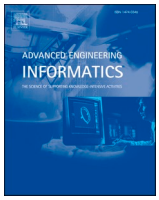
Advanced Engineering Informatics 66 (2025) 103477

Contents lists available at ScienceDirect

Advanced Engineering Informatics

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



LIO-BIM – Coupling lidar inertial odometry with building information modeling for robot localization and mapping

Jan Stührenberg *, Kay Smarsly

Institute of Digital and Autonomous Construction, Hamburg University of Technology, Germany

ARTICLE INFO

Keywords:
Simultaneous localization and mapping
Building information modeling
Quadruped robots
Construction inspection
Scan matching
Factor graphs

ABSTRACT

Mobile robots deployed to automate tasks in the construction industry require accurate robot localization and navigation. Building information modeling (BIM) is increasingly prevalent, and BIM models are interpretable by robots to help navigate in or around buildings. Most approaches towards robot localization through BIM models solely rely on maps derived from the BIM models requiring models with a high level of development (LOD) and the accurate modeling of non-structural objects. In practice, however, BIM models often employ a limited LOD and non-structural objects, if modeled, may appear in different locations, which may result in scan-BIM deviations and thus in localization errors. This paper presents the so called "LIO-BIM" framework, which couples lidar inertial odometry (LIO) and BIM for robust mobile robot localization and mapping, using 3D lidar to overcome the issues related to scan-BIM deviations. LIO-BIM builds upon simultaneous localization and mapping techniques and performs scan matching multiple times, i.e. (i) scan matching of the latest lidar scan with lidar scans previously recorded to maintain an accurate map of the environment, and (ii) scan matching of a local map around the robot with a BIM model to enable localization and mapping relative to the BIM model. The maps may be used at run-time, e.g., for construction progress monitoring or quality inspection. The framework, whose code is provided as open source, is implemented on a quadruped robot equipped with a 3D lidar, an inertial measurement unit, and a camera, and it is validated in a cluttered indoor office environment represented by a BIM model. Furthermore, the framework is validated on the ConSLAM dataset showcasing a cluttered construction site environment. As a result, the validation tests demonstrate accurate and robust 3D localization and mapping aligned with BIM models in real-time.

1. Introduction

Routine inspections of buildings help meet quality standards and reduce the impact of delays during construction, and inspections ensure safety and integrity of buildings during operation [1]. However, manual inspections may be time-consuming and hazardous processes that may result in subjective quality assessments [2]. Therefore, robots have been proposed for automated inspections of buildings, representing safer alternatives that allow for more frequent and more objective inspections, compared to manual inspections [3].

Mobile robots are used for maintenance inspections, construction quality inspection, progress monitoring, as-built/as-is modeling, and safety inspection [1]. To accomplish the inspection tasks, mobile robots record data relevant to inspection of buildings, which may include point cloud data [4], visual data [5], and vibration data [3]. All sensor data recorded by mobile robots for monitoring and inspection of buildings has in common that the location of the recording is relevant, i.e. the quality of the data recorded by robots, optimal inspection task planning, and reliable autonomous navigation rely on precise and robust localization of the robots.

In a typical pipeline for robot navigation, robots are teleoperated in the first step to explore an unknown environment, while running a simultaneous localization and mapping (SLAM) program to create a map of the environment [6]. In the next steps, the robots autonomously navigate by localizing themselves in the map, planning paths to goal poses, and following the paths while avoiding obstacles. As the initial creation of a map using SLAM can be time-consuming, current research investigates how to leverage existing building information modeling (BIM) models as maps for robot indoor localization and navigation [7–17]. Furthermore, BIM models provide robots access to valuable semantic and geometric information of buildings. By aligning robot sensor data with BIM models, the information can be used to support

* Corresponding author.
E-mail address: jan.stuehrenberg@tuhh.de (J. Stührenberg).

https://doi.org/10.1016/j.aei.2025.103477
Received 28 January 2025; Received in revised form 17 April 2025; Accepted 14 May 2025
Available online 24 May 2025

various construction-related applications.

The most critical issue when using BIM models as maps for indoor robot navigation is the spatial accuracy of the models. In practice, deviations between as-designed BIM models and the as-built reality, referred to in this paper as scan-BIM deviations, are mentioned in various publications. BIM models are often designed with a limited level of development (LOD), rarely higher than LOD 300 [18]. Non-structural objects, such as doors, appliances, and furniture, are often either not modeled or, if modeled, may appear in different locations [12]. Construction inaccuracies result in deviations from the as-planned model [19]. Scan-BIM deviations are described in [14] as "unavoidable". In this regard, the approaches either explicitly assume a spatially correct BIM model [7], report localization errors due to scan-BIM deviations [13], or try to make the localization more robust against scan-BIM deviations.

The purpose of this research is to enable robust 3D localization and mapping of mobile robots in real-time relative to BIM models, particularly in complex environments with scan-BIM deviations. Real-time localization relative to BIM models represents the basis for enabling autonomous capabilities of mobile robots in conjunction with BIM models. LIO-BIM may be used in conjunction with path planning [20] to enable indoor navigation using BIM models, and thus, to automate tasks such as building inspections [1], progress monitoring [21], and construction logistics [22].

To fulfill the purpose, this paper presents a framework for lidar inertial odometry (LIO) coupled with BIM (LIO-BIM) and combines the advantages of leveraging SLAM and BIM models. Unlike existing methods, LIO-BIM enables 3D localization in real-time and explicitly maps the as-built state using factor graph-based SLAM to account for scan-BIM deviations. Geometric and semantic information about the environment derived from BIM models is integrated to provide localization relative to the BIM models and to reduce the long-term drift common to SLAM systems. LIO-BIM relies on scan matching using an efficient feature-based iterative closest point (ICP) algorithm. Scan matching is performed multiple times (i) for the latest lidar point cloud with immediately preceding recorded lidar point clouds for LIO, and (ii) for the local map around the robot with the BIM model to provide localization relative to the BIM model. Scan matches of the local map with the BIM model with low compliance are rejected to account for errors introduced by scan-BIM deviations. On the other hand, long-term drift errors of SLAM are corrected by scan matches of the local map with the BIM models showing high compliance. The locations of the robots and the created maps aligned with the BIM models can be used in real-time for navigation, visualization, and the completion of inspection tasks. In conclusion, LIO-BIM provides a solution for complex environments with scan-BIM deviations encountered in practical applications and thus has the potential to increase the adoption of robots for construction-related applications.

The remainder of this paper is organized as follows: The background and state of the art of robot localization in BIM models is discussed in Section 2. The methodology and implementation of the LIO-BIM framework are presented in Section 3. Validation tests conducted in an indoor office environment and on a dataset of a real-world construction site are described in Section 4. The results of the validation tests are presented in Section 5 and discussed in Section 6. Section 7 concludes the paper and suggests potential future research directions.

2. Robot localization using building information models

The localization problem essentially involves estimating the position and orientation of robots with respect to a coordinate system. Localization is described as one of the most fundamental tasks in enabling autonomous capabilities for mobile robots [23]. The localization problem can be grouped into pose tracking and the global localization problem [24]. In pose tracking, the initial pose of robots is known and localization seeks to correct errors in the odometry. As for the global localization problem, robots have no initial knowledge of their pose and therefore must determine the pose from scratch.

Localization can be achieved using a variety of sensor technologies. A popular technology used for localizing robots is the global navigation satellite system (GNSS). However, GNSS-based localization encounters challenges in urban or indoor environments where signals may be blocked by buildings, causing unreliable performance. By deploying signal emitters in buildings, robots are able to localize themselves through trilateration, for example using ultra-wideband communication [25]. Similarly, visual tags deployed in buildings allow localization [26]. Both methods require the distribution of calibrated emitters or tags throughout the buildings, which renders the methods cumbersome to install and inflexible, particularly in large buildings. Therefore, onboard sensors are a viable alternative to the above methods and do not require modifications of the buildings.

For robot localization, probabilistic methods have become widely accepted [27]. Based on environment maps, a particle filter or Monte Carlo localization (MCL) may be used for global localization [24]. In MCL, the belief of robot poses is represented by sets of samples. Initially, sample sets are uniformly distributed over maps. In each iteration, control inputs are incorporated to predict the next state of each sample, and measurement updates are incorporated by matching observed landmarks to the map to calculate importance weights for each sample. Based on the importance weights, resampling is performed. After a few iterations, the probability mass of the sample sets (ideally) converges to the true robot poses.

If no maps are given, and must thus be created, the SLAM problem may be used as a means to create maps while localizing the robot in the maps [28]. There exist two main forms of the SLAM problem: the online SLAM problem and the full SLAM problem. Online SLAM attempts to recover current robot poses. For example, measurements recorded from inertial measurement units may be integrated, and scan matching may be performed on subsequent lidar measurements to compute robot poses, resulting in LIO. ICP algorithms are a popular choice for performing scan matching [29]. ICP algorithms iteratively determine correspondences between two scans and compute transformations that minimizes the distances between the correspondences. Typically, online SLAM is performed at high frequencies, and full SLAM is performed less frequently to guarantee real-time capabilities.

Full SLAM attempts to recover maps together with entire paths, rather than current poses, to reduce drift. State-of-the-art SLAM frameworks, such as "LIO-SAM" [30], build on factor graphs to solve the full SLAM problem. Factor graphs are probabilistic graphical models similar to Bayesian networks providing abstraction for solving inference problems, such as the full SLAM problem, in robotics [31]. Formally, factor graphs are bipartite graphs that have two types of nodes, variables and factors. The variables represent unknown states, which are robot poses and landmarks of the environment in the SLAM problem. Factors represent measurements that constrain the variables, known for example from LIO. To determine the variables, maximum a posteriori inference is performed on the factor graph by solving nonlinear optimization problems. To guarantee real-time capabilities of SLAM, the computational efficiency can be improved by employing sparsity. For example, sparsity may be employed by reducing the number of lidar scans in the factor graph by selecting a reduced number of so-called "keyframes" [30]. Sparsity may also be employed in scan matching, where lidar point clouds are processed to contain fewer points, but descriptive features [32].

BIM is a method that allows using standardized digital building models throughout the lifecycle of buildings, to improve the information flow and increase efficiency [33]. Throughout the lifecycle of buildings, mobile robots may advantageously be used to automate a variety of tasks to increase the operational efficiency or to improve the safety of workers. In the construction phase, mobile robots may be used to transport heavy equipment [22], to automate construction activities [34], and to monitor the construction progress [21]. In the operation phase, robots may be used for maintenance inspections and safety

inspections [1]. In the demolition phase, robots may be used to sort and recycle demolition waste [35]. The deployment of mobile robots in the aforementioned phases requires accurate and real-time localization. In this context, BIM models contain valuable geometric and semantic information to support robot localization. Therefore, BIM models need to be interpreted as maps to be utilized by robots for localization. BIM models have been leveraged as maps for robot indoor localization and navigation in [7 –17]. In addition to BIM models, three-dimensional CAD models [19,36] and two-dimensional floor plans [37 –41] have also been used. An overview of the relevant literature on lidar-based robot localization using BIM models or floor plans is given in Table 1.

Commonly, the robots are localized in a horizontal plane in 2D, assuming movement restricted to a horizontal plane. For localizing in 2D, the Adaptive MCL (AMCL) algorithm is used in several studies [7,9,10,12,41]. AMCL requires a two-dimensional binary occupancy grid map of the environment. In [15], MCL techniques are compared with graph-based SLAM methods for 2D localization in maps derived from BIM models in Gazebo. In the tests conducted, MCL techniques have performed better for global localization, but graph-based SLAM methods have shown higher accuracy in the pose tracking problem. Factor graph-based localization methods in 2D are developed in [8,11,37,38,40]. 2D localization methods are fundamentally limited, as 2D localization implicitly assumes a “flat and rectangular world” with planar vertical walls, planar horizontal floors, and/or planar horizontal ceilings [19]. The assumption is no longer valid at higher levels of accuracy or in more complex environments, and an extension to 3D is necessary.

To address the limitations of 2D localization, several studies have explored extending localization methods into 3D. In 3D localization, the increased computational demands require efficient methods, often leading to a trade-off between accuracy and speed. In [13], BIM models are converted into point clouds to run an ICP algorithm to localize a handheld lidar. The approach is extended in [14], where semantic information from the BIM model is used to label the point cloud. The semantic-aided ICP algorithm gives correct data associations higher weights. The initial position is provided manually. However, experiments conducted in [14] have shown that the localization method is outperformed by state-of-the-art SLAM systems in the horizontal plane. In [16], the generation of session data, i.e. pose graph-based maps including descriptors, from simulating a robot moving through the BIM model is introduced. In the as-built environment, the robot attempts to align with the session data by matching point cloud feature descriptors. The approach is extended in [17]. A drift reduction compared to state-of-the-art SLAM systems is reported. However, the approach is computationally expensive and the system is not running in real-time.

The LIO-BIM framework designed to overcome the above limitations is presented in the following section. LIO-BIM leverages BIM models and LIO for real-time localization in indoor and construction environments.

3. Coupling LIO with BIM

This section presents the methodology and the implementation of the LIO-BIM framework. The LIO-BIM framework is devised to integrate BIM into 3D graph-based SLAM systems. LIO-BIM accounts for scan-BIM deviations by explicitly mapping the as-built state. Efficient feature-based scan matching with BIM models is conducted, enabling real-time localization and mapping relative to BIM models and allowing for improved indoor localization by reducing long-term drift of SLAM systems. In outdoor environments without BIM information, LIO-BIM reverts to regular SLAM. LIO-BIM builds atop LIO-SAM, a state-of-the-art framework for SLAM [30], and it receives sensor data from a 3D lidar, an inertial measurement unit (IMU) with 9 degrees of freedom (DoF) as well as from a camera defined in a coordinate system referenced to the robot, the robot frame $\boldsymbol{R}$. The sensor data is used to estimate the state of the robot and its trajectory in the map frame $\boldsymbol{M}$. In addition to the robot frame $\boldsymbol{R}$ and the map frame $\boldsymbol{M}$, the BIM model frame $\boldsymbol{B}$ represents the coordinate system of the BIM model. Furthermore, the following transformations are defined:

- Transformation from the robot frame to the map frame ${}^{M}_{t}\boldsymbol{T}_{R} \in \mathrm{SE}(3)$
- Transformation from the map frame to the BIM model frame ${}^{B}\boldsymbol{T}_{M} \in \mathrm{SE}(3)$

Table 1
Relevant literature on lidar-based robot localization using BIM models or floor plans.

Ref.	Map source	2D/3D localization	Method	Validation tests	Handling of deviations	Limitations
[7]	BIM	2D	AMCL	Simulation	–	Assumes precise representation of reality, 2D
[9]	BIM	2D	AMCL	Simulation	–	Accuracy not evaluated, 2D
[10]	BIM	2D	AMCL	Simulation	–	Accuracy not evaluated, 2D
[12]	BIM	2D	Modified AMCL with object recognition and map updating	Simulation and real world	Insertion of recognized objects in map	Errors in the meter range, 2D
[41]	Floor plan	2D	Modified AMCL with object recognition and map updating	Real world	Insertion of recognized objects in map	2D
[15]	BIM	2D	Comparing MCL to SLAM-based approaches	Simulation	Pose graph-based maps	2D
[8]	BIM	2D	Feature matching using spatial-semantic database and factor graph	Real world	Factor graph	Accuracy not evaluated, 2D
[11]	BIM	2D	Graph-based SLAM	Simulation and real world	Pose graph-based maps	2D
[37]	Floor plan	2D	Graph-based SLAM with scan-to-map matching using GICP	Real world	Explicit mapping of changes	2D
[38]	Floor plan	2D	Graph-based SLAM with scan-to-map matching using GICP	Real world	Explicit mapping of changes	2D
[40]	Floor plan	2D	Graph-based SLAM with scan-to-map matching using edge features	Real world	Max-mixture error model	2D
[13]	BIM	3D	Point-to-plane ICP	Real world	–	Scan-BIM deviations cause errors
[14]	BIM	3D	Semantic ICP	Real world	Weight function in semantic ICP	Outperformed by SLAM systems in the horizontal plane
[16]	BIM	3D	Alignment with session data generated from BIM model	Simulation and real world	Explicit mapping of as-built condition	Not real-time capable
[17]	BIM	3D	Extension of [16]	Real world	Explicit mapping of as-built condition	Not real-time capable
LIO-BIM	BIM	3D	Graph-based SLAM incorporating scan matching with BIM model	Real world	Explicit mapping of as-built condition	

- Transformation from the robot frame to the BIM model frame ${}^{B}_{t}\boldsymbol{T}_{R}={}^{B}\boldsymbol{T}_{M}\ {}^{M}_{t}\boldsymbol{T}_{R}\in \mathrm{SE}(3)$

The subscript t denotes dynamic transformations. Fig. 1 shows an overview of the system structure and the features of LIO-BIM. Fig. 2 depicts a flow chart of LIO-BIM to highlight the data and program flow. Prior to runtime, edge and planar feature points are extracted from BIM models, described in Subsection 3.1. During runtime, when robots traverse buildings, LIO-BIM leverages lidar and IMU data to perform LIO as presented in [30], and the video stream of the camera is observed for fiducial tags. Upon detecting fiducial tags, the static transformation from the map frame to the BIM model frame ${}^{B}\boldsymbol{T}_{M}$ is computed, explained in Subsection 3.2. Once ${}^{B}\boldsymbol{T}_{M}$ has been obtained, scan matching of local maps around robots with the extracted BIM model features is conducted, detailed in Subsection 3.3. Scan matches of the local map with the BIM model with high compliance are used to integrate BIM factors, i.e. unary factors, that correct the trajectory and map to align with BIM models, into factor graphs, shown in Subsection 3.4. The general approach towards knowledge representation in LIO-BIM is briefly described in Subsection 3.5.

3.1. Extracting edge and planar features of BIM models

LIO-BIM performs scan matching multiple times and at high frequencies. To enable real-time capabilities, a feature-based ICP algorithm is employed, which operates on geometrical feature points extracted from lidar point clouds. Feature points are selected on sharp edges and planar surface patches by evaluating the smoothness of points over a local region. For a detailed background of the feature point extraction in lidar point clouds and of the methodology for finding feature point correspondences, the reader is referred to [32]. To employ feature-based scan matching of the local map with BIM models, geometrical feature points are extracted in the BIM models. Again, the feature points represent sharp edges and planar surfaces, allowing to match edges and planes. The approach towards scan matching with BIM models presented in this paper is runtime efficient because feature lidar point clouds are reused from the LIO process and the feature points of the BIM models are extracted prior to runtime. Furthermore, the method of extracting edge and planar features from BIM models provides a computationally more efficient solution, compared to point-to-point matching with featureless point clouds extracted from BIM models as presented in [13]. In the following, the process of extracting edge and planar features of BIM models is described.

Initially, a BIM model of interest is exported to the Industry Foundation Classes (IFC) format. LIO-BIM iterates over all geometrical elements in the IFC file, i.e. *IfcProducts*, leveraging the IfcOpenShell library [42]. Inbuilt filter functions from the IfcOpenShell are used to exclude elements that potentially deteriorate the scan matching with BIM models during runtime. Examples of such elements are windows, which reflect laser beams and are therefore not detected by lidars, or doors, which may have different opening angles. All remaining elements in the IFC file are converted into a triangulated mesh file and temporarily stored.

The mesh file represents all surfaces in the BIM model. A point cloud is generated by uniformly sampling points with sufficient density ρ_B from the triangulated meshes using the Point Cloud Library (PCL) [43]. Along with the cartesian coordinates, the surface normal of the points are saved. With the cartesian coordinates and the surface normal, the principal curvature of the sampled points can be estimated. The principal curvature is used to determine, if a point can be regarded as an edge or planar feature point. For each sampled point, the neighboring points in a small radius are searched. Based on the surrounding points, the principal curvatures k_1 and k_2, representing the minimum and maximum values of the curvature, are calculated. For extracting feature points, principal curvatures near zero indicate planar feature points, while principal curvatures far off from zero indicate edge feature points.

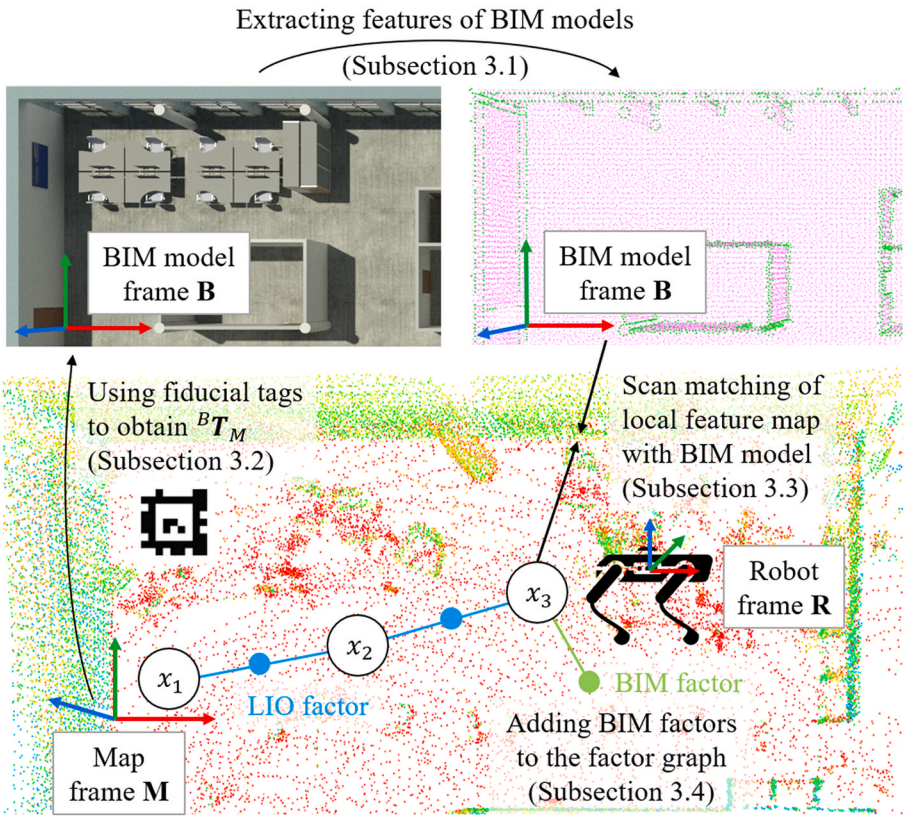


Fig. 1. Overview of the system structure of LIO-BIM.

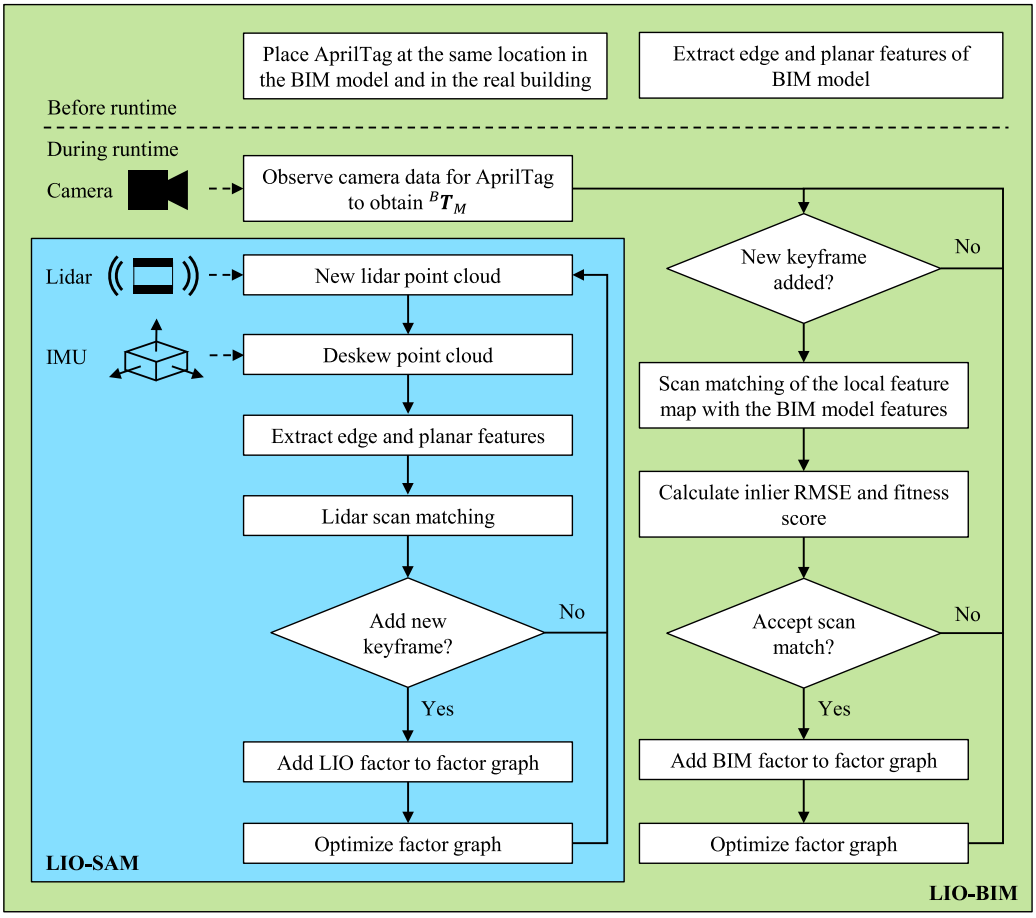


Fig. 2. Flow chart of LIO-BIM.

As the direction of the curvature is not evaluated, the principal curvatures k_1 and k_2 are combined in a curvature score k_B according to $k_B = \text{abs}(k_1) + \text{abs}(k_2)$. Points with a curvature score lower than the threshold for planar feature points $\tau_{B,p}$ are considered planar feature points, and points with a curvature score higher than the threshold for edge feature points $\tau_{B,e}$ are considered edge feature points. For efficiency in the scan matching process described in Subsection 3.3, the edge and planar feature point clouds should be sparse. To include the most prominent edges and planar features, the points are sorted according to their curvature score. Neighboring points to the most prominent points in a small radius $r_{B,f}$ are discarded and the point cloud is downsampled with a filter according to the same parameters as the lidar point cloud

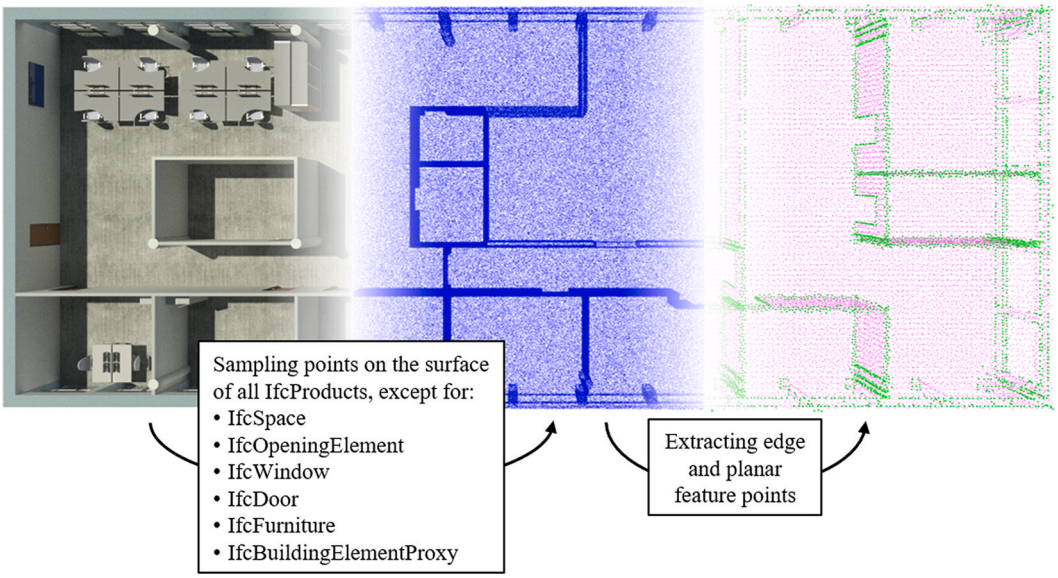


Fig. 3. Extracting edge and planar feature points of BIM models.

during run time. Finally, the sparse feature point cloud of the BIM model is stored in the Point Cloud Data (PCD) file format. Throughout this paper, the downsampled edge and planar feature point clouds of the BIM model are denoted as $\overline{\boldsymbol{P}}_B^e$ and $\overline{\boldsymbol{P}}_B^p$, respectively. The process of extracting edge and planar feature point clouds of BIM models is depicted in Fig. 3.

The extraction of edge and planar feature points is one of two prerequisites to perform the feature-based scan matching of the local feature map with the BIM model. As a second prerequisite, a transformation from the map frame to the BIM model frame ${}^{B}\boldsymbol{T}_M$ is retrieved using fiducial tags, described in the following subsection.

3.2. Using fiducial tags to obtain a transformation from map to BIM model frame

The second feature of LIO-BIM pertains to fiducial tags, artificial landmarks designed for easy recognition [44], allowing for obtaining a transformation from the map frame to the BIM model frame ${}^{B}\boldsymbol{T}_M$, which is used to calculate the starting value of the feature-based scan matching of the local map around the robot with the BIM model. Obtaining a transformation from the map frame to the BIM model frame corresponds to the global localization problem. While global localization solutions that rely on input provided by proprioceptive sensors and BIM models are desirable, challenges arise with large scan-BIM deviations, for symmetrical environments, and for environments with repeating and/or similar scenes, such as multi-story buildings with identical story layouts. Consequently, LIO-BIM utilizes fiducial tags that require minimal effort to quickly and unambiguously resolve the global localization problem. Fiducial tags have a payload that allows for unambiguous correspondences. Therefore, global localization using fiducial tags does not suffer from expensive computation in large environments or from problems in symmetrical or repeating environments. The downside of using fiducial tags pertains to limiting the starting positions of robots. The scan matching with features extracted from BIM models, as described in Subsection 3.3, can only start once a fiducial tag has been recognized. Placing fiducial tags at usual starting locations such as robot charging stations or at frequently traversed places, such as floors, can mitigate the limitation. Before a fiducial tag has been recognized, the framework employs basic LIO. In the remainder of this subsection, the acquisition of the transformation ${}^{B}\boldsymbol{T}_M$ from the map frame $\boldsymbol{M}$ to the BIM model frame $\boldsymbol{B}$ using fiducial tags is described.

LIO-BIM uses AprilTags, which are 2D barcode-style tags with a small payload, i.e. an ID number [45]. AprilTags can be printed on a piece of paper and stuck on a wall. When a camera defined in the camera frame $\boldsymbol{C}$ is pointed at a 2D AprilTag of known size, an accurate 6-DoF transformation from the tag to the camera ${}^{C}_{t}\boldsymbol{T}_A$ can be obtained. To obtain a transformation from the map frame to the BIM model frame ${}^{B}\boldsymbol{T}_M$, an AprilTag with the same ID number must be placed at the same location in the BIM model and in the real building. From a practical standpoint, regarding the placement in the BIM model, LIO-BIM provides an AprilTag family compatible with Autodesk Revit, which can be aligned with arbitrary surfaces in the BIM model. The tag is exported as an IfcBuildingElementProxy in IFC format. The calculation of the transformation ${}^{B}\boldsymbol{T}_M$ is depicted in Fig. 4. The AprilTag frame is denoted as $\boldsymbol{A}$ and the camera center frame as $\boldsymbol{C}$. When a robot equipped with a camera detects the AprilTag, multiple transformations are referenced:

- The transformation ${}^{B}\boldsymbol{T}_A$ from the AprilTag frame $\boldsymbol{A}$ to the BIM model frame $\boldsymbol{B}$ is included in the coordinates of the AprilTag in the BIM model. To obtain the coordinates, the IFC file is parsed for IfcBuildingElementProxys with object types named “AprilTag” and the matching ID number.
- Using the AprilTag visual fiducial detector [46], the transformation ${}^{C}_{t}\boldsymbol{T}_A$ from the AprilTag frame $\boldsymbol{A}$ to the camera center frame $\boldsymbol{C}$ at time t is computed.
- The transformation ${}^{R}\boldsymbol{T}_C$ from the camera center frame $\boldsymbol{C}$ to the robot frame $\boldsymbol{R}$ is static and can be measured. The transformation is provided in the robot description in a Universal Robot Description Format (URDF) file.
- The transformation ${}^{M}_{t}\boldsymbol{T}_R$ from the robot frame $\boldsymbol{R}$ to the map frame $\boldsymbol{M}$ at time t is estimated by the odometry of the SLAM algorithm, which is consistent with the original LIO-SAM implementation up to this point.

With the transformations listed above, the transformation ${}^{B}\boldsymbol{T}_M$ from the map frame $\boldsymbol{M}$ to the BIM model frame $\boldsymbol{B}$ can be calculated as follows:

$$ {}^{B}\boldsymbol{T}_M = {}^{B}\boldsymbol{T}_A \left({}^{M}_{t}\boldsymbol{T}_R \; {}^{R}\boldsymbol{T}_C \; {}^{C}_{t}\boldsymbol{T}_A \right)^{-1} \tag{1} $$

The calculation in Equation (1) is based on multiplying four transformations and is therefore prone to several errors. To account for poor quality estimates of the transformation ${}^{C}_{t}\boldsymbol{T}_A$, LIO-BIM checks the tag detections in subsequent camera images and only accepts the transformation if the robot moves slowly. Nevertheless, errors may occur due to an inaccurately supplied transformation ${}^{R}\boldsymbol{T}_C$, due to deviations between the placement of the AprilTag in the BIM model and in the real building, and due to possible drift in the SLAM odometry ${}^{M}_{t}\boldsymbol{T}_R$. However,

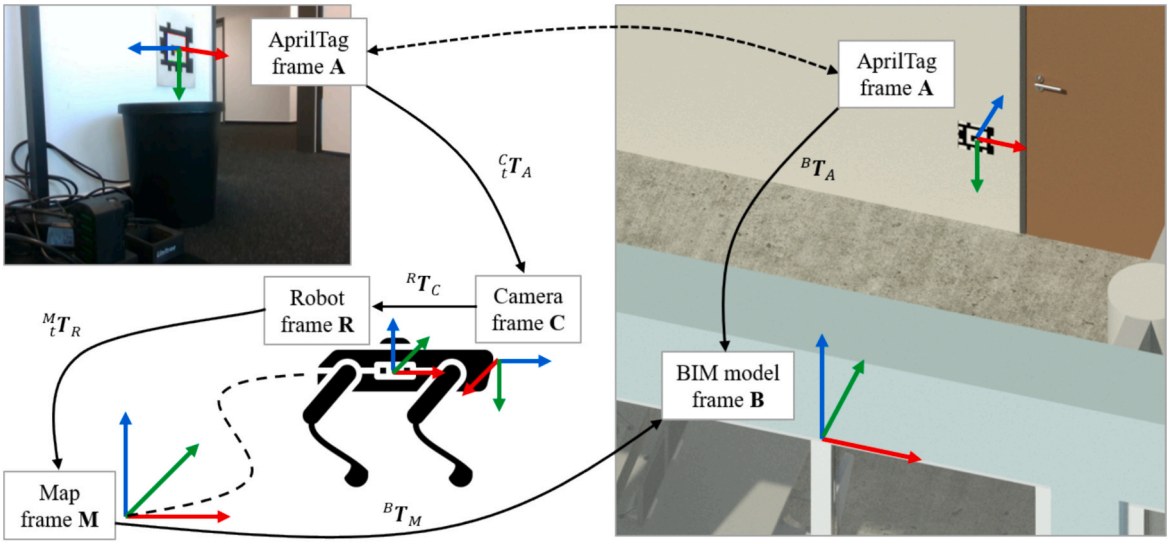


Fig. 4. Obtaining the transformation from the map frame to the BIM model frame using fiducial tags.

the transformation ${}^{B}\boldsymbol{T}_{M}$ obtained from Equation (1) only needs to be a sufficiently accurate starting value for the scan matching discussed in Subsection 3.3, as studies show that ICP algorithms converge robustly to perturbations of a few meters and degrees in real-world environments [47]. Accepted scan matches correct the trajectory of the robot in the map and thus errors introduced in Equation (1), since the origin of the map is arbitrary. The transformation ${}^{B}\boldsymbol{T}_{M}$ is modified to account for coordinate system requirements in the BIM model and the map according to the Robot Operating System (ROS) enhancement proposal (REP) 105 [48]. The conventions prescribe that the z-axes point upwards, i.e., in the opposite direction of gravity. Small pitch and roll angles incorporated in ${}^{B}\boldsymbol{T}_{M}$, that follow from the errors mentioned above, are thus corrected to zero. The final transformation ${}^{B}\boldsymbol{T}_{M}$ is used as a starting value for the scan matching of the local feature map with the BIM model, described in the next section.

3.3. Scan matching with the extracted features of the BIM model

Once the transformation ${}^{B}\boldsymbol{T}_{M}$ is established, the scan matching with the BIM model begins, utilizing the downsampled feature point clouds $\overline{\boldsymbol{P}}_{B}^{e}$ and $\overline{\boldsymbol{P}}_{B}^{p}$ of the BIM model. For efficiency, LIO-BIM reuses lidar point clouds processed in the LIO process for efficiency of scan matching with the BIM model. The LIO process is adopted from [30] and [32], where the latest lidar scan represented in $\boldsymbol{R}$ is deskewed, edge and planar features are extracted by evaluating the smoothness of points in a local region, and the feature point clouds are downsampled. To obtain the LIO in the map frame $\boldsymbol{M}$, the processed lidar point cloud is transformed to $\boldsymbol{M}$ using the predicted robot motion from the IMU, and matched against immediately preceding lidar point clouds. The scan matching aims to find edge and planar correspondences between the latest and the immediately preceding lidar point clouds. Subsequently, the Levenberg-Marquardt algorithm is employed to refine the odometry by minimizing the distance between features and their corresponding edge or planar patches, formulated as a non-linear optimization problem. While the LIO is computed for every lidar scan, a smaller number of keyframes are used for scan matching with the feature point clouds of the BIM model. Keyframes are added after exceeding user-defined thresholds on changes in distance $\tau_{M,d}$ and orientation $\tau_{M,o}$ of the robot.

The feature-based scan matching with the BIM model is shown in Fig. 5 and the pseudo code is provided in Listing 1. LIO-BIM reuses the processed lidar point clouds of keyframes in a small radius $r_{M,l}$ around the robot, denoted as "local feature map" ${}_{i}\boldsymbol{P}_{M}^{l}$ in this paper (line 5), where the index i refers to the number of the latest keyframe added to the factor graph. The local feature map containing edge and planar features is iteratively aligned to the BIM model by computing the transformation ${}_{i}^{B}\boldsymbol{T}_{M}^{\text{opt.}}$ which minimizes point-to-line and point-to-plane distances. To start the optimization, ${}_{i}^{B}\boldsymbol{T}_{M}^{\text{opt.}}$ is initialized with ${}^{B}\boldsymbol{T}_{M}$ obtained from Equation (1) (line 6). For a maximum number of iterations $n_{B,\text{LM}}$ (lines 7–23), ${}_{i}^{B}\boldsymbol{T}_{M}^{\text{opt.}}$ transforms the local feature map from the map frame to the BIM model frame (line 8). Each edge feature in the local feature map finds a corresponding edge line in the edge feature point cloud from the BIM model (line 10), computes the point-to-line distance (line 11), and stacks the equations (line 12). Accordingly, each planar feature in the local map locates a corresponding planar patch in the planar feature point cloud of the BIM model (line 15), computes the point-to-plane distance (line 16), and stacks the equations (line 17). A nonlinear iteration step based on the Levenberg-Marquardt algorithm is conducted to update the transformation ${}_{i}^{B}\boldsymbol{T}_{M}^{\text{opt.}}$ to minimize the point-to-line and point-to-plane distances of the correspondences and thereby to improve the alignment of the local feature map with the BIM model (line 19). If the convergence of the solution is detected, the optimization is stopped (lines 20–21). Finally, the transformation ${}_{i}^{B}\boldsymbol{T}_{M}^{\text{opt.}}$ and the point-to-line and point-to-plane distances of the feature points of the last iteration are returned.Listing 1: Scan matching of local feature map and BIM model.

1 **input:** $\overline{\boldsymbol{P}}_{B} = \{\overline{\boldsymbol{P}}_{B}^{e}, \overline{\boldsymbol{P}}_{B}^{p}\}$, ${}^{B}\boldsymbol{T}_{M}$, ${}_{i}\boldsymbol{P}_{M} = \{\ {}_{i}\boldsymbol{P}_{M}^{e}, {}_{i}\boldsymbol{P}_{M}^{p}\}$
2 **output:** ${}_{i}^{B}\boldsymbol{T}_{M}^{\text{opt.}}, \boldsymbol{d}_{B}$
3 **begin**
4 $\boldsymbol{d}_{B}^{e} = \boldsymbol{d}_{B}^{p} = \{\ \}$
5 Extract the local feature map ${}_{i}\boldsymbol{P}_{M}^{l}$ from ${}_{i}\boldsymbol{P}_{M}$
6 Initialize the optimal transformation from the map to the BIM model frame at keyframe i ${}_{i}^{B}\boldsymbol{T}_{M}^{\text{opt.}} = {}^{B}\boldsymbol{T}_{M}$
7 **for** a maximum number of iterations $n_{B,\text{LM}}$ **do**
8 Transform the local feature map to the BIM model frame ${}_{i}\boldsymbol{P}_{B}^{l} = {}_{i}^{B}\boldsymbol{T}_{M}^{\text{opt.}}\,{}_{i}\boldsymbol{P}_{M}^{l}$
9 **for** each edge point ${}_{k}\boldsymbol{p}_{B}^{e}$ in ${}_{i}\boldsymbol{P}_{B}^{l,e}$ **do**
10 Find a corresponding edge line in $\overline{\boldsymbol{P}}_{B}^{e}$ formed by two points ${}_{u}\overline{\boldsymbol{p}}_{B}^{e}$ and ${}_{v}\overline{\boldsymbol{p}}_{B}^{e}$
11 Compute point-to-line distance ${}_{k}d_{B}^{e} = \dfrac{\|({}_{k}\boldsymbol{p}_{B}^{e} - {}_{u}\overline{\boldsymbol{p}}_{B}^{e}) \times ({}_{k}\boldsymbol{p}_{B}^{e} - {}_{v}\overline{\boldsymbol{p}}_{B}^{e})\|}{\|{}_{u}\overline{\boldsymbol{p}}_{B}^{e} - {}_{v}\overline{\boldsymbol{p}}_{B}^{e}\|}$
12 Insert in vector of point-to-line distances $\boldsymbol{d}_{B}^{e} = \{\boldsymbol{d}_{B}^{e}, {}_{k}d_{B}^{e}\}$
13 **end**
14 **for** each planar point ${}_{k}\boldsymbol{p}_{B}^{p}$ in ${}_{i}\boldsymbol{P}_{B}^{l,p}$ **do**
15 Find a corresponding planar patch in $\overline{\boldsymbol{P}}_{B}^{p}$ formed by three points ${}_{u}\overline{\boldsymbol{p}}_{B}^{p}$, ${}_{v}\overline{\boldsymbol{p}}_{B}^{p}$, and ${}_{w}\overline{\boldsymbol{p}}_{B}^{p}$
16 Compute point-to-plane distance ${}_{k}d_{B}^{p} = \dfrac{\|({}_{k}\boldsymbol{p}_{B}^{p} - {}_{u}\overline{\boldsymbol{p}}_{B}^{p})(({}_{u}\overline{\boldsymbol{p}}_{B}^{p} - {}_{v}\overline{\boldsymbol{p}}_{B}^{p}) \times ({}_{u}\overline{\boldsymbol{p}}_{B}^{p} - {}_{w}\overline{\boldsymbol{p}}_{B}^{p}))\|}{\|({}_{u}\overline{\boldsymbol{p}}_{B}^{p} - {}_{v}\overline{\boldsymbol{p}}_{B}^{p}) \times ({}_{u}\overline{\boldsymbol{p}}_{B}^{p} - {}_{w}\overline{\boldsymbol{p}}_{B}^{p})\|}$
17 Insert in vector of point-to-plane distances $\boldsymbol{d}_{B}^{p} = \{\boldsymbol{d}_{B}^{p}, {}_{k}d_{B}^{p}\}$
18 **end**
19 Solve for the optimal transformation ${}_{i}^{B}\boldsymbol{T}_{M}^{\text{opt.}}$ at keyframe i using an iteration step of the Levenberg-Marquardt method $\min_{{}_{i}^{B}\boldsymbol{T}_{M}^{\text{opt.}}} \left(\sum_{{}_{k}\boldsymbol{p}_{B}^{e} \in {}_{i}\boldsymbol{P}_{B}^{l,e}} {}_{k}\boldsymbol{d}_{B}^{e} + \sum_{{}_{k}\boldsymbol{p}_{B}^{p} \in {}_{i}\boldsymbol{P}_{B}^{l,p}} {}_{k}\boldsymbol{d}_{B}^{p}\right)$
20 **if** ${}_{i}^{B}\boldsymbol{T}_{M}^{\text{opt.}}$ converges **then**
21 break
22 **end**
23 **end**
24 return ${}_{i}^{B}\boldsymbol{T}_{M}^{\text{opt.}}, \boldsymbol{d}_{B} = \{\boldsymbol{d}_{B}^{e}, \boldsymbol{d}_{B}^{p}\}$
25 **end**

LIO-BIM evaluates the alignment of the local feature map with the feature point cloud of the BIM model by calculating the root mean square error (RMSE) of all inlier correspondences and a fitness score, which measures the share of inlier correspondences. The inlier RMSE (RMSE_{d}) is defined as follows: Let $\boldsymbol{P}_{s}$ denote a source point cloud consisting of points $\boldsymbol{p}_{s}$, which is registered to a target point cloud $\boldsymbol{P}_{t}$ by applying the rigid transformation ${}^{t}\boldsymbol{T}_{s}$. Furthermore, let NN () denote the nearest neighbor function, which searches the nearest neighbor of the transformed source point in the target point cloud. For a given distance threshold d, the inlier RMSE amounts to:

$$\begin{aligned}\text{RMSE}_{d} &= \sqrt{\frac{1}{|\boldsymbol{S}_{d}|}\sum_{\boldsymbol{p}_{s}\in\boldsymbol{S}_{d}} \|\text{NN}({}^{t}\boldsymbol{T}_{s}\boldsymbol{p}_{s}) - {}^{t}\boldsymbol{T}_{s}\boldsymbol{p}_{s}\|^{2}}, \text{where } \boldsymbol{S}_{d} \\ &= \{\boldsymbol{p}_{s} \,|\, \|\text{NN}({}^{t}\boldsymbol{T}_{s}\boldsymbol{p}_{s}) - {}^{t}\boldsymbol{T}_{s}\boldsymbol{p}_{s}\| < d\} \subset \boldsymbol{P}_{s}\end{aligned} \tag{2}$$

The fitness score then equals the number of points in the inlier set $\boldsymbol{S}_{d}$ divided by the number of points in the source point cloud $\boldsymbol{P}_{s}$. In the case of evaluating the alignment of the local feature map with the feature point cloud of the BIM model, the local feature map corresponds to the source point cloud, the feature point cloud of the BIM model corresponds to the target point cloud, and the rigid transformation registering the point clouds is given by ${}_{i}^{B}\boldsymbol{T}_{M}^{\text{opt.}}$. Instead of searching for nearest neighbors, the distances between feature points $\boldsymbol{d}_{B}$ have already been calculated in Listing 1 and are reused, allowing for an efficient implementation. To ensure reliability and accuracy of the transformations in the presence of scan-BIM deviations, LIO-BIM only incorporates converged solutions of ${}_{i}^{B}\boldsymbol{T}_{M}^{\text{opt.}}$ with a low inlier RMSE and a high fitness score, which indicate a high compliance of the scan matching.

The acceptance of scan matches of the local feature map with the BIM model including the implementation of calculating the inlier RMSE and the fitness score is detailed in Listing 2. After checking the convergence of ${}_{i}^{B}\boldsymbol{T}_{M}^{\text{opt.}}$ (line 4), the point-to-line and point-to-plane distances of the

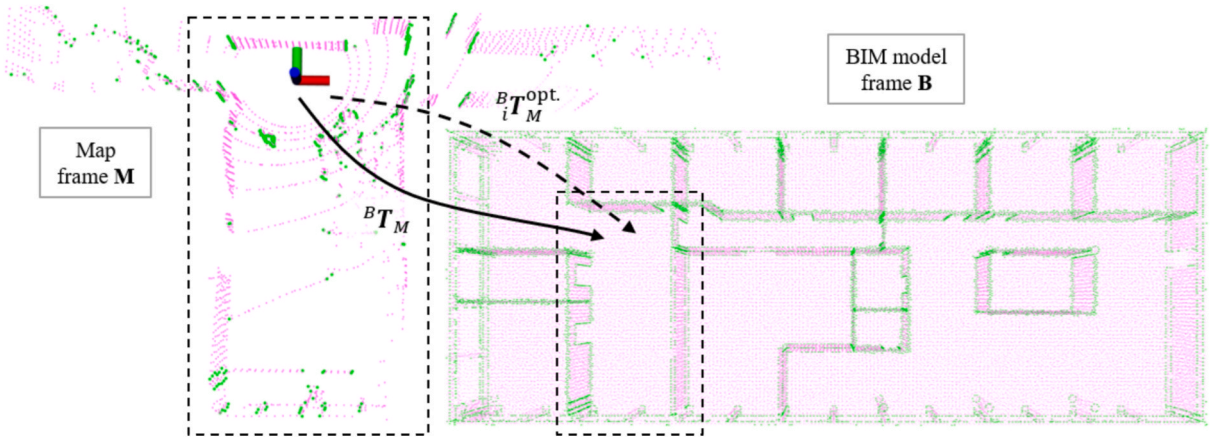


Fig. 5. Scan matching of the edge and planar features of the local map around the robot with the feature point cloud extracted from the BIM model.

last iteration step of Listing 1 stacked in $\boldsymbol{d}_B$ are evaluated. If distances are smaller than the distance threshold $\tau_{B,d}$, the corresponding points are considered inliers and count towards the computation of the inlier RMSE (line 9–10). After iterating over all distances in $\boldsymbol{d}_B$, the inlier RMSE is calculated as the mean of the squared distances of the inliers (line 16). The fitness score is calculated in (line 17). If the inlier RMSE is smaller than the RMSE threshold $\tau_{B,\text{RMSE}}$ and the fitness score is bigger than the fitness score threshold $\tau_{B,\text{fit}}$, scan matches are accepted (line 19). Otherwise, scan matches are rejected (line 22).

Listing 2: Acceptance of scan matches of the local feature map with the BIM model

```
1   input: ᴮᵢTₘ^opt., d_B
2   output: acceptance
3   begin
4     if ᴮᵢTₘ^opt. converged then
5       Initialize error for computing RMSE_d e = 0
6       Initialize number of inliers n_i = 0
7       for each entry _k d_B in d_B do
8         if _k d_B smaller than τ_B,d then
9           e = e + _k d_B^2
10          n_i = n_i + 1
11        end
12      end
13      if n_i == 0
14        return acceptance = false
15      end
16      RMSE_d = sqrt(e / n_i)
17      fit = n_i / |d_B|
18      if RMSE_d < τ_B,RMSE and fit > τ_B,fit then
19        return acceptance = true
20      end
21    end
22    return acceptance = false
23  end
```

Accepted scan matches are subsequently used to add BIM factors to the factor graph. The BIM factors align the trajectory and the map with the BIM model, explained in the next subsection.

3.4. Adding BIM factors to the factor graph

Every time the local feature map around a keyframe matches the BIM model with high compliance, a BIM factor associated to the keyframe is added to the underlying factor graph of LIO-BIM. The factor graph is implemented using the Georgia Tech Smoothing and Mapping (GTSAM) library [31]. A BIM factor is a unary factor, that corrects the trajectory and mapping to align with the BIM model. Since the factor graph is represented in the map frame $\boldsymbol{M}$, the accepted solution ${}^{B}_{i}\boldsymbol{T}^{\text{opt.}}_{M}$ calculated in Listing 1 is modified to obtain the BIM factor ${}_{i}\boldsymbol{T}^{\text{BIM}}_{M}$. ${}^{B}_{i}\boldsymbol{T}^{\text{opt.}}_{M}$ represents a transformation, that accurately aligns the local feature map around the last keyframe i to the BIM model frame. In other words, ${}^{B}_{i}\boldsymbol{T}^{\text{opt.}}_{M}$ is a local correction of the transformation from the map frame to the BIM model frame ${}^{B}\boldsymbol{T}_{M}$ obtained in Subsection 3.2. To obtain BIM factors that anchor the aligned pose of the local feature map in the factor graph, the inverse of ${}^{B}\boldsymbol{T}_{M}$ is multiplied with ${}^{B}_{i}\boldsymbol{T}^{\text{opt.}}_{M}$ and the pose of the last keyframe i in the map ${}^{M}_{i}\boldsymbol{T}_{R}$:

$$ {}_{i}\boldsymbol{T}^{\text{BIM}}_{M} = {}^{B}\boldsymbol{T}^{-1}_{M}\ {}^{B}_{i}\boldsymbol{T}^{\text{opt.}}_{M}\ {}^{M}_{i}\boldsymbol{T}_{R} \tag{3} $$

The noise associated with the BIM factor ${}_{i}\boldsymbol{T}^{\text{BIM}}_{M}$ is modeled using independent Gaussian noise with the variance equal to the inlier RMSE obtained from Equation (2) for each of the six dimensions of the BIM factor. Upon adding a BIM factor to the factor graph, the factor graph is optimized using the Bayes tree data structure and the approach introduced in [49]. The optimization process aims to compute the maximum a posteriori solution, which estimates robot trajectories and maps accurately. To avoid adding multiple BIM factors in confined spaces and to lower the computational complexity, the scan matching with the BIM model is skipped for a small number of keyframes $n_{B,kf}$ after a BIM factor has been added. The factor graph of a robot traversing an indoor office environment is visualized along with the BIM model in Fig. 6. In the following subsection, the general approach towards knowledge representation in LIO-BIM is described.

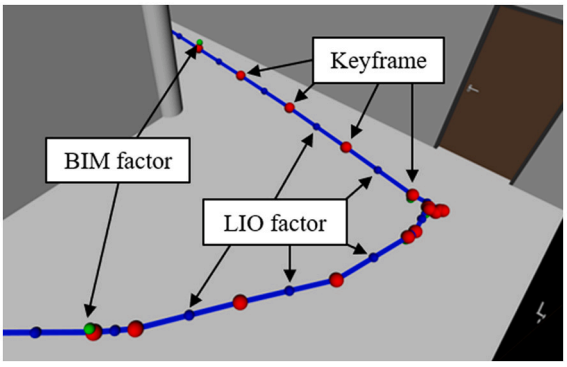


Fig. 6. Factor graph of a robot traversing an indoor office environment consisting of keyframes (red), LIO factors (blue), and BIM factors (green).

3.5. Knowledge representation in LIO-BIM

The LIO-BIM framework uses both semantic and geometric knowledge stored in IFC-based BIM models. First, semantic knowledge about the type of building elements is used to exclude objects that are either unlikely to be detected by lidar sensors, such as windows (IfcWindow), or that typically exhibit spatial variability in as-built conditions, such as doors (IfcDoor) and furniture (IfcFurniture). The semantic filtering operationalizes prior domain knowledge about sensor limitations and environmental uncertainty. Second, geometric knowledge in the form of surfaces and surface normals of the remaining building elements is used to extract edge and planar feature points, yielding a reduced yet salient representation of the environment. The edge and planar feature points form an abstract geometric model that supports efficient and robust scan matching in real time, where the abstraction corresponds to a knowledge-driven transformation, translating semantically enriched building models into representations optimized for sensor-based registration tasks.

Beyond the BIM-specific knowledge, the LIO-BIM framework also integrates robot-specific knowledge. For instance, the placement of fiducial tags, the design of the feature-based matching pipeline, and the keyframe selection strategy reflect established practices in robot localization under uncertainty. The integration of building model semantics and robotic heuristics into a unified SLAM-BIM coupling illustrates a hybrid use of domain knowledge from both the built environment and autonomous systems. The results of the feature-based scan matching with the BIM model are incorporated into a probabilistic factor graph as BIM factors. The BIM factors constrain the robot trajectory and map estimation relative to the BIM coordinate frame and act as knowledge-based anchoring mechanisms that fuse geometric observations with semantically structured prior information.

From the perspective of engineering informatics, the approach presented in this study constitutes an instance of knowledge representation and operationalization. The reduction of semantically rich BIM models to geometric feature abstractions represents a model-driven transformation, aligning structured domain semantics with computational efficiency. In the following section, experiments conducted to validate the LIO-BIM framework are described.

4. Validation tests

As will be presented in this section, tests are conducted to validate the LIO-BIM framework, each of which serving a distinct purpose. The validation tests are designed to showcase (i) the accuracy of trajectories created by the framework, (ii) the accuracy of maps in the form of point clouds created by the framework, and (iii) the real-time capabilities of the framework. The LIO-BIM framework is validated in two tests representing different environments. The three metrics mentioned above are calculated for each validation test. In the first test, LIO-BIM is validated using a quadruped robot traversing a cluttered indoor office environment represented by a BIM model. The second validation test is conducted on the ConSLAM dataset [51] showcasing a cluttered construction site environment. Both test environments contain significant scan-BIM deviations. An overview of the validation tests is given in Table 2, and detailed descriptions are provided in the following paragraphs. Background on the aforementioned three metrics will be provided in the Subsections 4.1–4.3 respectively, followed by a listing of the set of parameters used in the validation tests in Subsection 4.4. The validation test in the indoor office environment is presented in Subsection 4.5 and the validation test on the ConSLAM dataset is discussed in Subsection 4.6.

Table 2
Overview of the validation tests.

Tests	Indoor office environment	ConSLAM dataset
Environment	Cluttered indoor office environment	Construction site
Trajectory length	113 m	• Sequence 2: 225 m • Sequence 3: 340 m • Sequence 4: 275 m • Sequence 5: 320 m
Duration	325 s	• Sequence 2: 421 s • Sequence 3: 630 s • Sequence 4: 518 s • Sequence 5: 589 s
Purpose	Validate the framework in existing buildings with scan-BIM deviations	Validate the framework in buildings under construction with scan-BIM deviations at different times
Equipment	Robot equipped with • Velodyne VLP-16 lidar • LORD MicroStrain 3DM-GX5-25 IMU • Intel RealSense D435i camera	Handheld system consisting of • Velodyne VLP-16 lidar • Xsens MTi-610 IMU • Alvium U-319c, 3.2 MP camera
Computing device	Intel NUC11TNKV7	Intel NUC11TNKV7
Validation of accuracy of trajectories	–	• Method: Comparison against ground truth trajectories supplied by the ConSLAM dataset and the SLAM2REF paper • Metric: RMSE of translational APEs and rotational APEs
Validation of accuracy of point clouds	• Method: Comparison against point cloud of Faro Focus S 70 TLS • Metric: Inlier RMSE and fitness score	• Comparison against point cloud of Leica RTC 360 TLS • Metric: Inlier RMSE and fitness score
Validation of real-time capabilities	Recording of processing times and frames skipped	Recording of processing times and frames skipped

4.1. Accuracy of trajectories

The accuracy of trajectories is assessed by the absolute pose error (APE). The APE is a metric used for investigating the global consistency of SLAM trajectories [50]. The APE is calculated by comparing trajectories created by the LIO-BIM framework with a reference trajectory, which is considered the ground truth. Corresponding poses in the trajectories are identified by matching timestamps. The APE is based on the absolute relative pose $\boldsymbol{E}_i \in \mathrm{SE}(3)$ between an estimated pose ${}_i\widehat{\boldsymbol{p}}_{\mathrm{est}} \in \mathrm{SE}(3)$ and a reference pose ${}_i\widehat{\boldsymbol{p}}_{\mathrm{ref}} \in \mathrm{SE}(3)$ at timestamp i:

$$\boldsymbol{E}_i = {}_i\widehat{\boldsymbol{p}}_{\mathrm{est}}^{-1}\,{}_i\widehat{\boldsymbol{p}}_{\mathrm{ref}} \tag{4}$$

From $\boldsymbol{E}_i$, translational errors are obtained by calculating the translational APE and rotational errors are obtained by calculating the rotational APE:

$$\mathrm{tAPE}_i = \|\mathrm{trans}(\boldsymbol{E}_i)\| \tag{5}$$

$$\mathrm{rAPE}_i = \left|\mathrm{angle}\left(\log_{\mathrm{SO}(3)}(\mathrm{rot}(\boldsymbol{E}_i))\right)\right| \tag{6}$$

To combine the APEs of all N timestamps in the trajectories, the RMSE is used:

$$\mathrm{RMSE}_{\mathrm{tAPE}} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\mathrm{tAPE}_i^2} \tag{7}$$

$$\mathrm{RMSE}_{\mathrm{rAPE}} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\mathrm{rAPE}_i^2} \tag{8}$$

4.2. Accuracy of point clouds

The accuracy of maps, which are created in the form of point clouds

in LIO-BIM, is assessed by calculating the inlier RMSE as introduced in Equation (2). The inlier RMSE is a metric commonly used for measuring the fit of registered point clouds [51]. The inlier RMSE is calculated by comparing point clouds created by the LIO-BIM framework with a reference point cloud. Reference point clouds considered as ground truth are recorded using a terrestrial laser scanner (TLS). The distance threshold in the inlier RMSE allows to filter out outliers that may be present when certain areas are observed in the point cloud created by LIO-BIM, but not in the TLS point cloud. In the validation tests, the distance threshold is chosen as 0.3 m.

4.3. Real-time capabilities

To validate the real-time capabilities of the LIO-BIM framework, processing times are recorded. The real-time capabilities depend on the frequency of the lidar, which is operated at 10 Hz. If processing times exceed the times of incoming lidar scans, such that new lidar scans are available before previous ones have been processed, lidar scans are skipped to compute the LIO. If lidar scans are skipped only occasionally during operation, the LIO of the framework is considered to be real-time capable. The scan matching with the BIM model is computed for each keyframe (unless skipped after an accepted match) in parallel with the LIO and does not have to comply with the 10 Hz frequency, but with the addition of keyframes. Again, if the processing times exceed the times of keyframes being added, keyframes are skipped for the scan matching with the BIM model. If keyframes are skipped only occasionally during operation, such that subsequent local feature maps around keyframes still overlap, the scan matching with the BIM model is considered to be real-time capable.

4.4. Set of parameters

The set of parameters used in the validation tests have been chosen experimentally and are shown in Table 3. The set of parameters provides a tradeoff between accuracy and computation speed. The parameters $\tau_{B,d}$, $\tau_{B,\mathrm{RMSE}}$, and $\tau_{B,\mathrm{fit}}$ have a significant influence on accepting or rejecting scan matches with the BIM model and need to be chosen carefully. Optimally, the set of parameters allows for few, but highly accurate, accepted scan matches with the BIM model distributed along the trajectory.

4.5. Quadruped robot traversing an indoor office environment

The quadruped robot used for the first part of the validation tests, the "Intelligent DOcumentation Gadget" (IDOG) shown in Fig. 7, has successfully been utilized in different applications related to mobile structural health monitoring [52] and robot-based structural assessment [53]. The IDOG is built around the Unitree A1 robot [54] and is equipped with a lidar, an IMU, a camera, an external computer, and an additional battery powering the external computer. The hardware specification is shown in Table 4.

Table 3
Set of parameters used in the validation tests.

Parameter	Variable	Value
Initial sampling density of the surface of the BIM model	ρ_B	500 points/m^2
Curvature score threshold for edge feature points	$\tau_{B,e}$	0.6
Curvature score threshold for planar feature points	$\tau_{B,p}$	0.1
Minimum distance between features	$r_{B,f}$	0.1 m
Changes in distance to add keyframes	$\tau_{M,d}$	1.0 m
Changes in orientation to add keyframes	$\tau_{M,o}$	0.2 rad
Local feature map search radius	$r_{M,l}$	5.0 m
Keyframes skipped after accepted scan match with the BIM model	$n_{B,kf}$	3
Maximum number of LM iterations for scan matching with BIM model	$n_{B,\mathrm{LM}}$	30
Distance threshold for outliers	$\tau_{B,d}$	0.6 m
Inlier RMSE threshold	$\tau_{B,\mathrm{RMSE}}$	0.1 m
Fitness score threshold	$\tau_{B,\mathrm{fit}}$	0.65

The validation tests are conducted in an indoor office environment (39 m x 16 m) associated with a BIM model. Planar and edge feature points are extracted from the BIM model, as described in Section 3.1. An AprilTag is placed in the BIM model and at the same location in the real office environment. The IDOG is manually controlled to observe the AprilTag and to traverse the indoor office environment, while sensor data from the lidar at 10 Hz, from the IMU at 500 Hz, and from the camera at 15 Hz with a resolution of 640x480 pixels are recorded and stored in a rosbag file. The rosbag file is played back to the LIO-BIM framework to compute the trajectory of the IDOG and the as-built map of the indoor office environment. Furthermore, the rosbag file can be played back to other SLAM frameworks for comparison. In the validation tests, comparisons with LIO-SAM are conducted.

The feature extraction of the BIM model is conducted by filtering out windows, doors, and furniture that may disturb the scan matching with the BIM model. The surface of the mesh of the BIM model, which has an area of 6,506.51 m^2, is sampled with a density of 500 points per m^2, resulting in 3,253,255 initial sample points. The principal curvature of the points is estimated by considering all neighboring points within a radius of 0.15 m. The curvature scores are calculated and points with a curvature score less than 0.1 are considered planar feature points, and points with a curvature score greater than 0.6 are considered edge feature points. The feature points are sorted by their curvature score. Neighboring points to the most prominent edge and surface feature points in a radius of 0.1 m are discarded. Finally, the feature point cloud extracted from the BIM model contains 9,032 edge feature points and 77,743 planar feature points. The extracted edge and planar feature point clouds of the BIM model are shown in Fig. 8.

The AprilTag is placed on a wall next to the charging stations for the IDOG and in the associated BIM model. After starting the LIO-BIM framework, the transformation from map frame to BIM model frame ${}^{B}\boldsymbol{T}_{M}$ is obtained by observing the AprilTag.

A reference scan of the indoor office environment is recorded using the Faro Focus S 70 terrestrial laser scanner [59]. The maximum point error for registering the individual TLS scans is 0.35 cm, indicating a high accuracy of the reference scan. The accuracy of the point cloud created by the LIO-BIM framework is validated by calculating the inlier RMSE of the point cloud created by LIO-BIM registered to the TLS reference scan. The ICP function of the program CloudCompare [60] is used to finely register the point clouds. Since no ground truth data is available for the trajectory, the accuracy of the trajectory is not

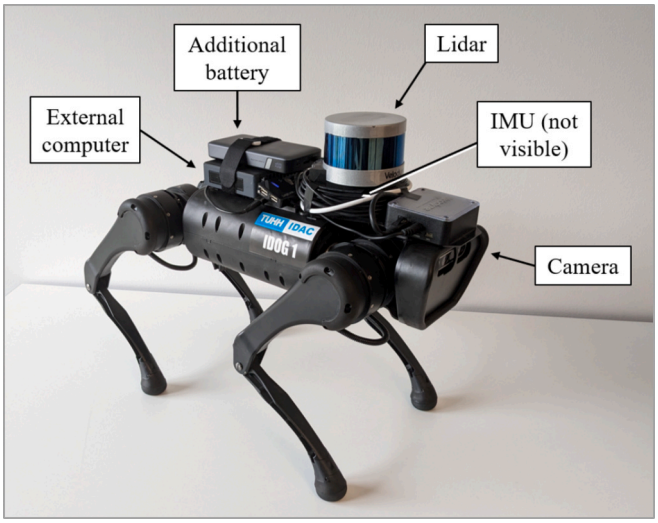


Fig. 7. Quadruped robot "IDOG".

Table 4
Hardware specification of the components of the IDOG.

Lidar	Velodyne VLP-16 [55]		• 16 channels • Measurement range: 100 *m* • Range accuracy: Up to ± 3 cm • Field of view (vertical): +15.0° to −15.0° (30°) • Angular resolution (vertical): 2.0° • Field of view (horizontal): 360° • Angular resolution (horizontal/azimuth): 0.1° – 0.4° • Rotation rate: 5 Hz – 20 Hz
IMU	LORD MicroStrain 3DM-GX5-25 [56]	Accelero-meter	• Measurement range: ±8 g • Resolution: 0.02 mg • Sampling rate: 1 kHz
		Gyroscope	• Measurement range: ±300°/sec • Resolution: <0.003°/sec • Sampling rate: 4 kHz
		Magneto-meter	• Measurement range: ±2.5 Gauss • Sampling rate: 50 Hz
Camera	Intel RealSense D435i [57]		• Up to 1920x1080 RGB resolution • RGB field of view: Horizontal 69°±1°, vertical 42°±1°
External computer	Intel NUC11TNKV7 [58]		• CPU: Intel Core i7-1185G7 • RAM: 32 GB

validated in the first test. The validation of the trajectory is conducted in the second test detailed in Subsection 4.6.

4.6. ConSLAM dataset

The second validation test is conducted using the publicly available ConSLAM dataset [51]. The dataset includes five sequences of data recorded periodically on a real-world construction site with a handheld system. The data incorporates synchronized lidar scans, IMU measurements, and color and near-infrared camera images. Furthermore, TLS scans are provided for each sequence as the ground truth for the point cloud of the construction site, and ground truth poses of lidar scans, i.e. the trajectory, have been recovered. The reader is referred to [51] for a detailed description of the dataset. In the validation test, the BIM model supplied by [17] is used, which has been created based on the ground truth TLS scan of sequence number two of the ConSLAM dataset. AprilTags are added in the BIM model at the respective locations the AprilTags are visible in the color camera images of the real-world recordings. An example is shown in Fig. 9, where the AprilTag with the ID 2 of the color camera images of sequence 2 is shown. All computations are carried out on the same computer as in the previous validation test, presented in Subsection 4.5. The feature extraction is conducted as described in the first validation test. The surface of the mesh of the BIM model has a size of 60,632.2 m^2, which results in 30,316,119 initial sample points by sampling with a density of 500 points per m^2. After extracting features, the feature point cloud contains 82,503 edge feature points and 1,110,114 planar feature points. The BIM model including the extracted edge and planar feature point clouds are shown in Fig. 10.

Since the recording of the first sequence of the ConSLAM dataset contains faulty IMU measurements, the LIO-BIM framework is tested on the remaining sequences number two, three, four, and five. In the end of each run, the trajectory is saved in the “tum” format [50], which contains timestamps and poses for every keyframe. The trajectory is compared against the ground truth trajectory of the ConSLAM dataset and against the trajectory of LIO-SAM to showcase that LIO-BIM is able to correct the long-term drift of traditional SLAM frameworks. The comparison is conducted using the evo package [61], devised for the evaluation of odometry and SLAM frameworks. Since the ground truth data provided in the ConSLAM dataset does not follow the REP 105 convention, the final trajectories of LIO-BIM and LIO-SAM are aligned to the ground truth trajectories using the Umeyama alignment [62]. Furthermore, the trajectories of LIO-BIM are compared against the ground truth trajectories proposed by the SLAM2REF framework in [17]. As discussed in [17], significant discrepancies are present in the

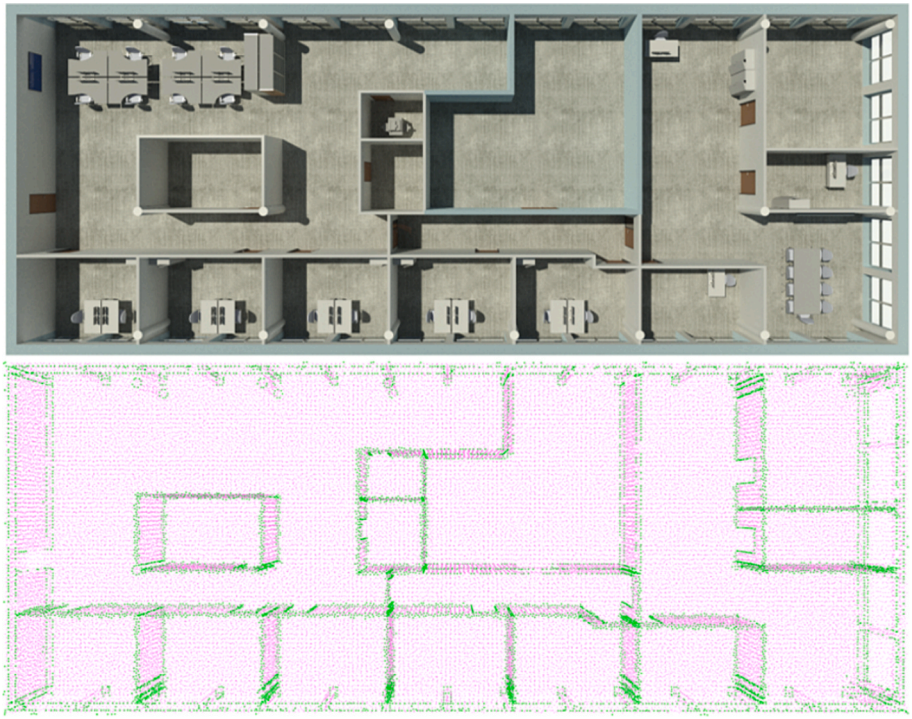


Fig. 8. Extracted edge and planar feature points of the BIM model of the indoor office environment.

Fig. 9. AprilTags visible in the color camera images of the ConSLAM dataset are placed in the BIM model to obtain the transformation from map frame to BIM model frame.

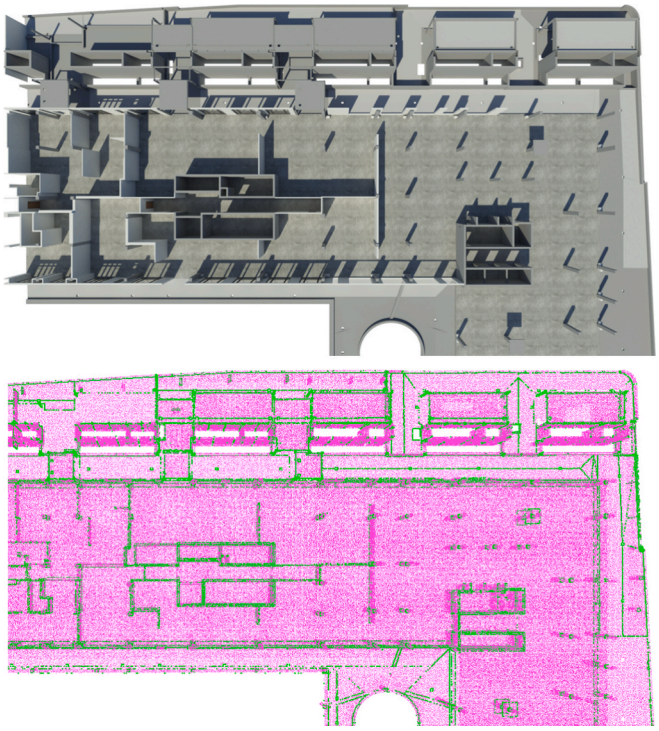


Fig. 10. Extracted edge and planar feature points of the BIM model of the ConSLAM dataset.

ground truth poses provided in the ConSLAM dataset in small sections of the trajectories.

The point clouds created by LIO-BIM are compared against the ground truth TLS scans provided in the ConSLAM dataset. The maximum error for registering individual TLS scans is given with an RMSE of 0.902 cm for sequence number four to 0.994 cm for sequence number five [51], indicating a high accuracy of the ground truth TLS scan. Since the BIM model has been created based on the TLS scan of sequence number two of the ConSLAM dataset according to [17] and the TLS scans of all sequences are georeferenced according to [51], the point clouds created by LIO-BIM are expected to lie in the same coordinate frame (i.e., the BIM model frame $\boldsymbol{B}$). However, the BIM model shows to be slightly shifted and the TLS scans in the ConSLAM dataset are not finely registered to each other. Therefore, the point clouds created by LIO-BIM are registered to the TLS scans of the ConSLAM dataset in the same way as described in Subsection 4.5. The results of both validation tests are presented in the next section.

5. Results

In this section, the results of the validation tests are presented. After showing the trajectories and point clouds aligned with the BIM models created by LIO-BIM for both validation tests, the results regarding the three metrics defined in the previous section are presented for both validation tests.

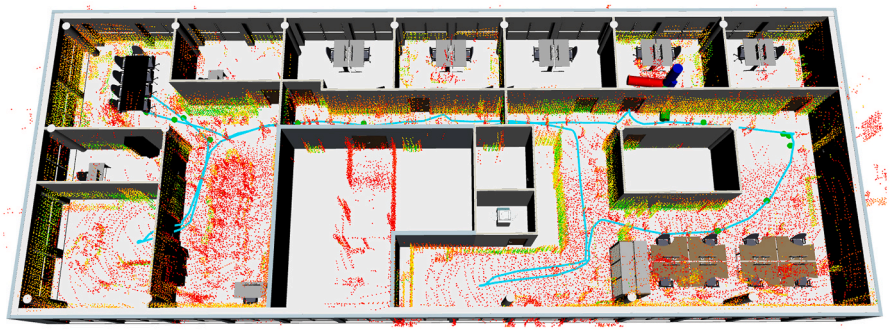
Fig. 11 shows the trajectories and point clouds created by LIO-BIM aligned with the BIM models in the visualization software RViz. Fig. 11a displays the trajectory, the point cloud, and the BIM model corresponding to the indoor office environment, while Fig. 11 (b-e) visualizes the trajectories, the point clouds, and the BIM model corresponding to sequences two to five of the ConSLAM dataset, at the end of each run.

Table 5 provides the translational and rotational errors of the trajectories compared against the ground truth provided in the ConSLAM dataset for the sequences two to five of the ConSLAM dataset according to Equations (4)-(8). In addition to the RMSE, the maximum error is provided. Accordingly, Table 6 provides the translational and rotational errors of the trajectories compared against the ground truth provided in the ConSLAM dataset. To show the impact of the scan matching with the BIM model of LIO-BIM on the overall accuracy, the tests are repeated and compared with LIO-SAM. Fig. 12 depicts the translational APE mapped on to the trajectory created by LIO-BIM for each sequence compared against the ground truth trajectory provided by [17]. The translational APE, which represents the alignment of the trajectory created by LIO-BIM and the ground truth trajectory, is displayed as a color gradient.

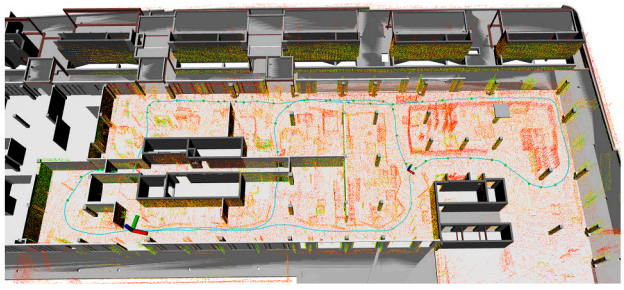
Table 7 shows the inlier RMSEs and fitness scores of point clouds created by LIO-BIM compared to the ground truth with a distance threshold of 0.3 m. Again, the results of LIO-SAM are given for comparison. Fig. 13 displays the alignment of the point clouds created by LIO-BIM and the ground truth TLS scans. Fig. 13a shows the point cloud created by LIO-BIM and the TLS scan of the indoor office environment, and Fig. 13b of sequence number three of the ConSLAM dataset, respectively. The point clouds created by LIO-BIM are colored according to a color gradient displaying the distance of each point to the nearest neighboring point in the TLS scans. The color gradient moves from green (0 cm) to yellow (15 cm) to red (30 cm and above) to display outliers and errors in the point cloud created by LIO-BIM.

The results and processing times of the scan matching with the BIM model of LIO-BIM are given in Table 8. Table 9 shows the results and processing times of the lidar scan matching of LIO-BIM. The processing times are visualized in boxplots in Fig. 14. Fig. 14a displays the processing times of the scan matching with the BIM model and Fig. 14b shows the processing times of the lidar scan matching of LIO-BIM.

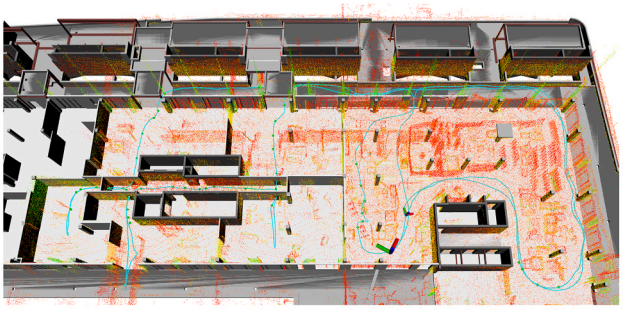
The results of the validation tests are discussed in the next section.



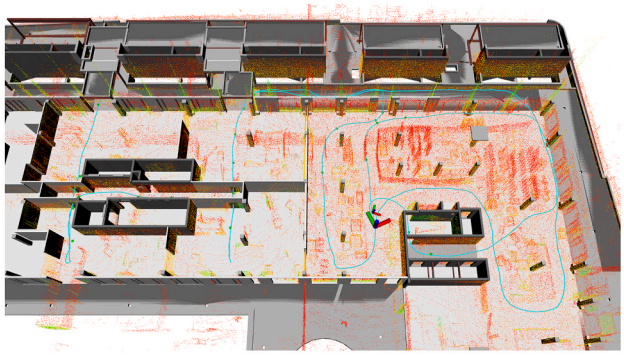
(a) Indoor office environment



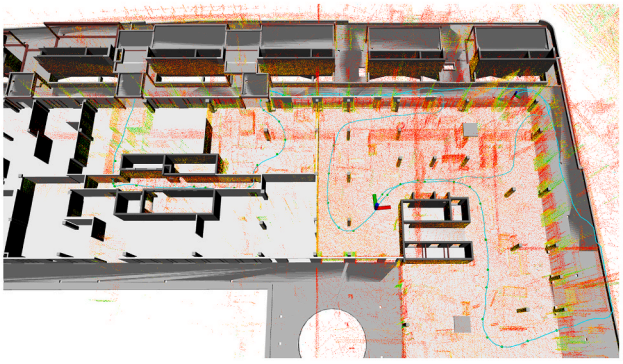
(b) Sequence 2



(c) Sequence 3



(d) Sequence 4



(e) Sequence 5

Fig. 11. Trajectories and point clouds aligned with the corresponding BIM model created by the LIO-BIM framework (Sequence 1 is omitted as described in Subsection 3.2).

Table 5
Translational and rotational errors of trajectories compared against the ground truth provided in the ConSLAM dataset.

			Sequence 2 (225 m)	Sequence 3 (340 m)	Sequence 4 (275 m)	Sequence 5 (320 m)
LIO-BIM	Translational error [cm]	RMSE	10.21	10.40	12.85	15.68
		Max	103.84	32.06	37.29	73.66
	Rotational error [deg]	RMSE	1.3665	1.3791	1.1199	1.4438
		Max	16.8973	7.5847	4.2863	7.1668
LIO-SAM	Translational error [cm]	RMSE	27.28	9.94	10.68	16.09
		Max	112.84	33.48	29.15	54.83
	Rotational error [deg]	RMSE	1.5796	1.3275	1.2403	1.4824
		Max	12.070	6.4213	5.8163	6.9557

Table 6
Translational and rotational errors of trajectories compared against the ground truth provided in [17].

			Sequence 2 (225 m)	Sequence 3 (340 m)	Sequence 4 (275 m)	Sequence 5 (320 m)
LIO-BIM	Translational error [cm]	RMSE	5.97	26.68	10.25	12.57
		Max	20.67	195.70	23.20	34.88
	Rotational error [deg]	RMSE	0.7992	1.4730	1.0910	1.2362
		Max	4.4042	7.7004	4.5813	6.8999
LIO-SAM	Translational error [cm]	RMSE	6.58	26.23	10.09	18.02
		Max	28.62	190.18	27.50	44.30
	Rotational error [deg]	RMSE	0.8347	1.4690	1.0193	1.3595
		Max	4.4082	6.2511	4.9040	5.3831

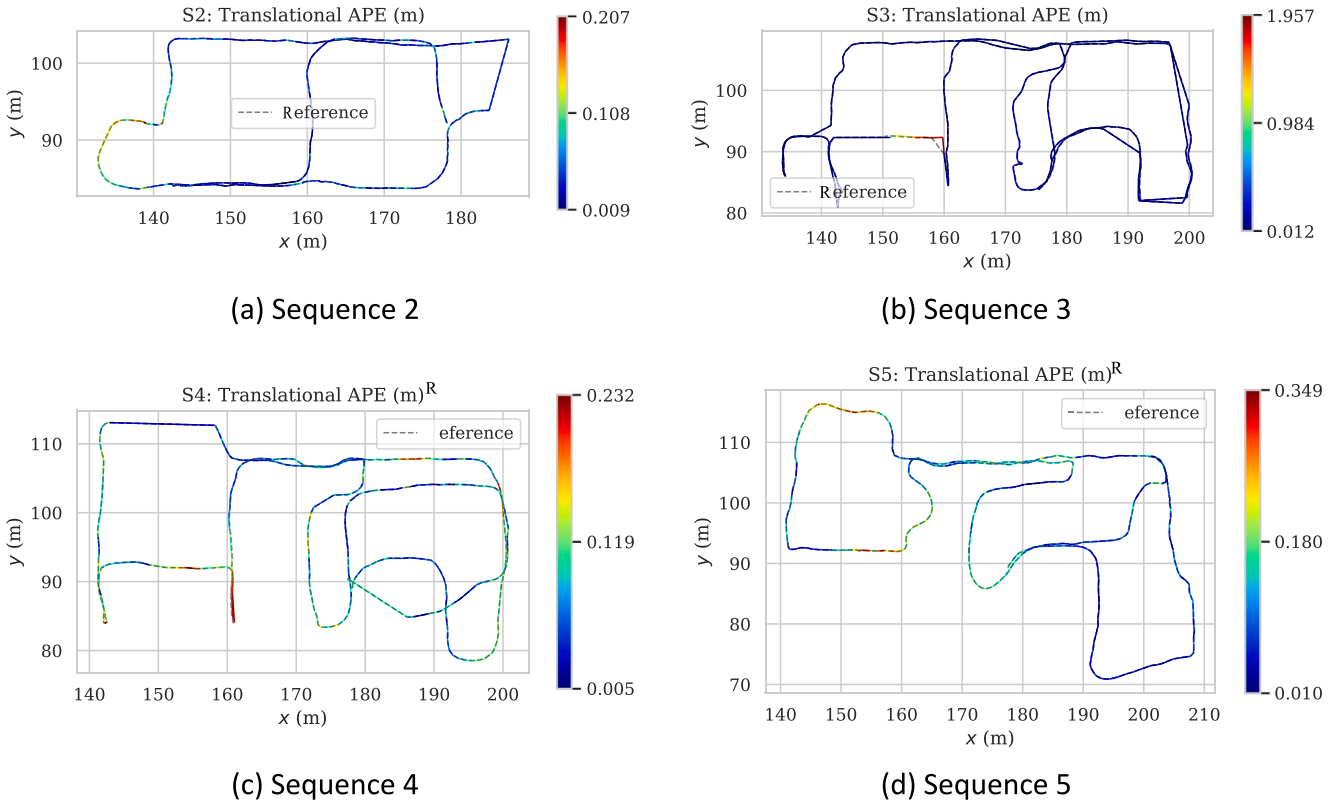


Fig. 12. Translational APE mapped onto trajectories created by LIO-BIM.

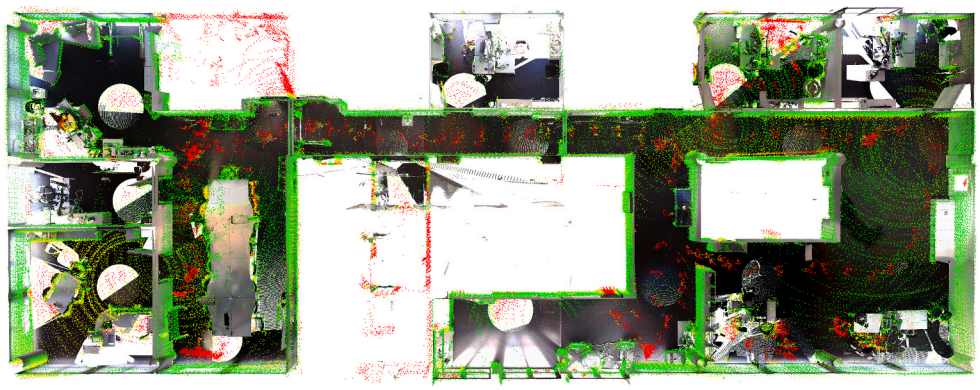
Table 7
Inlier RMSEs and fitness scores of point clouds against the ground truth with a distance threshold of 0.3 m.

		Indoor office environment	ConSLAM			
			Sequence 2	Sequence 3	Sequence 4	Sequence 5
LIO-BIM	$RMSE_{0.3}$ [cm]	6.57	6.33	6.30	6.38	7.89
	Fitness score	0.9535	0.9776	0.9650	0.9292	0.9290
LIO-SAM	$RMSE_{0.3}$ [cm]	8.42	11.78	6.27	6.12	9.07
	Fitness score	0.9536	0.8988	0.9670	0.9241	0.9047

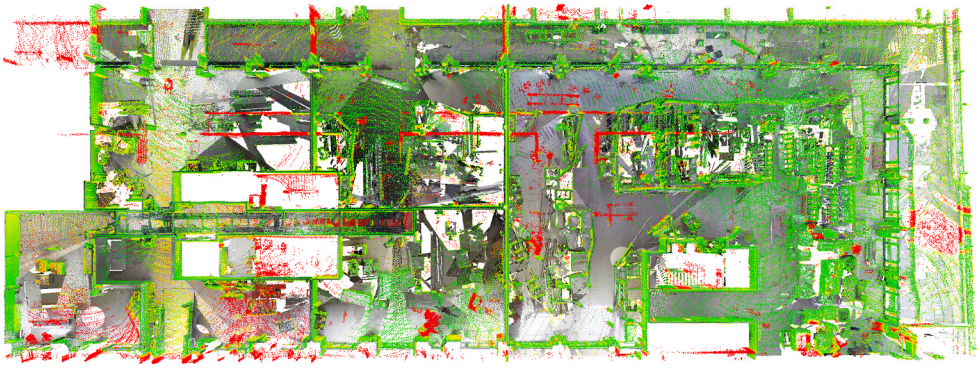
6. Discussion

The validation tests show that the LIO-BIM framework is capable of 3D localization and mapping with respect to BIM models in real-time. As can be seen from Fig. 11, LIO-BIM creates trajectories and point clouds consistent with the respective BIM models.

The trajectories created by LIO-BIM show errors in the low centimeter range. Since the BIM model created by [17] is based on the TLS scan of sequence number two of the ConSLAM dataset, scan-BIM deviations between the as-planned BIM model and the as-built state increase with each sequence. The increasing deviations are reflected in the APE results compared against the ground truth trajectories provided in the ConSLAM dataset displayed in Table 5. The translational error is the lowest for sequence number two with an RMSE of 10.21 cm and grows to 15.68 cm in sequence number five, recorded four and a half months after sequence number two. Although the error has grown slightly, LIO-BIM is able to create a consistent trajectory aligning with the BIM model. The rotational error lies between an RMSE of 1.1199° (sequence number four) and 1.4438° (sequence number five). Comparing LIO-BIM against the ground truth trajectories provided in [17] yields lower errors for the sequences number two, four, and five, as can be seen in Table 6. In the sequences number two, four, and five, the RMSE of the translational APE



(a) Point cloud created by LIO-BIM (color gradient) aligned with the TLS scan of the indoor office environment



(b) Point cloud created by LIO-BIM (color gradient) aligned with the TLS scan of sequence number three of the ConSLAM dataset

Fig. 13. Alignment of point clouds created by LIO-BIM and TLS scans including a visualization of the point-to-point distance.

Table 8
Results and processing times of the scan matching with the BIM model of LIO-BIM.

		Indoor office environment	ConSLAM			
			Sequence 2	Sequence 3	Sequence 4	Sequence 5
Keyframes	Total	225	399	712	529	542
	Accepted	15	52	76	45	73
	Rejected	148	92	172	184	122
	Skipped after acceptance	45	156	228	135	219
	Skipped due to processing times	15	99	236	165	128
	Percentage skipped due to processing times	6.667 %	24.812 %	33.146 %	31.191 %	23.616 %
Processing times	Mean [s]	0.84947	1.48145	1.71745	1.58183	1.76646
	Standard deviation [s]	0.36499	0.76778	0.84832	0.74531	0.92354
	Median [s]	0.74824	1.29328	1.56127	1.47903	1.68111
	Min [s]	0.28527	0.38502	0.12126	0.23809	0.23664
	Max [s]	2.26235	3.17739	3.25303	3.22739	3.36233

Table 9
Results and processing times of the lidar scan matching of LIO-BIM.

		Indoor office environment	ConSLAM			
			Sequence 2	Sequence 3	Sequence 4	Sequence 5
Lidar frames	Total	3120	4163	6244	5125	5835
	Skipped	0	255	856	683	836
	Percentage skipped	0 %	6.125 %	13.709 %	13.327 %	14.327 %
Processing times	Mean [s]	0.02836	0.08240	0.09430	0.09174	0.09549
	Standard deviation [s]	0.01407	0.04119	0.05252	0.05406	0.05451
	Median [s]	0.02648	0.07560	0.08682	0.08134	0.09241
	Min [s]	0.00005	0.00003	0.00005	0.00005	0.00005
	Max [s]	0.15316	0.48707	0.51106	0.83815	0.44883

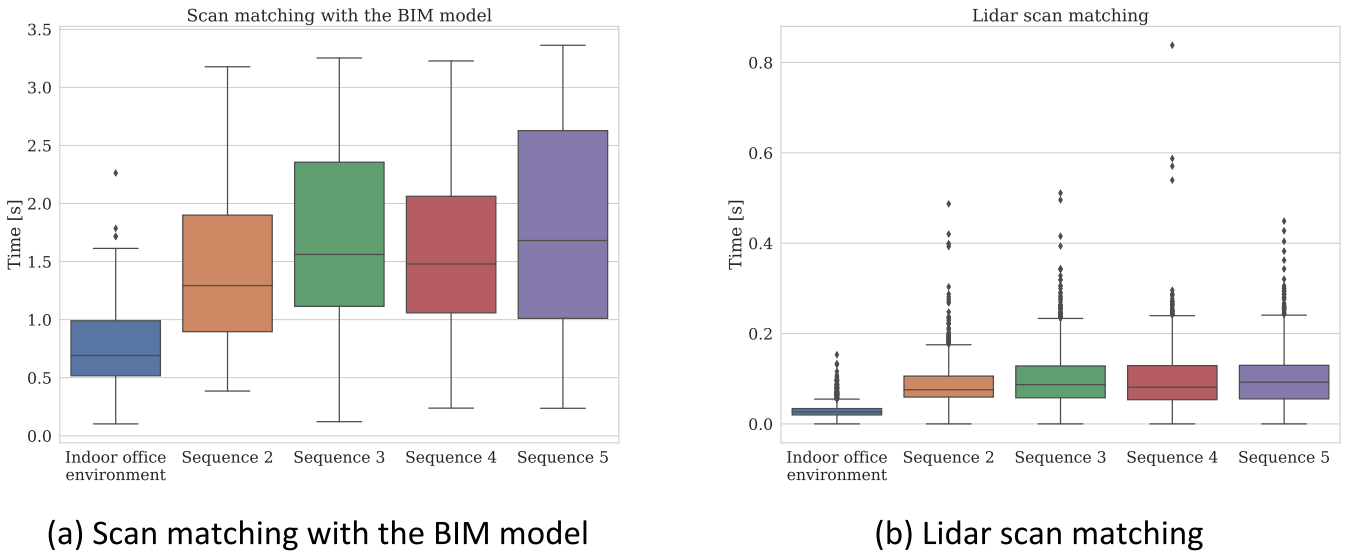


Fig. 14. Processing times of the scan matching of LIO-BIM.

ranges from 5.97 cm to 12.57 cm and the maximum error is reduced significantly. In sequence number 3, outliers are present in a small part of the trajectory, which can be seen in Fig. 12b. However, since the trajectory of LIO-BIM aligns accurately with the ground truth trajectory provided by the ConSLAM dataset at the respective part and no discrepancies are visible in the related point cloud in Fig. 13b in the respective area, the outlier may originate from an error in the ground truth provided in [17]. Compared with LIO-SAM, LIO-BIM achieves a more accurate trajectory on sequence number two, slightly less accurate trajectories on sequences number three and four, and a slightly more accurate trajectory on sequence number five. Indicated by the results on sequence number two and five, the scan matching with the BIM model in LIO-BIM is able to decrease the drift observed in traditional SLAM frameworks. As a result, the localization accuracy of LIO-BIM is sufficient to enable reliable indoor navigation [63]. Furthermore, the accuracy is sufficient to automate several building inspection tasks that require centimeter-level localization accuracy. For example, half-cell potential measurements conducted to detect corrosion in reinforced concrete require an accuracy of $\pm$ 10 cm, which is achieved except for a few outliers as shown in Fig. 12 [64].

Similar to the trajectories, the point clouds created by LIO-BIM show errors in the low centimeter range. As displayed in Table 7, the inlier RMSE for a distance threshold of 0.3 m is 6.57 cm for the indoor office environment and lies between 6.30 cm and 7.89 cm for the ConSLAM dataset. In comparison with LIO-SAM, LIO-BIM achieves a lower inlier RMSE for the indoor office environment, a significantly lower inlier RMSE on sequence number two, slightly higher inlier RMSEs on sequences number three and four, and a lower inlier RMSE on sequence number five of the ConSLAM dataset. The results in terms of the accuracy of the trajectories are similar to the results in terms of the accuracy of the point clouds, since the point clouds are composed of lidar scans whose poses depend on the trajectories. The fitness scores obtained in the validation tests lie between 0.9290 and 0.9776, indicating a high overlap between the point clouds created by LIO-BIM and the TLS scans, both in the indoor office environment and on the ConSLAM dataset. Fig. 13 shows that most outliers account to areas that have been observed with the lidars, but not with the TLS scanners. Fig. 13a displays the author controlling the quadruped robot being mapped in the point cloud, resulting in further outliers. In Fig. 13b, outliers in the point cloud created by LIO-BIM are visible originating from reflections in windows. Fig. 15 shows the error in detail. The windows at the top of the figure reflect lidar beams, resulting in the reconstruction of fictitious obstacles,

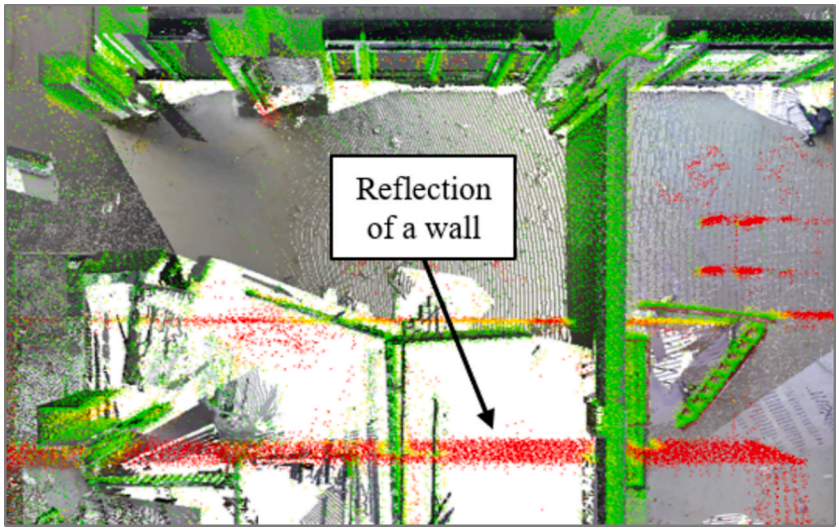


Fig. 15. Reflection of a wall disturbing the point cloud created by LIO-BIM.

in this case a wall.

The results regarding the accuracy of the trajectories and the point clouds show, that LIO-BIM is able to deal with cluttered environments and scan-BIM deviations. An example is provided in Fig. 16. Although a wall is placed at a wrong position in the BIM model of the indoor office environment, LIO-BIM is able to provide a consistent and correct trajectory and point cloud, as scan matches with the BIM model in the area around the wrongly positioned wall are rejected due to low compliance.

The results and processing times of the scan matching with the BIM model displayed in Table 8 show that the scan matching takes on average 0.84947 s for the indoor office environment and 1.48145 s up to 1.76646 s for the ConSLAM dataset. As a result, 6.667 % of keyframes in the indoor office environment and 23.616 % up to 33.146 % of keyframes in the ConSLAM dataset are skipped to perform scan matching due to exceeding the times of keyframes being added. Since the radius of the local map around the robot is set to be five meters and keyframes are being added after each meter or exceeding 0.2 rad, subsequent local feature maps around keyframes have sufficient overlap. Therefore, the scan matching with the BIM model is considered to run in real-time. The reason of the higher computation times for the ConSLAM dataset is believed to be the larger size of the BIM model. To ensure real-time capabilities with larger BIM models, splitting up the feature point clouds extracted from BIM models into multiple smaller feature point clouds may reduce the lookup times to find nearest neighbors and thereby the total computation times.

The lidar scan matching is supposed to run at 10 Hz, i.e., faster than 0.1 s. The results and processing times displayed in Table 9 show that on average, lidar scans are processed in 0.02836 s in the indoor office environment tests and between 0.08240 s and 0.09549 s in the ConSLAM dataset tests. Consequently, no lidar scan is skipped in the indoor office environment tests, and small fluctuations in the processing times result in 6.125 % up to 14.327 % lidar scans being skipped for scan matching in the ConSLAM dataset tests. The lidar scans are skipped only occasionally, so the lidar scan matching for LIO is considered to run in real-time, as discussed in Subsection 4.3.

7. Summary and conclusions

The deployment of mobile robots requires accurate robot localization and navigation to automate tasks. In the construction industry, BIM models have become increasingly prevalent and can be used by robots to help localize and navigate. However, BIM models represent as-planned states of buildings, which may diverge from the as-built states, complicating the robot localization problem. In this paper, the LIO-BIM framework is introduced for 3D localization and mapping aligned with BIM models in real-time. Fiducial tags are employed to solve the initial alignment from the map to the BIM model frame. Building upon SLAM techniques, LIO-BIM integrates additional scan matching of local maps around robots with BIM models. High-compliant scan matches result in BIM factors being added to the underlying factor graphs. BIM factors tie trajectories and maps of robots to the BIM models in real-time. Except for the manual placement of fiducial tags in the building and the BIM model to solve the initial alignment, LIO-BIM represents a fully automated solution.

In this study, LIO-BIM has been validated using a robot traversing an indoor office environment and on the publicly available ConSLAM dataset. The validation tests show that LIO-BIM achieves real-time 3D localization and mapping with respect to BIM models with accuracy errors in the low centimeter range. LIO-BIM creates trajectories and maps in the format of point clouds with high accuracy, the translational errors obtained in the validation tests are in the low centimeter range. During run time, LIO-BIM skips a manageable amount of scan matching calculations, displaying reliable real-time capabilities. In conclusion, LIO-BIM outperforms similar approaches devised for localization and mapping with respect to BIM models reported in the literature. Similar approaches either provide only 2D localization and mapping, do not run in real-time, or are not able to handle significant scan-BIM deviations. LIO-BIM provides a robust solution to all the points mentioned, while achieving high accuracies, improving upon state-of-the-art SLAM systems. The code of LIO-BIM is provided as open source at https://github.com/janstueh/LIO-BIM.

However, LIO-BIM also presents several limitations. While the approach of using fiducial tags to solve the initial alignment between the map and the BIM model frame requires minimal effort and works robustly, the approach requires manual input and modifications to the environment. Furthermore, the set of parameters has been experimentally selected for the validation tests. While the set of parameters gives good results in the tests conducted, the set of parameters may be

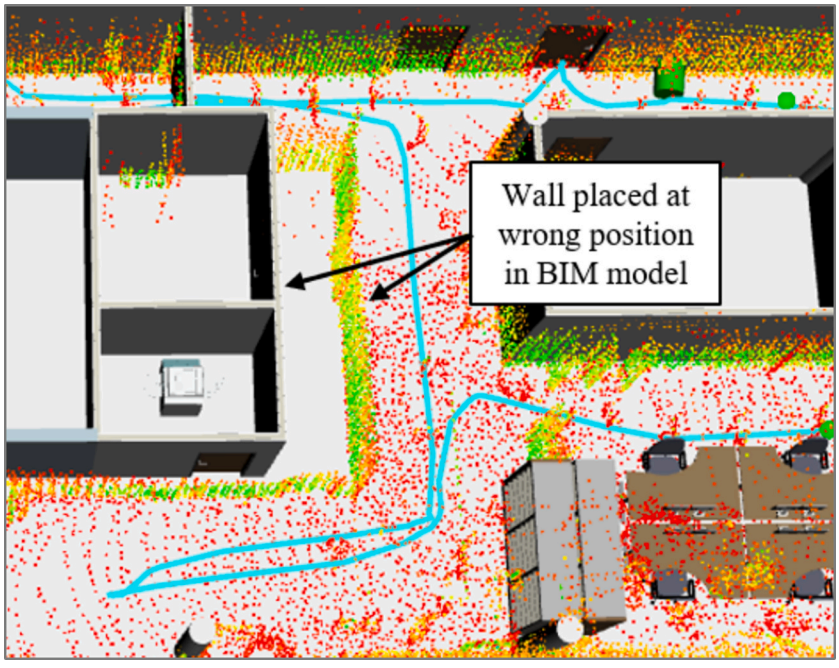


Fig. 16. Robustness against scan-BIM deviations.

optimized for different environments, BIM models of varying LODs, different sensors, or different computing devices. Lastly, lidar reflections in windows result in erroneous maps and may deteriorate the localization accuracy.

In future work, investigations to address the limitations to improve LIO-BIM may be conducted. The calculation of initial alignments solely based on proprioceptive sensors may be considered, so as no modifications to the environment will be required. However, approaches using only proprioceptive sensors tend to be computationally complex and pose challenges on real-time capabilities. Lightweight approaches scaling well with the size of BIM models need to be investigated to rely solely on proprioceptive sensors. Learnable parameters may be investigated that automatically adapt to changing circumstances. The issue of lidar reflections may be addressed by utilizing information on the position of windows in the BIM model or in the camera data to selectively filter out lidar beams hitting windows. Eventually, path planning in BIM models may be investigated allowing robots to plan and execute paths to goal points defined in the BIM model frame, for example, to inspect assets. By combining path planning in BIM models with LIO-BIM, an end-to-end automated pipeline for autonomous robot navigation within BIM models is expected to be achieved.

CRediT authorship contribution statement

Jan Stühenberg: Writing – original draft, Visualization, Validation, Software, Methodology, Investigation, Formal analysis, Data curation, Conceptualization. **Kay Smarsly:** Writing – review & editing, Supervision, Resources, Project administration.

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.

Acknowledgments

The authors would like to express their gratitude to the Institute of Technical Logistics of Hamburg University of Technology for recording the TLS scan of the indoor office environment.

Appendix

Table 10 lists all mathematical variables used in the paper along descriptions.

Table 10
Summary of variables used in the paper.

Variable	Description
$\boldsymbol{R}$	Robot frame
$\boldsymbol{M}$	Map frame
$\boldsymbol{B}$	BIM model frame
${}^{M}_{t}\boldsymbol{T}_{R}$	Transformation from the robot frame to the map frame
${}^{B}\boldsymbol{T}_{M}$	Transformation from the map frame to the BIM model frame
${}^{B}_{t}\boldsymbol{T}_{R}$	Transformation from the robot frame to the BIM model frame
ρ_B	Initial sampling density of the surface of the BIM model
k_1, k_2	Principal curvatures
k_B	Curvature score
$\tau_{B,e}$	Curvature score threshold for edge feature points
$\tau_{B,p}$	Curvature score threshold for planar feature points
$r_{B,f}$	Minimum distance between features
$\overline{\boldsymbol{P}_B}$	Downsampled feature point cloud of the BIM model
$\overline{\boldsymbol{P}}^{e}_{B}$	Downsampled edge feature point cloud of the BIM model
$\overline{\boldsymbol{P}}^{p}_{B}$	Downsampled planar feature point cloud of the BIM model
$\boldsymbol{C}$	Camera frame
$\boldsymbol{A}$	AprilTag frame
${}^{C}_{t}\boldsymbol{T}_{A}$	Transformation from the AprilTag frame to the camera frame
${}^{B}\boldsymbol{T}_{A}$	Transformation from the AprilTag frame to the BIM model frame
${}^{R}\boldsymbol{T}_{C}$	Transformation from the camera frame to the robot frame
$\tau_{M,d}$	Changes in distance to add keyframes
$\tau_{M,o}$	Changes in orientation to add keyframes
$r_{M,l}$	Local feature map search radius
i	Index of keyframe
${}_{i}\boldsymbol{P}^{l}_{M}$	Local feature map
${}^{B}_{i}\boldsymbol{T}^{\text{opt.}}_{M}$	Transformation aligning the local feature map to the BIM model frame
${}_{i}\boldsymbol{P}_{M}$	Downsampled feature point cloud of the map
${}_{i}\boldsymbol{P}^{e}_{M}$	Downsampled edge feature point cloud of the map
${}_{i}\boldsymbol{P}^{p}_{M}$	Downsampled planar feature point cloud of the map
$\boldsymbol{d}_B$	Vector of feature distances
$\boldsymbol{d}^{e}_{B}$	Vector of point-to-line distances
$\boldsymbol{d}^{p}_{B}$	Vector of point-to-plane distances
${}_{i}\boldsymbol{P}^{l}_{B}$	Local feature map transformed to the BIM model frame
${}_{u}\overline{\boldsymbol{P}}^{e}_{B}$, ${}_{v}\overline{\boldsymbol{P}}^{e}_{B}$	Two points in the edge feature point cloud of the BIM model forming a line correspondence
${}_{u}\overline{\boldsymbol{P}}^{p}_{B}$, ${}_{v}\overline{\boldsymbol{P}}^{p}_{B}$, ${}_{w}\overline{\boldsymbol{P}}^{p}_{B}$	Three points in the planar feature point cloud of the BIM model forming a planar correspondence
RMSE_d	Inlier RMSE
$\boldsymbol{P}_s$	Source point cloud
$\boldsymbol{p}_s$	Points in the source point cloud
$\boldsymbol{P}_t$	Target point cloud
${}^{t}\boldsymbol{T}_{s}$	Rigid transformation

(*continued on next page*)

Table 10 (*continued*)

Variable	Description
NN ()	Nearest neighbor function
d	Distance threshold
$\boldsymbol{S}_d$	Set of inliers
$n_{B,kf}$	Keyframes skipped after accepted scan match with the BIM model
$n_{B,\mathrm{LM}}$	Maximum number of LM iterations for scan matching with BIM model
$\tau_{B,d}$	Distance threshold for outliers
$\tau_{B,\mathrm{RMSE}}$	Inlier RMSE threshold
$\tau_{B,\mathrm{fit}}$	Fitness score threshold
e	Error for computing RMSE_d
n_i	Number of inliers
${}_i\boldsymbol{T}_M^{\mathrm{BIM}}$	BIM factor
$\boldsymbol{E}_i$	Absolute pose error
${}_i\widehat{\boldsymbol{P}}_{\mathrm{est}}$	Estimated pose
${}_i\widehat{\boldsymbol{P}}_{\mathrm{ref}}$	Reference pose
tAPE_i	Translation part of the absolute pose error
rAPE_i	Rotational part of the absolute pose error
$\mathrm{RMSE}_{\mathrm{tAPE}}$	Root mean square error of the translational errors
$\mathrm{RMSE}_{\mathrm{rAPE}}$	Root mean square error of the rotational errors

Data availability

The source code of LIO-BIM is available at https://github.com/janstueh/LIO-BIM.

References

[1] S. Halder, K. Afsari, Robots in inspection and monitoring of buildings and infrastructure: A systematic review, Appl. Sci. 13 (4) (2023) 2304, https://doi.org/10.3390/app13042304.

[2] Tandon, A., Stühmenberg, J., Dragos, K., & Smarsly, K., 2024. Autonomous navigation of quadruped robots for monitoring and inspection of civil infrastructure. In: Proceedings of the 20th International Conference on Computing in Civil and Building Engineering. Montréal, Canada, 08/25/2024. DOI: 10.1007/978-3-031-87364-5_29.

[3] K. Smarsly, K. Dragos, J. Stühmenberg, M. Worm, Mobile structural health monitoring based on legged robots, Infrastructures 8 (2023) 136, https://doi.org/10.3390/infrastructures8090136.

[4] Ibrahim, A., Sabet, A., & Golparvar-Fard, M., 2019. BIM-driven mission planning and navigation for automatic indoor construction progress detection using robotic ground platform. In: Proceedings of the 2019 European Conference on Computing in Construction. Chania, Greece, 07/10/2019, pp. 182-189. DOI: 10.35490/EC3.2019.195.

[5] J.J. Lin, M. Golparvar-Fard, Construction progress monitoring using cyber-physical systems, in: C.J. Anumba, N. Roofigari-Esfahan (Eds.), Cyber-Physical Systems in the Built Environment, Springer International Publishing, Cham, Switzerland, 2020, https://doi.org/10.1007/978-3-030-41560-0_5.

[6] P. Pauwels, R. de Koning, B. Hendrikx, E. Torta, Live semantic data from building digital twins for robot navigation: Overview of data transfer methods, Adv. Eng. Inf. 56 (2023) 101959, https://doi.org/10.1016/j.aei.2023.101959.

[7] C. Follini, V. Magnago, K. Freitag, M. Terzer, C. Marcher, M. Riedl, A. Giusti, D. Matt, BIM-integrated collaborative robotics for application in building construction and maintenance, Robotics 10 (1) (2021) 2, https://doi.org/10.3390/robotics10010002.

[8] Hendrikx, R. W. M., Pauwels, P., Torta, E., Bruyninckx, H. J.P., & van de Molengraft, M. J. G., 2021. Connecting semantic building information models and robotics: An application to 2D LiDAR-based localization. In: Proceedings of the 2021 IEEE International Conference on Robotics and Automation. Xi'an, China, 05/30/2021, pp. 11654-11660. DOI: 10.1109/ICRA48506.2021.9561129.

[9] S. Kim, M. Peavy, P.-C. Huang, K. Kim, Development of BIM-integrated construction robot task planning and simulation system, Autom. Constr. 127 (2021) 103720, https://doi.org/10.1016/j.autcon.2021.103720.

[10] K. Kim, M. Peavy, BIM-based semantic building world modeling for robot task planning and execution in built environments, Autom. Constr. 138 (2022) 104247, https://doi.org/10.1016/j.autcon.2022.104247.

[11] Moura, M. S., Rizzo, C., & Serrano, D., 2021. BIM-based localization and mapping for mobile robots in construction. In: Proceedings of the 2021 IEEE International Conference on Autonomous Robot Systems and Competitions. Santa Maria da Feira, Portugal, 04/28/2021, pp. 12-18. DOI: 10.1109/ICARSC52212.2021.9429779.

[12] M. Peavy, P. Kim, H. Oyediran, K. Kim, Integration of real-time semantic building map updating with adaptive Monte Carlo localization (AMCL) for robust indoor mobile robot localization, Appl. Sci. 13 (2) (2023) 909, https://doi.org/10.3390/app13020909.

[13] Yin, H., Liew, J. M., Lee, W. L., Ang, M. H., & Yeoh, J. K.W., 2022. Towards BIM-based robot localization: A real-world case study. In: Proceedings of the 39th International Symposium on Automation and Robotics in Construction. Bogotá, Colombia, 07/13/2022, pp. 71-77. DOI: 10.22260/ISARC2022/0012.

[14] H. Yin, Z. Lin, J.K.W. Yeoh, Semantic localization on BIM-generated maps using a 3D LiDAR sensor, Autom. Constr. 146 (2023) 104641, https://doi.org/10.1016/j.autcon.2022.104641.

[15] Vega Torres, M. A., Braun, A., & Borrmann, A., 2022. Occupancy grid map to pose graph-based map: Robust BIM-based 2D LiDAR localization for lifelong indoor navigation in changing and dynamic environments. In: Proceedings of the 14th European Conference on Product and Process Modelling. Trondheim, Norway, 09/14/2022, pp. 567-574. DOI: 10.1201/9781003354222-72.

[16] Vega Torres, M. A., Braun, A., & Borrmann, A., 2023. BIM-SLAM: Integrating BIM models in multi-session SLAM for lifelong mapping using 3D LiDAR. In: Proceedings of the 40th International Symposium on Automation and Robotics in Construction. Chennai, India, 07/07/2023. DOI: 10.22260/ISARC2023/0070.

[17] M.A. Vega Torres, A. Braun, A. Borrmann, SLAM2REF: Advancing long-term mapping with 3D LiDAR and reference map integration for precise 6-DoF trajectory estimation and map extension, Construction Robotics 8 (2) (2024) 13, https://doi.org/10.1007/s41693-024-00126-w.

[18] BIMForum, 2024. Level of development specification, Version 2024, Part I. Available online: https://bimforum.org/wp-content/uploads/2024/11/LOD-Spec-2024-Part-I-official-English.pdf (accessed 04/09/2025).

[19] Blum, H., Stiefel, J., Cadena, C., Siegwart, R., & Gawel, A., 2021. Precise robot localization in architectural 3D plans. In: Proceedings of the 38th International Symposium on Automation and Robotics in Construction (ISARC). Dubai, UAE, 11/02/2021, pp. 755-762. DOI: 10.22260/ISARC2021/0102.

[20] L. Liu, B. Li, S. Zlatanova, P. Van Oosterom, Indoor navigation supported by the Industry Foundation Classes (IFC): A survey, Autom. Constr. 121 (2021) 103436, https://doi.org/10.1016/j.autcon.2020.103436.

[21] Lee, J. H., Park, J.-H., & Jang, B.-T., 2018. Design of robot based work progress monitoring system for the building construction site. In: Proceedings of the 2018 International Conference on Information and Communication Technology Convergence. Jeju, South Korea, 10/17/2018, pp. 1420-1422. DOI: 10.1109/ICTC.2018.8539444.

[22] C. Follini, M. Terzer, C. Marcher, A. Giusti, D.T. Matt, in: Combining the Robot Operating System with Building Information Modeling for Robotic Applications in Construction Logistics, Springer, Cham, Switzerland, 2020, https://doi.org/10.1007/978-3-030-48989-2_27.

[23] I.J. Cox, Blanche – an experiment in guidance and navigation of an autonomous robot vehicle, IEEE Trans Rob Autom 7 (2) (1991) 193–204, https://doi.org/10.1109/70.75902.

[24] D. Fox, Adapting the sample size in particle filters through KLD-sampling, The International Journal of Robotics Research 22 (12) (2003) 985–1003, https://doi.org/10.1177/0278364903022012001.

[25] A. Prorok, A. Martinoli, Accurate indoor localization with ultra-wideband using spatial models and collaboration, The International Journal of Robotics Research 33 (4) (2014) 547–568, https://doi.org/10.1177/0278364913500364.

[26] N. Kayhani, W. Zhao, B. McCabe, A.P. Schoellig, Tag-based visual-inertial localization of unmanned aerial vehicles in indoor construction environments using an on-manifold extended Kalman filter, Autom. Constr. 135 (2022) 104112, https://doi.org/10.1016/j.autcon.2021.104112.

[27] Thrun, S., Burgard, W., & Fox, D., 2005: Probabilistic robotics. Cambridge, MA, USA: MIT Press. ISBN: 978-0-262-20162-9.

[28] C. Stachniss, J.J. Leonard, S. Thrun, Simultaneous localization and mapping, in: B. Siciliano, O. Khatib (Eds.), Springer Handbook of Robotics, Springer International Publishing, Cham, Switzerland, 2016, pp. 1153–1176, https://doi.org/10.1007/978-3-319-32552-1_46.

[29] Segal, A., Haehnel, D., & Thrun, S., 2009. Generalized-ICP. In: Proceedings of Robotics: Science and Systems V. Seattle, WA, USA, 06/28/2009. DOI: 10.15607/RSS.2009.V.021.

[30] Shan, T., Englot, B., Meyers, D., Wang, W., Ratti, C., & Rus, D., 2020. LIO-SAM: Tightly-coupled lidar inertial odometry via smoothing and mapping. In: Proceedings of the 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems. Las Vegas, NV, USA, 10/24/2020, pp. 5135-5142. DOI: 10.1109/IROS45743.2020.9341176.
[31] F. Dellaert, M. Kaess, Factor graphs for robot perception, Foundations and Trends in Robotics 6 (1–2) (2017) 1–139, https://doi.org/10.1561/2300000043.
[32] J. Zhang, S. Singh, Low-drift and real-time lidar odometry and mapping, Auton. Robot. 41 (2) (2017) 401–416, https://doi.org/10.1007/s10514-016-9548-2.
[33] Borrmann, A., König, M., Koch, C., & Beetz, J., 2018. Building information modeling: Technology foundations and industry practice. Cham, Switzerland: Springer. DOI: 10.1007/978-3-319-92862-3.
[34] A. Zhu, P. Pauwels, E. Torta, H. Zhang, B. De Vries, Data linking and interaction between BIM and robotic operating system (ROS) for flexible construction planning, Autom. Constr. 163 (2024) 105426, https://doi.org/10.1016/j.autcon.2024.105426.
[35] Z. Wang, H. Li, X. Yang, Vision-based robotic system for on-site construction and demolition waste sorting and recycling, Journal of Building Engineering 32 (2020) 101769, https://doi.org/10.1016/j.jobe.2020.101769.
[36] Ercan Jenny, S., Blum, H., Gawel, A., Siegwart, R., Gramazio, F., & Kohler, M., 2020. Online synchronization of building model for on-site mobile robotic construction. In: Proceedings of the 37th International Symposium on Automation and Robotics in Construction. Kitakyushu, Japan, 10/14/2020, pp. 1508-1514. DOI: 10.22260/ISARC2020/0209.
[37] Boniardi, F., Caselitz, T., Kümmerle, R., & Burgard, W., 2017. Robust LiDAR-based localization in architectural floor plans. In: Proceedings of the 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems. Vancouver, Canada, 09/24/2017, pp. 3318-3324. DOI: 10.1109/IROS.2017.8206168.
[38] F. Boniardi, T. Caselitz, R. Kümmerle, W. Burgard, A pose graph-based localization system for long-term navigation in CAD floor plans, Rob. Auton. Syst. 112 (2019) 84–97, https://doi.org/10.1016/j.robot.2018.11.003.
[39] Gao, L. & Kneip, L., 2022. FP-Loc: Lightweight and drift-free floor plan-assisted LiDAR localization. In: Proceedings of the 2022 International Conference on Robotics and Automation. Philadelphia, PA, USA, 05/23/2022, pp. 4142-4148. DOI: 10.1109/ICRA46639.2022.9812361.
[40] Wang, X., Marcotte, R. J., & Olson, E., 2019. GLFP: Global localization from a floor plan. In: Proceedings of the 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems. Macau, China, 11/03/2019, pp. 1627-1632. DOI: 10.1109/IROS40897.2019.8968061.
[41] N. Zimmerman, T. Guadagnino, X. Chen, J. Behley, C. Stachniss, Long-term localization using semantic cues in floor plan maps, IEEE Rob. Autom. Lett. 8 (1) (2023) 176–183, https://doi.org/10.1109/LRA.2022.3223556.
[42] IfcOpenShell, 2024. IfcOpenShell: The open source IFC toolkit and geometry engine. Available online: https://ifcopenshell.org/ (accessed 08/09/2024).
[43] PointCloudLibrary, 2024. PointCloudLibrary. Available online: https://pointclouds.org/ (accessed 08/09/2024).
[44] Olson, E., 2011. AprilTag: A robust and flexible visual fiducial system. In: Proceedings of the 2011 IEEE International Conference on Robotics and Automation. Shanghai, China, 05/09/2011, pp. 3400-3407. DOI: 10.1109/ICRA.2011.5979561.
[45] Krogius, M., Haggenmiller, A., Olson, E., 2019. Flexible layouts for fiducial tags. In: Proceedings of the 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems. Macau, China, 11/03/2019, pp. 1898-1903. DOI: 10.1109/IROS40897.2019.8967787.
[46] AprilRobotics, 2024. AprilTag 3. Available online: https://github.com/AprilRobotics/apriltag (accessed 08/09/2024).
[47] F. Pomerleau, F. Colas, R. Siegwart, S. Magnenat, Comparing ICP variants on real-world data sets: Open-source library and experimental protocol, Auton. Robot. 34 (3) (2013) 133–148, https://doi.org/10.1007/s10514-013-9327-2.
[48] Meeussen, W., 2010. REP 105: Coordinate Frames for Mobile Platforms. Available online: https://www.ros.org/reps/rep-0105.html (accessed 08/09/2024).
[49] M. Kaess, H. Johannsson, R. Roberts, V. Ila, J.J. Leonard, F. Dellaert, iSAM2: Incremental smoothing and mapping using the Bayes tree, The International Journal of Robotics Research 31 (2) (2012) 216–235, https://doi.org/10.1177/0278364911430419.
[50] Sturm, J., Engelhard, N., Endres, F., Burgard, W., & Cremers, D., 2012. A benchmark for the evaluation of RGB-D SLAM systems. In: Proceedings of the 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems. Vilamoura-Algarve, Portugal, 10/07/2012, pp. 573-580. DOI: 10.1109/IROS.2012.6385773.
[51] M. Trzeciak, K. Pluta, Y. Fathy, L. Alcalde, S. Chee, A. Bromley, I. Briliakis, P. Alliez, ConSLAM: Construction data set for SLAM, J. Comput. Civ. Eng. 37 (3) (2023) 04023009, https://doi.org/10.1061/JCCEE5.CPENG-5212.
[52] Smarsly, K., Dragos, K., Stührenberg, J., & Worm, M., 2023. Structural health monitoring of civil infrastructure using mobile robots. In: Proceedings of the 19th International Conference on Computing in Civil and Building Engineering. Cape Town, South Africa, 10/26/2022, pp. 127-138. DOI: 10.1007/978-3-031-32515-1_10.
[53] Johann, S., Stührenberg, J., Tandon, A., Dragos, K., Bartholmai, M., Strangfeld, C., & Smarsly, K., 2024. Implementation and validation of robot-enabled embedded sensors for structural health monitoring. In: Proceedings of the 11th European Workshop on Structural Health Monitoring. Potsdam, Germany, 06/10/2024. DOI: 10.58286/29679.
[54] Unitree Robotics, 2020. A1 robot. Available online: https://www.unitree.com/en/a1/ (accessed 08/09/2024).
[55] Velodyne LiDAR, Inc., 2018. Puck LITE datasheet. Available online: https://www.mapix.com/wp-content/uploads/2018/07/63-9286_Rev-H_Puck-LITE_Datasheet_Web.pdf (accessed 08/09/2024).
[56] MicroStrain Sensing Systems, 2018. 3DM-GX5-25 datasheet. Available online: https://www.microstrain.com/sites/default/files/3dm-gx5-25_datasheet_8400-0093.pdf (accessed 08/09/2024).
[57] Intel, 2019. Intel RealSense D400 Series Product Family datasheet. Available online: https://www.intel.com/content/dam/support/us/en/documents/emerging-technologies/intel-realsense-technology/Intel-RealSense-D400-Series-Datasheet.pdf (accessed 08/09/2024).
[58] Intel, 2023. Intel NUC Board/Kit/Mini PC: NUC11TNi3 / NUC11TNi5 / NUC11TNv5 / NUC11TNi7 / NUC11TNv7 Technical Product Specification. Available online: https://www.intel.com/content/dam/support/us/en/documents/intel-nuc/NUC11TN_TechProdSpec.pdf (accessed 08/09/2024).
[59] FARO, 2021. FARO Focus Laser Scanner. Available online: https://media.faro.com/-/media/Project/FARO/FARO/FARO/Resources/2_TECH-SHEET/TechSheet_Focus_Laser_Scanner/TechSheet_Focus_Laser_Scanner_EN.pdf (accessed 08/09/2024).
[60] CloudCompare, 2024. CloudCompare: 3D point cloud and mesh processing software. Available online: https://www.cloudcompare.org/ (accessed 08/09/2024).
[61] Grupp, M., 2017. evo: Python package for the evaluation of odometry and SLAM. Available online: https://github.com/MichaelGrupp/evo (accessed 08/09/2024).
[62] S. Umeyama, Least-squares estimation of transformation parameters between two point patterns, IEEE Trans. Pattern Anal. Mach. Intell. 13 (4) (1991) 376–380, https://doi.org/10.1109/34.88573.
[63] C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, I. Reid, J. J. Leonard, Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age, IEEE Trans. Rob. 32 (6) (2016) 1309–1332, https://doi.org/10.1109/TRO.2016.2624754.
[64] C. Sodeikat, Merkblatt B3 – Elektrochemische Potentialmessungen zur Detektion von Bewehrungsstahlkorrosion, Beton- Stahlbetonbau 105 (8) (2010) 529–538, https://doi.org/10.1002/best.201000043.