM, in: 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2018, pp. 1183–1189, https://doi.org/10.1109/IROS.2018.8593641.
[6] M.W.M. Gamini Dissanayake, P. Newman, S. Clark, H.F. Durrant-Whyte, M. Csorba, A solution to the simultaneous localization and map building (SLAM) problem, IEEE Trans. Robot. Autom. 17 (3) (Jun. 2001) 229–241, https://doi.org/10.1109/70.938381.
[7] T. Shan, B. Englot, LeGO-LOAM: lightweight and ground-optimized lidar odometry and mapping on variable terrain, in: 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), IEEE, Oct. 2018, pp. 4758–4765, https://doi.org/10.1109/IROS.2018.8594299.
[8] J. Zhang, S. Singh, LOAM: lidar odometry and mapping in real-time, in: Robotics: Science and Systems X, Robotics: Science and Systems Foundation, Jul. 2014, pp. 1–9, https://doi.org/10.15607/RSS.2014.X.007.
[9] T. Shan, B. Englot, D. Meyers, W. Wang, C. Ratti, D. Rus, LIO-SAM: tightly-coupled lidar inertial odometry via smoothing and mapping, in: 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), IEEE, Oct. 2020, pp. 5135–5142, https://doi.org/10.1109/IROS45743.2020.9341176.
[10] F. Huang, W. Wen, J. Zhang, L.T. Hsu, Point wise or feature wise? A benchmark comparison of publicly available lidar odometry algorithms in urban canyons, IEEE Intell. Transp. Syst. Mag. 14 (6) (2022) 155–173, https://doi.org/10.1109/MITS.2021.3092731.
[11] P. Moutarlier and R. Chatila, "An experimental system for incremental environment modelling by an autonomous mobile robot," Experimental Robotics I. Springer-Verlag, pp. 327–346. doi: https://doi.org/10.1007/bfb0042528.
[12] H.R. Künsch, Particle filters, Bernoulli 19 (4) (2013) 1391–1403, https://doi.org/10.3150/12-BEJSP07.
[13] R.E. Kalman, A new approach to linear filtering and prediction problems, J. Basic Eng. 82 (1) (Mar. 1960) 35–45, https://doi.org/10.1115/1.3662552.
[14] J.J. Moré, The Levenberg-Marquardt algorithm: Implementation and theory, in: G. A. Watson (Ed.), Numerical Analysis, Springer Berlin Heidelberg, Berlin, Heidelberg, 1978, pp. 105–116, https://doi.org/10.1007/BFb0067700.
[15] L. Armesto, G. Ippoliti, S. Longhi, J. Tornero, FastSLAM 2.0: least-squares approach, in: IEEE International Conference on Intelligent Robots and Systems, 2006, pp. 5013–5018, https://doi.org/10.1109/IROS.2006.282528.
[16] D. Fox, S. Thrun, W. Burgard, F. Dellaert, Particle filters for mobile robot localization, in: Sequential Monte Carlo Methods in Practice, N. and G. N. Doucet Arnaud and de Freitas, Ed, Springer New York, New York, NY, 2001, pp. 401–428, https://doi.org/10.1007/978-1-4757-3437-9_19.

[17] G.A. Terejanu, Extended Kalman Filter Tutorial, University at Buffalo, 2008. Accessed: Jun. 09, 2023. [Online]. Available: http://homes.cs.washington.edu/~todorov/courses/cseP590/readings/tutorialEKF.pdf.
[18] S.J. Julier, J.K. Uhlmann, New extension of the Kalman filter to nonlinear systems, in: I. Kadar (Ed.), Signal Processing, Sensor Fusion, and Target Recognition VI, SPIE, Jul. 1997, p. 182, https://doi.org/10.1117/12.280797.
[19] L.M. Paz, P. Jensfelt, J.D. Tardos, J. Neira, EKF SLAM updates in O(n) with divide and conquer SLAM, in: Proceedings 2007 IEEE International Conference on Robotics and Automation, 2007, pp. 1657–1663, https://doi.org/10.1109/ROBOT.2007.363561.
[20] F. Dellaert, M. Kaess, et al., Factor graphs for robot perception, Foundations and Trends® in Robotics 6 (1–2) (2017) 1–139. Accessed: Jun. 05, 2023. [Online]. Available: http://www.cs.cmu.edu/~kaess/pub/Dellaert17fnt.pdf.
[21] P. Agarwal, G.D. Tipaldi, L. Spinello, C. Stachniss, W. Burgard, Robust map optimization using dynamic covariance scaling, in: 2013 IEEE International Conference on Robotics and Automation, 2013, pp. 62–69, https://doi.org/10.1109/ICRA.2013.6630557.
[22] M. Kaess, H. Johannsson, R. Roberts, V. Ila, J. Leonard, F. Dellaert, iSAM2: incremental smoothing and mapping with fluid relinearization and incremental variable reordering, in: 2011 IEEE International Conference on Robotics and Automation, 2011, pp. 3281–3288, https://doi.org/10.1109/ICRA.2011.5979641.
[23] R. Kümmerle, G. Grisetti, H. Strasdat, K. Konolige, W. Burgard, G2o: A general framework for graph optimization, in: Proceedings - IEEE International Conference on Robotics and Automation, 2011, pp. 3607–3613, https://doi.org/10.1109/ICRA.2011.5979949.
[24] F. Dellaert, G. Contributors, borglab/gtsam, Georgia Tech Borg Lab, May 2022, https://doi.org/10.5281/zenodo.5794541.
[25] L. Xu, C. Feng, V.R. Kamat, C.C. Menassa, A scene-adaptive descriptor for visual SLAM-based locating applications in built environments, Automation in Construction 112 (Apr. 2020) 103067, https://doi.org/10.1016/j.autcon.2019.103067.
[26] P.-Y. Tseng, J.J. Lin, Y.-C. Chan, A.Y. Chen, Real-time indoor localization with visual SLAM for in-building emergency response, Automation in Construction 140 (Aug. 2022) 104319, https://doi.org/10.1016/j.autcon.2022.104319.
[27] L. Xu, C. Feng, V.R. Kamat, C.C. Menassa, An Occupancy Grid Mapping enhanced visual SLAM for real-time locating applications in indoor GPS-denied environments, Automation in Construction 104 (Aug. 2019) 230–245, https://doi.org/10.1016/j.autcon.2019.04.011.
[28] T. Lu, et al., A novel methodology for the path alignment of visual SLAM in indoor construction inspection, Automation in Construction 127 (Jul. 2021) 103723, https://doi.org/10.1016/j.autcon.2021.103723.
[29] S. Zhang, L. Zheng, W. Tao, Survey and evaluation of RGB-D SLAM, IEEE Access 9 (2021) 21367–21387, https://doi.org/10.1109/ACCESS.2021.3053188.
[30] C. Jing, J. Potgieter, F. Noble, R. Wang, A comparison and analysis of RGB-D cameras' depth performance for robotics application, in: 2017 24th International Conference on Mechatronics and Machine Vision in Practice (M2VIP), 2017, pp. 1–6, https://doi.org/10.1109/M2VIP.2017.8211432.
[31] H. Yu, Q. Fu, Z. Yang, L. Tan, W. Sun, M. Sun, Robust robot pose estimation for challenging scenes with an RGB-D camera, IEEE Sensors J. 19 (6) (2019) 2217–2229, https://doi.org/10.1109/JSEN.2018.2884321.
[32] M. Leingartner, J. Maurer, G. Steinbauer, A. Ferrein, Evaluation of sensors and mapping approaches for disasters in tunnels, in: 2013 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR), 2013, pp. 1–7, https://doi.org/10.1109/SSRR.2013.6719349.
[33] J. Lambert, et al., Performance analysis of 10 models of 3D LiDARs for automated driving, IEEE Access 8 (2020) 131699–131722, https://doi.org/10.1109/ACCESS.2020.3009680.
[34] Z. Zou, H. Lang, Y. Lou, J. Lu, Plane-based global registration for pavement 3D reconstruction using hybrid solid-state LiDAR point cloud, Autom. Constr. 152 (Aug. 2023) 104907, https://doi.org/10.1016/j.autcon.2023.104907.
[35] X. Yuan, et al., Automatic evaluation of rebar spacing and quality using LiDAR data: field application for bridge structural assessment, Automation in Construction 146 (Feb. 2023) 104708, https://doi.org/10.1016/j.autcon.2022.104708.
[36] C. Wang, Y.K. Cho, C. Kim, Automatic BIM component extraction from point clouds of existing buildings for sustainability applications, Automation in Construction 56 (Aug. 2015) 1–13, https://doi.org/10.1016/J.AUTCON.2015.04.001.
[37] B. Wang, Q. Wang, J.C.P. Cheng, C. Song, C. Yin, Vision-assisted BIM reconstruction from 3D LiDAR point clouds for MEP scenes, Automation in Construction 133 (Jan. 2022) 103997, https://doi.org/10.1016/J.AUTCON.2021.103997.
[38] L. Zhang, et al., Hilti-Oxford dataset: a millimeter-accurate benchmark for simultaneous localization and mapping, IEEE Robotics and Automation Letters 8 (1) (Jan. 2023) 408–415, https://doi.org/10.1109/LRA.2022.3226077.
[39] K. Chen, R. Nemiroff, B.T. Lopez, Direct LiDAR-inertial odometry: lightweight LIO with continuous-time motion correction, in: 2023 IEEE International Conference on Robotics and Automation (ICRA), 2023, pp. 3983–3989, https://doi.org/10.1109/ICRA48891.2023.10160508.
[40] M. Frosi, M. Matteucci, ART-SLAM: Accurate Real-Time 6DoF LiDAR SLAM, IEEE Robotics and Automation Letters 7 (2) (Apr. 2022) 2692–2699, https://doi.org/10.1109/LRA.2022.3144795.
[41] J. Zhang, Y. Yao, B. Deng, Fast and robust iterative closest point, IEEE Trans. Pattern Anal. Mach. Intell. 44 (7) (Jul. 2022) 3450–3466, https://doi.org/10.1109/TPAMI.2021.3054619.
[42] C. Forster, L. Carlone, F. Dellaert, D. Scaramuzza, IMU preintegration on Manifold for Efficient Visual-Inertial Maximum-a-Posteriori Estimation, 2015. Accessed: Jun. 05, 2023. [Online]. Available: https://github.com/borglab/gtsam.
[43] A. Keitaanniemi, P. Rönnholm, A. Kukko, M.T. Vaaja, Drift analysis and sectional post-processing of indoor simultaneous localization and mapping (SLAM)-based laser scanning data, Automation in Construction 147 (Mar. 2023), https://doi.org/10.1016/j.autcon.2022.104700.
[44] F. Dellaert, Factor Graphs and GTSAM: A Hands-on Introduction 2, Georgia Institute of Technology, Tech. Rep, 2012, p. 4. Accessed: Jan. 03, 2024. [Online]. Available: https://www.cc.gatech.edu/fac/Frank.Dellaert/FrankDellaert/Frank_Dellaert/Entries/2013/5/10_Factor_Graphs_Tutorial_files/gtsam.pdf.
[45] K.A. Tsintotas, L. Bampis, A. Gasteratos, The revisiting problem in simultaneous localization and mapping: a survey on visual loop closure detection, IEEE Trans Intell Transp Syst 23 (11) (2022) 19929–19953, https://doi.org/10.1109/TITS.2022.3175656.
[46] K.L. Ho, P. Newman, Detecting loop closure with scene sequences, Int. J. Comput. Vis. 74 (3) (2007) 261–286, https://doi.org/10.1007/s11263-006-0020-1.
[47] A. Schischmanow, D. Dahlke, D. Baumbach, I. Ernst, M. Linkiewicz, Seamless navigation, 3D reconstruction, thermographic and semantic mapping for building inspection, Sensors 22 (13) (2022), https://doi.org/10.3390/s22134745.
[48] K. Ebadi, M. Palieri, S. Wood, C. Padgett, A. Agha-mohammadi, DARE-SLAM: degeneracy-aware and resilient loop closing in perceptually-degraded environments, J. Intell. Robot. Syst. 102 (1) (May 2021) 2, https://doi.org/10.1007/s10846-021-01362-w.
[49] Z. Guo, Q. Yu, R. Guo, H. Lu, Structural features based visual odometry for indoor textureless environments, Proceedings - 2020 Chinese Automation Congress, CAC 2020 (Nov. 2020) 3984–3989, https://doi.org/10.1109/CAC51589.2020.9327452.
[50] X. Chen, Y. Yu, HLE-SLAM: SLAM for overexposed construction environment, in: ISARC. Proceedings of the International Symposium on Automation and Robotics in Construction, 2023, pp. 585–588, https://doi.org/10.22260/ISARC2023/0078.
[51] B. Williams, G. Klein, I. Reid, Real-Time SLAM relocalisation, in: 2007 IEEE 11th International Conference on Computer Vision, IEEE, 2007, pp. 1–8, https://doi.org/10.1109/ICCV.2007.4409115.
[52] R. Mur-Artal, J.M.M. Montiel, J.D. Tardos, ORB-SLAM: a versatile and accurate monocular SLAM system, IEEE Trans. Robot. 31 (5) (Oct. 2015) 1147–1163, https://doi.org/10.1109/TRO.2015.2463671.
[53] Z. Zhan, W. Jian, Y. Li, Y. Yue, A SLAM map restoration algorithm based on submaps and an undirected connected graph, IEEE Access 9 (2021) 12657–12674, https://doi.org/10.1109/ACCESS.2021.3049864.
[54] R. Mur-Artal, J.D. Tardos, ORB-SLAM2: an open-source SLAM system for monocular, stereo, and RGB-D cameras, IEEE Transactions on Robotics 33 (5) (Oct. 2017) 1255–1262, https://doi.org/10.1109/TRO.2017.2705103.
[55] J. Zhang, S. Singh, Visual-lidar odometry and mapping: low-drift, robust, and fast, in: 2015 IEEE International Conference on Robotics and Automation (ICRA), IEEE, May 2015, pp. 2174–2181, https://doi.org/10.1109/ICRA.2015.7139486.
[56] C.-C. Chou, C.-F. Chou, Efficient and accurate tightly-coupled visual-lidar SLAM, IEEE Trans Intell Transp Syst 23 (9) (2022) 14509–14523, https://doi.org/10.1109/TITS.2021.3130089.
[57] F. Ye, F. Shi, Y. Lai, X. Zhou, K. Li, Heading angle estimation using rotating magnetometer for mobile robots under environmental magnetic disturbances, Intell. Serv. Robot. 13 (4) (2020) 459–477, https://doi.org/10.1007/s11370-020-00334-7.
[58] A. Broggi, P. Grisleri, P. Zani, Sensors technologies for intelligent vehicles perception systems: A comparison between vision and 3D-LIDAR, in: 16th International IEEE Conference on Intelligent Transportation Systems (ITSC 2013), 2013, pp. 887–892, https://doi.org/10.1109/ITSC.2013.6728344.
[59] M. Ramezani, et al., Wildcat: Online Continuous-Time 3D Lidar-Inertial SLAM, May 2022, https://doi.org/10.48550/arXiv.2205.12595.
[60] L.F. Castanheiro, A.M.G. Tommaselli, M.V. Machado, G.H. Santos, I.S. Norberto, T. T. Reis, The use of a wide FOV laser scanning system and a slam algorithm for mobile applications, Int. Arch. Photogramm. Remote. Sens. Spat. Inf. Sci. XLIII-B1-2022 (May 2022) 181–187, https://doi.org/10.5194/isprs-archives-XLIII-B1-2022-181-2022.
[61] Y. Li, R. Bu, M. Sun, W. Wu, X. Di, B. Chen, PointCNN: Convolution on X-transformed points, in: S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, R. Garnett (Eds.), Advances in Neural Information Processing Systems, Curran Associates, Inc, 2018. Accessed: Jun. 05, 2023. [Online]. Available: https://proceedings.neurips.cc/paper/2018/file/f5f8590cd58a54e94377e6ae2eded4d9-Paper.pdf.
[62] Y. Li, L. Ma, Z. Zhong, D. Cao, J. Li, TGNet: geometric graph CNN on 3-D point cloud segmentation, IEEE Trans. Geosci. Remote Sens. 58 (5) (2020) 3588–3600, https://doi.org/10.1109/TGRS.2019.2958517.
[63] C.R. Qi, L. Yi, H. Su, L.J. Guibas, PointNet++: Deep hierarchical feature learning on point sets in a metric space, in: I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, R. Garnett (Eds.), Advances in Neural Information Processing Systems, Curran Associates, Inc, 2017 [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2017/file/d8bf84be3800d12f74d8b05e9b89836f-Paper.pdf.
[64] Y. Wang, Y. Sun, Z. Liu, S.E. Sarma, M.M. Bronstein, J.M. Solomon, Dynamic graph CNN for learning on point clouds, ACM Trans. Graph. 38 (5) (Oct. 2019) 1–12, https://doi.org/10.1145/3326362.
[65] J. Li, X. Zhang, J. Li, Y. Liu, J. Wang, Building and optimization of 3D semantic map based on Lidar and camera fusion, Neurocomputing 409 (2020) 394–407, https://doi.org/10.1016/j.neucom.2020.06.004.

[66] C. Campos, R. Elvira, J.J.G. Rodriguez, J.M.M. Montiel, J.D. Tardos, ORB-SLAM3: an accurate open-source library for visual, visual–inertial, and multimap SLAM, IEEE Transactions on Robotics 37 (6) (Dec. 2021) 1874–1890, https://doi.org/10.1109/TRO.2021.3075644.
[67] J. Engel, V. Koltun, D. Cremers, Direct sparse odometry, IEEE Trans. Pattern Anal. Mach. Intell. 40 (3) (Mar. 2018) 611–625, https://doi.org/10.1109/TPAMI.2017.2658577.
[68] X. Gao, R. Wang, N. Demmel, D. Cremers, LDSO: direct sparse odometry with loop closure, in: 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2018, pp. 2198–2204, https://doi.org/10.1109/IROS.2018.8593376.
[69] A. Tourani, H. Bavle, J.L. Sanchez-Lopez, H. Voos, Visual SLAM: what are the current trends and what to expect? Sensors 22 (23) (Nov. 2022) 9297, https://doi.org/10.3390/s22239297.
[70] J. Lin, F. Zhang, Loam livox: A fast, robust, high-precision LiDAR odometry and mapping package for LiDARs of small FoV, in: 2020 IEEE International Conference on Robotics and Automation (ICRA), IEEE, May 2020, pp. 3126–3131, https://doi.org/10.1109/ICRA40945.2020.9197440.
[71] P.J. Besl, N.D. McKay, A method for registration of 3-D shapes, IEEE Trans. Pattern Anal. Mach. Intell. 14 (2) (1992) 239–256, https://doi.org/10.1109/34.121791.
[72] A. Segal, D. Haehnel, S. Thrun, Generalized-icp, in: Robotics: Science and Systems, 2009, p. 435, https://doi.org/10.15607/RSS.2009.V.021.
[73] K. Koide, M. Yokozuka, S. Oishi, A. Banno, Voxelized GICP for fast and accurate 3D point cloud registration, in: Proceedings - IEEE International Conference on Robotics and Automation, vol. 2021-May, 2021, pp. 11054–11059, https://doi.org/10.1109/ICRA48506.2021.9560835.
[74] A. Geiger, P. Lenz, R. Urtasun, Are we ready for autonomous driving? The KITTI vision benchmark suite, in: 2012 IEEE Conference on Computer Vision and Pattern Recognition, 2012, pp. 3354–3361, https://doi.org/10.1109/CVPR.2012.6248074.
[75] I. Dryanovski, M. Günther, imu_filter_madgwick, Aug. 19, 2022. Accessed: Jun. 10, 2023. [Online]. Available: http://wiki.ros.org/imu_filter_madgwick.
[76] S.O.H. Madgwick, A.J.L. Harrison, R. Vaidyanathan, Estimation of IMU and MARG Orientation using a gradient descent algorithm, in: 2011 IEEE International Conference on Rehabilitation Robotics, IEEE, Jun. 2011, pp. 1–7, https://doi.org/10.1109/ICORR.2011.5975346.