

# S-Graphs 2.0 – A Hierarchical-Semantic Optimization and Loop Closure for SLAM

Hriday Bavle<sup>1</sup>, Jose Luis Sanchez-Lopez<sup>1</sup>, Muhammad Shaheer<sup>1</sup>,  
Javier Civera<sup>2</sup> and Holger Voos<sup>1</sup>

**Abstract**—The hierarchical nature of 3D scene graphs aligns well with the structure of man-made environments, making them highly suitable for representation purposes. Beyond this, however, their embedded semantics and geometry could also be leveraged to improve the efficiency of map and pose optimization, an opportunity that has been largely overlooked by existing methods. We introduce Situational Graphs 2.0 (*S-Graphs 2.0*), that effectively uses the hierarchical structure of indoor scenes for efficient data management and optimization. Our approach builds a four-layer situational graph comprising *Keyframes*, *Walls*, *Rooms*, and *Floors*. Our first contribution lies in the front-end, which includes a floor detection module capable of identifying stairways and assigning floor-level semantic relations to the underlying layers (*Keyframes*, *Walls*, and *Rooms*). Floor-level semantics allows us to propose a floor-based loop closure strategy, that effectively rejects false positive closures that typically appear due to aliasing between different floors of a building. Our second novelty lies in leveraging our representation hierarchy in the optimization. Our proposal consists of: (1) local optimization over a window of recent keyframes and their connected components across the four representation layers, (2) floor-level global optimization, which focuses only on keyframes and their connections within the current floor during loop closures, and (3) room-level local optimization, marginalizing redundant keyframes that share observations within the room, which reduces the computational footprint. We validate our algorithm extensively in different real multi-floor environments. Our approach shows state-of-the-art accuracy metrics in large-scale multi-floor environments, estimating hierarchical representations up to 10× faster, in average, than competing baselines. Our code is open-sourced at: [https://github.com/snt-arg/lidar\\_situational\\_graphs](https://github.com/snt-arg/lidar_situational_graphs)

**Index Terms**—SLAM, Localization, Mapping

## I. INTRODUCTION

Simultaneous Localization and Mapping (SLAM), that has typically focused on geometric aspects, should also enable a robot with a high-level understanding of its environment [1]. The combination of 3D scene graphs with traditional SLAM graphs have recently sparked great interest given its compact and intuitive approach to representing an environment as a

Manuscript received: February, 28, 2025; Accepted: September, 19, 2025.

This paper was recommended for publication by Editor Sven Behnke upon evaluation of the Associate Editor and Reviewers' comments.

\*This research was funded in whole, or in part, by the Luxembourg National Research Fund (FNR), DEUS Project, ref. C22/IS/17387634/DEUS. For the purpose of open access, and in fulfillment of the obligations arising from the grant agreement, the author has applied a Creative Commons Attribution 4.0 International (CC BY 4.0) license to any Author Accepted Manuscript version arising from this submission.

<sup>1</sup>Authors are with the Automation and Robotics Research Group, Interdisciplinary Centre for Security, Reliability and Trust, University of Luxembourg. Holger Voos is also associated with the Faculty of Science, Technology and Medicine, University of Luxembourg, Luxembourg. {hriday.bavle, joseluis.sanchezlopez, muhammad.shaheer, holger.voos}@uni.lu

<sup>2</sup>Author is with I3A, Universidad de Zaragoza, Spain [jcivera@unizar.es](mailto:jcivera@unizar.es)

Digital Object Identifier (DOI): see top of this page.

©2026 IEEE

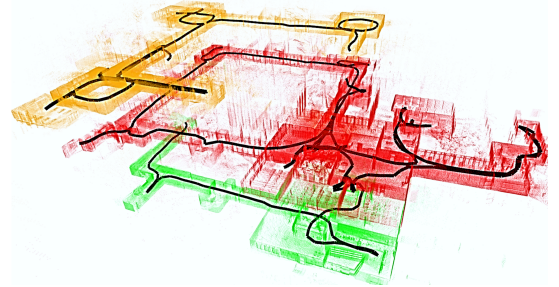


Fig. 1: Three-story map generated by our *S-Graphs 2.0*. Floor levels are color-coded in green, red, and orange, while the estimated robot trajectory is shown in black.

hierarchy of layers [2], [3]. In this direction, Situational Graphs (*S-Graphs*) [4] model the environment as a four-layered optimizable graph composed of *Keyframes*, *Walls*, *Rooms* and *Floors*. As the robot explores large environments, however, the size of the graph grows, along with its computational footprint. Essentially, while *S-Graphs* and 3D scene graphs are motivated by the need of higher-level representations, none of them have yet tailored their optimization to the specifics of these hierarchical representations. Leveraging them will result in a lower computational footprint, enabling real-time graph optimization for larger maps, which will result finally in enhanced accuracy. In this paper we present Situational Graphs 2.0 (*S-Graphs 2.0*), that builds over the work of [4] and contributes with two novel approaches that improve its robustness and computational footprint. Our first novelty consists on proposing a floor detection module capable of segmenting different floor levels and the stairways that connect them. This floor-level information is incorporated in all the rooms, walls and keyframes layers of the graph ensuring robust localization and loop closures, even across similarly looking areas on different floors. Our second novelty is the proposal of an optimization strategy that leverages the hierarchical structure of the graph to reduce its computational complexity, while maintaining pose and map accuracy. Specifically, as a robot explores an environment at a given floor-level, it estimates a hierarchical graph by local optimization over a subset of the last  $n$  keyframes and its connected layers. For loop closures at a given floor-level, we run a floor-level global optimization, optimizing only the subgraph of that floor-level. Additionally, on detection of a room candidate bounded by four walls, we exploit the *room-wall* hierarchy to perform room-level local optimization to marginalize keyframes and its connections removing redundant keyframes observing the same room. To summarize our main contributions are:

- *S-Graphs 2.0*, a hierarchical SLAM leveraging semantic

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

relations for efficient management and optimization of robot poses and map elements.

- Floor-level loop closure for robust localization in similar-looking environments at different levels.
- Floor- and room-level hierarchical optimization strategies, marginalizing out unnecessary entities while maintaining the map accuracy.

## II. RELATED WORKS

### A. Simultaneous Localization and Mapping

**Metric-Semantic SLAM.** The most referenced LiDAR SLAM pipelines include LOAM [5], LIO-SAM [6], BALM [7], Fast-LIO [8], [9]. All the above techniques utilize low-level geometric features in the environment to estimate the pose and map of the robot, due to which such techniques can be limited in their accuracy when exploring large and complex indoor environments. Additionally several other SLAM techniques exist in the literature that utilize high-level semantic features additional to geometric features to improve the environment understanding and the map and pose accuracy. Some of such techniques include LeGO-LOAM [10], SA-LOAM [11], SegMap [12], SUMA++ [13]. LeGO-LOAM [10] utilizes different planar semantics in the environment like ground plane to improve the map and pose estimate. While SA-LOAM [11] utilizes high-level semantics to improve the loop closure accuracy of the underlying metric SLAM, SegMap [12] utilizes learned high-level descriptors from the environment to perform robust localization with respect to the high-level descriptors. SUMA++ [13] completely segments the environment in different semantic features to perform object level semantic SLAM, removing dynamic entities from the scene.

However, one of the major limitations of the metric-semantic SLAM techniques is that they do not exploit the different semantic entities to perform a better map management strategies for improved optimization and map accuracy for large scale environments. Most of these techniques either clear major map elements as the map size increases to maintain real-time performance or do not provide a real-time analysis of the underlying SLAM when managing large scale maps.

**Hierarchical SLAM.** To overcome the inherent problem of improving the computation with increasing map size, works like [14], [15] exploit the methods to compress the graph into different sub-graphs to maintain real-time performance without the loss of map and pose accuracy. [14] present a technique grouping nodes into different sub-graphs based on a simple distance based criteria. While [15] present an information-theoretic approach for factor graph compression where laser scans measurements and their corresponding robot poses are removed such that the expected loss of information with respect to the current map is minimized. In [16] authors present a hierarchical continuous time SLAM algorithm dividing it into local sub-graphs aggregating measurements from 3D LiDAR, the generated sub-graphs are aligned with each other using local surfel-based registration techniques. In the above technique authors also use distance-based heuristic to create different sub-graphs for the optimization problem. [17] propose a local and global hierarchical optimization technique using sub-map strategies similar to [18]. They generate local sub-maps at given distance intervals performing local optimization and then

connecting different local sub-maps using global registration to perform efficient global optimization. The authors in [17] also choose a heuristic based strategy to generate the local and global maps.

Although hierarchical SLAM shows efficient optimization of the graph for real-time performance, currently most of the techniques rely on time-distance based heuristics to formulate and optimize the hierarchical graph.

### B. 3D Scene Graphs

Recently, 3D scene graphs have shown great potential in representing the environment in a more meaningful and compact manner [19]–[23]. Additionally methods such as, [2], [3], [4] tightly couple the 3D scene graph with SLAM graphs exploiting the hierarchy in the environment to improve the final pose and map accuracy. [2] focuses on the generation of a 3D scene graph mainly using RGB-D cameras, incorporating different hierarchies of objects, places, rooms and buildings, furthermore it utilizes the hierarchy to perform a top-down and bottom-up search improving the search of loop closure candidates, thus improving the final map accuracy. [3] presents a 3D scene graph for the outdoor environment, dividing it into hierarchies such as lanes, landmarks, intersections, and environment. They factor different connections of the hierarchical graph as a factor graph, continuously optimizing the pose and map. [4], [24] represent an indoor scene as a four-layered optimizable factor graph dividing it into layers of keyframes, walls, rooms, and floors, simultaneously optimizing all the layers to obtain improved map and pose accuracy.

However, most of the works based on 3D scene graphs only focus on generating the hierarchical representation of the environment and do not exploit the intuitive nature of these hierarchical graphs to perform enhanced management/optimization of the different map elements which eventually leads to the problem of increased computation with the increase in the size of the explored scene.

Thus in this work, we apply the concept of hierarchical SLAM to 3D scene graphs. The 3D scene graph structure allows to generate a meaningful hierarchical SLAM graph instead of using distance-time based heuristics. This hierarchical SLAM graph improves the management of generated map elements to scale for large scale environments, while maintaining the pose and map accuracy with real-time performance.

## III. OVERVIEW

### A. Background

**Situational Graphs.** We build on S-Graphs+ [4], that estimates a layered graph representation of the environment from 3D LiDAR. The four layers of the graph, illustrated in Fig. 2, are as follows. The **Keyframes Layer** contains robot poses at different distance-time intervals. The **Walls Layer** represents the walls in the environment as planar surfaces connected to the keyframes from which they were observed. The **Rooms Layer** models two-wall and four-wall rooms, connecting in this manner the previous walls layer. Finally, the **Floors Layer** represents the different floors in the building, connecting them with the underlying rooms layer.

### B. System Architecture

Fig. 2 shows the complete architecture of our approach. The front-end implements plane segmentation as in [4] and room

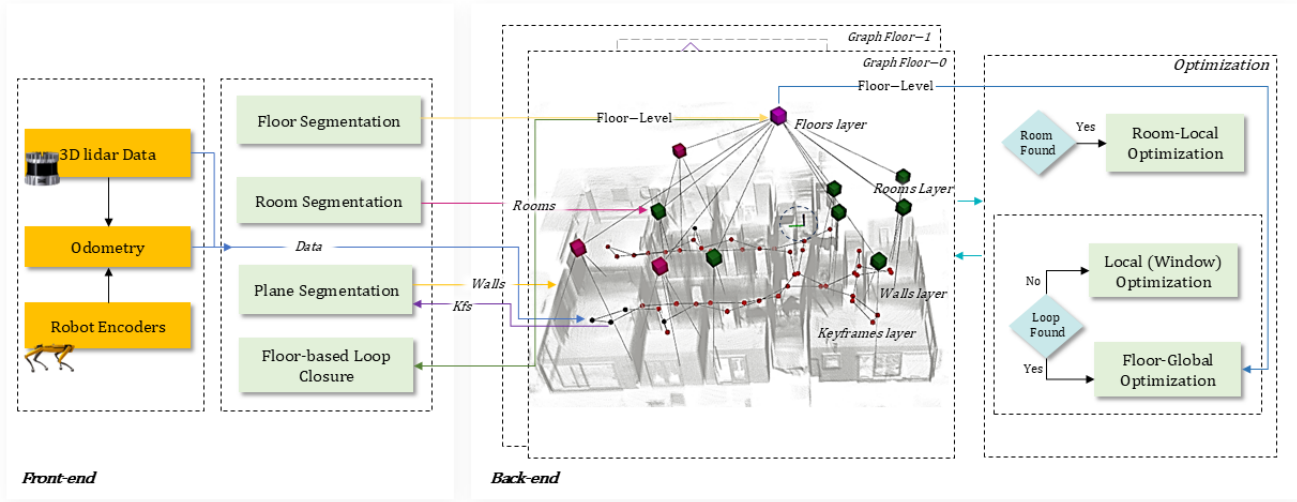


Fig. 2: **System Architecture.** The inputs to our method are the 3D LiDAR data along with the odometry measurements. It contains different modules in the front-end modules for generating the four-layered hierarchical graph and organizing it into floor-level. A back-end module which exploits the hierarchy in the graph to apply different optimization strategies.

segmentation as in [25], with the addition of a floor segmentation module able to detect changes in the floor levels by measuring changes in the slope of the trajectory of keyframes. In practice, floor segmentation will be highly relevant to perform loop closure only within the same floor. This is a reasonable strategy, as local sensor readings are commonly limited to a single floor and connecting staircases. Additionally, it removes a high rate of false positives, caused by aliasing between floor levels.

Our back-end of the proposed approach acts on a four-layered optimizable factor graph, as in [4], but adds several optimization strategies that leverage the hierarchy in the graph. Specifically, the optimization is performed at three levels. A local optimization acts over a window of  $n$  keyframes every time new nodes or edges are added to the factor graph. A floor-level global optimization runs every time a loop closure candidate is found between two keyframes. Unlike typical global optimization threads in SLAM, that operate on all keyframes, our floor-level optimization focuses on a subset of the graph containing only the current floor level. It optimizes the keyframes and all the connected layers in the current floor level, and previously connected floor levels are only optimized when the current floor is revisited. A room-level local optimization marginalizes redundant keyframes and its connections observing the same room. This hierarchical optimizations minimizes computational overhead while maintaining the map and pose accuracy across connected floor levels. The global state  $\mathbf{s}$  that we estimate is as follows

$$\mathbf{s} = \begin{bmatrix} {}^M \boldsymbol{\xi}_1, \dots, {}^M \boldsymbol{\xi}_F, \\ {}^M \mathbf{x}_{R_1 \xi_1}, \dots, {}^M \mathbf{x}_{R_T \xi_F}, {}^M \boldsymbol{\pi}_{1 \xi_1}, \dots, {}^M \boldsymbol{\pi}_{P \xi_F}, \\ {}^M \boldsymbol{\rho}_{1 \xi_1}, \dots, {}^M \boldsymbol{\rho}_{S \xi_F}, {}^M \boldsymbol{\kappa}_{1 \xi_1}, \dots, {}^M \boldsymbol{\kappa}_{K \xi_F}, \\ {}^M \mathbf{x}_O \end{bmatrix}^\top \quad (1)$$

where  ${}^M \boldsymbol{\xi}_f \in \mathbb{R}^3$ ,  $f \in \{1, \dots, F\}$  are the  $F$  floors levels. All the nodes in the state at a given time contain the semantic of the floor-level it belongs to.  ${}^M \mathbf{x}_{R_t \xi_f} \in SE(3)$ ,  $t \in \{1, \dots, T\}$

are the robot poses belonging to a given floor level.  ${}^M \boldsymbol{\pi}_{i \xi_f} \in \mathbb{R}^3$ ,  $i \in \{1, \dots, P\}$  are the plane parameters of the  $P$  wall planes belonging to a given floor level.  ${}^M \boldsymbol{\rho}_{j \xi_f} \in \mathbb{R}^3$ ,  $j \in \{1, \dots, S\}$  contains the parameters of the  $S$  four-wall rooms and  ${}^M \boldsymbol{\kappa}_k \in \mathbb{R}^3$ ,  $k \in \{1, \dots, K\}$  the parameters of the  $K$  two-wall rooms, all belonging to a given floor-level  $\xi_f$ . Finally,  ${}^M \mathbf{x}_O$  is the estimated drift between the map frame and the odometry frame.

## IV. FRONT-END

### A. Floor Segmentation

**Floor Center.** The floor segmentation module extracts the center of a current floor level from the largest wall planes currently extracted in a scene. Each time the pipeline is run, it creates a default floor node with the center placed at the origin of frame  $M$ . It then utilizes the information from all walls at the current floor level  ${}^M \boldsymbol{\xi}_f$  to create a sub-category of wall planes  ${}^M \boldsymbol{\Pi}_{i \xi_f}$  ( ${}^M \boldsymbol{\Pi}$  represents a set of  ${}^M \boldsymbol{\pi}$  planes) where  $i = \{1, \dots, T\}$ . The walls sub-categories consist of  $x$  vertical planar surfaces and  $y$  vertical planar surfaces based on their normal orientation. After receiving a plane set, it computes the widths  $w_x$  between all  $x$ -direction planes and similarly  $w_y$  for  $y$ -direction planes as in [4].

The wall plane set with the largest  $w_x$  and  $w_y$  is the chosen candidate for the current floor level. These planar pairs in both  $x$  and  $y$  direction undergo an additional dot prod-

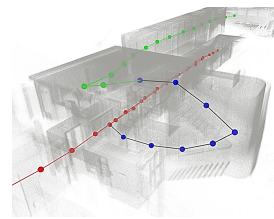


Fig. 3: **Stairway Detection**, with the keyframes corresponding to stairways as blue dots, and those corresponding to different floor levels as red and green dots.

**IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.**

uct check between their corresponding normal orientations,  $|\mathbf{n}_{x_{a_1}} \cdot \mathbf{n}_{x_{b_1}}| < t_n$  and  $|\mathbf{n}_{y_{a_1}} \cdot \mathbf{n}_{y_{b_1}}| < t_n$ , to remove wall planes originating outside the building structure. The floor segmentation then computes the floor center node using the obtained wall plane candidates as explained in [4].

**Stairway Detection.** The stairway detection module is responsible for identifying transitions between floor levels and associating corresponding keyframes with the detected stairs. We follow a process similar to [26] but extend it to our use case for detecting multiple floors and associating keyframes belonging to stairways and different floor-levels (see Fig. 3).

Our process relies on the analysis of sequential keyframes captured during navigation, particularly focusing on their vertical displacement. It begins by maintaining a queue of keyframes that are sequentially analyzed to compute the slope of their vertical trajectory. The slope is calculated using linear regression on the height values extracted from the poses of the keyframes. This slope quantifies the rate of vertical displacement and serves as the primary indicator for stairway traversal with its corresponding sign distinguishing upward or downward movements. A slope above threshold  $\tau_s$  is identified as the start of the stairway, after which all the keyframes are added into stairway keyframe sequence. When the gradient drops below  $\tau_s$  we mark end of that stairway, with all the keyframes  $k_s$  within the sequence being part of the stairway sequence. A new floor node is then created with its height set using the height of the final keyframe in the sequence and all subsequent keyframes assigned to the new floor-level.

### B. Floor-based Loop Closure

One of the challenges for SLAM in large environments with multiple floors is the similarity in the geometry and visual appearance of rooms and environments in same areas of the different floors. This aliasing cause conventional geometry- or semantic-based loop closure algorithms to fail, leading to incorrect associations between keyframes from different floors. Such errors can severely degrade the accuracy of the estimated maps.

To address this issue, we leverage the inherent hierarchical structure of our graph in combination with the floor segmentation module to incorporate floor-level information across all layers. Specifically, given a complete set of keyframes  ${}^M\mathbf{K}_i$  we create a subset of keyframes  ${}^M\mathbf{K}_f, \forall f \in {}^M\xi_f$  belonging to the current floor-level  ${}^M\xi_f$ , and perform a scan matching-based loop-closure algorithm [4] for this subset, ensuring that keyframes from different floors are excluded from matching.

To avoid false positives in transitional scenarios, such as climbing up/down of the stairs, our floor-based loop closure is temporarily disabled in such areas, as LiDAR measurements might capture regions that may not distinctly belong to any single floor. Floor-based loop closure is resumed once there is a full transition to a new floor-level.

### C. Room Keyframe Segmentation

In order to perform room-level local optimization (explained in Section V) we need to identify keyframes which are bounded within a four-wall room. We exploit the hierarchy within a room and its corresponding four-walls to identify the keyframes set  ${}^M\mathbf{K}_{\rho_i}$  belonging to a room  ${}^M\rho_i$ . To compute if a keyframe lies within a room, we first compute the vector  $\mathbf{v}_d = \mathbf{p} - \mathbf{q}_i$ , where  $\mathbf{p}$  is the translation component of the

keyframe and  $\mathbf{q}_i$  is a randomly selected point from the  $i$ -th plane of the room, where  $i \in \{1, 2, 3, 4\}$  corresponds to each of the four planes belonging to the room. We then evaluate the dot product of  $\delta = \mathbf{n}_i \cdot \mathbf{v}_d$ , where  $\mathbf{n}_i$  is the normal orientation of the plane. Keyframe positions with  $\delta$  positive are considered to be bounded within the walls of the room.

## V. BACK-END

The global state of the robot defined in Section III-B is optimized in three different stages exploiting the hierarchy of the graph. We mainly detail the optimization strategies, where relies the novelty of the paper. Details on the formulation of the cost function can be found in [4].

### A. Local Optimization

As a robot navigates through a scene we generate specific keyframe nodes and its observations thus incorporating new nodes and its edges within the graph. Each time the graph expands, we perform local optimization on a subset of the complete graph called the local graph, which consists of a sliding window containing the most recent  ${}^M\mathbf{K}_n$  keyframes. In order to incorporate nodes and edges belonging to other hierarchical layers of the graph into local graph, we first check the connections between the  ${}^M\mathbf{K}_n$  keyframes and the wall layer, incorporating all connected wall nodes  ${}^M\Pi_p$  and their corresponding edges. For all wall nodes  ${}^M\Pi_p$  within the optimization window, we check whether the entire set or a subset belongs to two-wall rooms  ${}^M\rho_s$  or four-wall rooms  ${}^M\kappa_k$ . If such connections exist, we incorporate these room nodes into the local optimization.

Furthermore, we incorporate the corresponding floor node  ${}^M\xi_f$  that the robot is currently navigating into the local graph. To maintain graph consistency during optimization, we adopt the strategy proposed in [27], fixing keyframes that lie outside the current optimization window but observe walls  ${}^M\Pi_p$  within the local window. The state  $s_l$  to be optimized is defined as:

$$\hat{s}_l = \arg \min_{s_l} (c_{k_n}, c_{k_g}, c_{\pi_p}, c_{\rho_s}, c_{\xi_f}) \quad (2)$$

where  $c_{k_n}, c_{\pi_p}, c_{\rho_s}$  are the cost functions for the  $n, p, s$  unfixed keyframes, walls and rooms respectively,  $c_{k_g}$  is the cost of the  $g$  fixed keyframes and  $c_{\xi_f}$  is the cost of the  $f^{th}$  floor node currently being explored. Fig. 4 shows the local optimization strategy, with orange color keyframes along

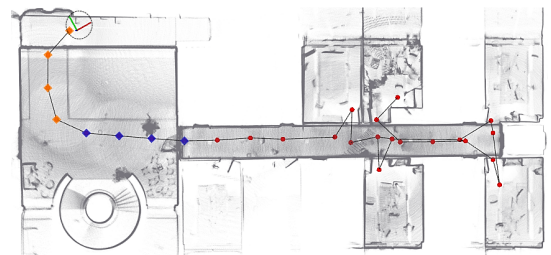
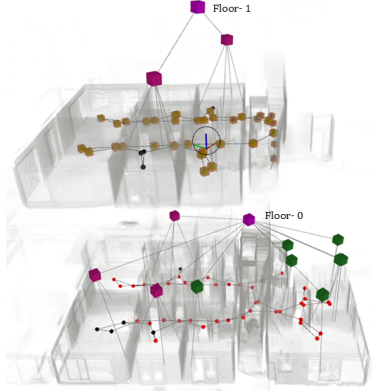


Fig. 4: **Local Optimization.** Orange- and blue-colored keyframes, along with their connected layers, are included in the local optimization, with blue keyframes being fixed. Red keyframes are not incorporated in the optimization, as they are outside the optimization window.

with the connected layers being optimized while the fixed keyframes are highlighted in blue.



**Fig. 5: Floor-level Global Optimization.** Floor-1 keyframes with orange colored boxes along with the connected wall, room and floor nodes are included in the floor-level global optimization on detection of a loop closure candidate on floor-1. Floor-0 (lower level) maintains its own independent pose graph structure with keyframes (orange spheres) with connected rooms (pink and green cubes) and walls. These elements are temporarily excluded from the Floor-1 optimization to reduce computational complexity and prevent cross-floor interference.

### B. Floor-level Global Optimization

Floor-level global optimization is performed in mainly two situations, specifically, when a loop closure is found between the keyframes of a floor-level, and when a duplicate wall is found when a room is re-detected as explained in [4]. Our main motivation for this strategy is as follows. When a new floor-level is explored, and a loop closure is found between two keyframes of such floor-level, this loop closure does not affect the keyframes or other connected layers of the preceding floor-levels. The same reasoning applies when duplicate walls are found for a detected room. When executing floor-level global optimization for floor-level  $f$ , we create a subset of the graph containing floor node  $\xi_f$ , keyframe nodes  $K_{n_{\xi_f}}$  and its edges with the neighboring keyframes, then incorporate all wall nodes  $\pi_{p_{\xi_f}}$  with their edges to keyframe nodes  $K_{n_{\xi_f}}$ . Floor-level global optimization incorporates all previously identified loop closure (relative pose) edges between the keyframes of that floor-level. We also incorporate four-wall  $\rho_{n_{\xi_f}}$  and two-wall room nodes  $\kappa_{n_{\xi_f}}$  with their edges to the underlying walls, finally also incorporating the edges between these rooms and the current floor node. To maintain consistency of the graph during optimization, we fix a keyframe which can either be the initial keyframe when the robot starts navigating, or it can be the last keyframe from the previous visited floor-level, this assures that the current floor-level map estimates do not diverge away from the previous floor-level map (See Fig. 5). The state  $s_g$  using floor-level global optimization is optimized

as:

$$\hat{s}_g = \arg \min_{s_g} (c_{K_{n_{\xi_f}}}, c_{K_{g_{\xi_f}}}, c_{K_{l_{\xi_f}}}, c_{\pi_{p_{\xi_f}}}, c_{\rho_{s_{\xi_f}}}, c_{\kappa_{k_{\xi_f}}}, c_{\xi_f}) \quad (3)$$

where  $c_{K_{n_{\xi_f}}}, c_{K_{g_{\xi_f}}}, c_{K_{l_{\xi_f}}}$  are the cost functions for the  $n$  unfixed,  $g$  fixed,  $l$  loop closure keyframes.  $c_{\pi_{p_{\xi_f}}}, c_{\rho_{s_{\xi_f}}}, c_{\kappa_{k_{\xi_f}}}$  are the costs of  $p$ ,  $s$  and  $k$  walls, two-wall and four-wall rooms respectively.  $c_{\xi_f}$  is the cost of the current floor node being explored.

As a relevant effect of floor-level global optimization, if a robot revisits a given floor-level and finds a suitable loop closure candidate, the floor-level global optimization incorporates all the nodes and edges not only from the current floor-level but also from all the previous visited ones. This approach also enables loop closures spanning multiple floors through staircases, ensuring comprehensive error correction across the entire multi-floor structure. This ensures the correction of all the accumulated errors from the revisited floor-level, as well as the previously visited ones.

### C. Room-level Local Optimization

Room-level local optimization is performed every time a four-wall room (bounded by four walls) is detected by the room segmentation module and room keyframes belonging to that particular room are identified (Section. IV-C). The room-level local optimization creates a graph subset containing the room node  $\rho_i$  along with the walls nodes  $\pi_{p_{\rho_i}}$  lying within the room along with the connected keyframes  $K_{n_{\rho_i}}$ . Additionally as in local and floor-level global optimization to maintain the consistency of the graph we incorporate fixed set of keyframes  $K_{g_{\rho_i}}$  which are outside the room but observe the walls of the room. The state  $s_r$  during room-level local optimization is given as:

$$\hat{s}_r = \arg \min_{s_r} (c_{k_{n_{\rho_i}}}, c_{k_{g_{\rho_i}}}, c_{\pi_{p_{\rho_i}}}, c_{\rho_i}) \quad (4)$$

Where  $c_{k_{n_{\rho_i}}}, c_{k_{g_{\rho_i}}}$  is the cost for  $n$  unfixed and  $g$  fixed keyframes and  $c_{\pi_{p_{\rho_i}}}$  and  $c_{\rho_i}$  are the costs related to  $p$  walls and the  $i$ -th room.

The main strategy of the room-level local optimization is to marginalize redundant keyframe nodes and its edges that observe the same room structure. Thus, after a room-level local optimization is performed, all the keyframe nodes and its corresponding edges except the first keyframe node/edges are marginalized out from the main global graph. Marginalization leads to a generation of a disconnected graph between keyframes. These disconnections are between the marginalized keyframes. To obtain a connected global graph, we incrementally check for all the edges  $e_{i,n}$ , between the  $i$ -th marginalized keyframe  $K_i$  with  $n$ -th non-marginalized neighbor  $K_n$ , and connect the closest non-marginalized neighbors  $K_{i-1}$  and  $K_n$  with a new edge  $e_f$  (See Fig. 6). The information matrix of the new edge is the summation of the information matrices of the edges removed in  $e_{i,n}$ :

$$\Omega_{e_f} = \sum_{(K_i, K_n) \in \mathcal{E}_m} \Omega_{e_{i,n}} \quad (5)$$

where  $\Omega_{e_{i,n}}$  is the information matrix of the original edge  $e_{i,n}$

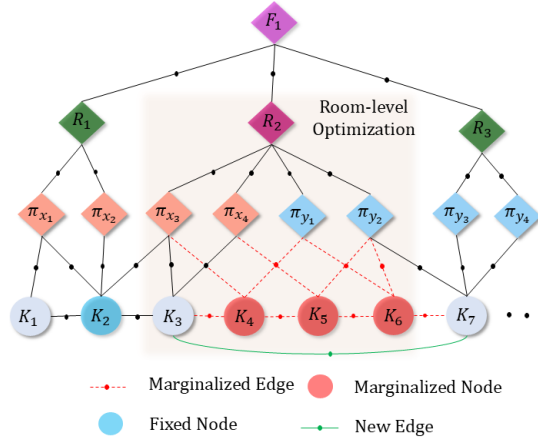


Fig. 6: **Room-level Local Optimization.** Overview of the compressed graph after room-level local optimization. Orange box shows nodes used for room-level optimization, while the dotted red box indicates keyframe marginalization post-optimization. The green line shows the new edge connecting the first keyframe of the room with the first keyframe outside the room.

and  $\Omega_{e_f}$  is the resulting information matrix for the new edge  $e_f = (K_{i-1}, K_n)$ . The new edge  $e_f$  is added back to the global graph edges:

$$\mathcal{E}_{global} \leftarrow \mathcal{E}_{global} \cup \{(K_{i-1}, K_n, \Omega_{e_f})\} \quad (6)$$

The room-level local optimization is always performed after the execution of at least one of the local or floor-level global optimization.

## VI. EXPERIMENTAL RESULTS

### A. Methodology

We validate our algorithm on different real indoor environments (single-floor and multi-floor) from university buildings to construction sites, validating it with different state-of-the-art SLAM algorithms, including the baseline *S-Graphs+* [4]. Our datasets are recorded using Velodyne VLP-16 or Ouster OS-1 64 3D LiDAR, with odometry from robot encoders or FAST-LIO2 [9]. These sensors demonstrate that our algorithm is sensor-agnostic. Despite differing beam counts, both sensors are downsampled to uniform resolution for processing. Dynamic objects such as humans are present in many of these datasets; while handling dynamic objects falls outside the scope of this paper, the reported results reflect their presence as we neither explicitly include nor exclude them from our analysis.

**Single-Floor Dataset.** The single-floor dataset consists of small-scale indoor environments with the robot navigating around one single floor. The main motivation for these experiments is to fairly validate the accuracy and cost reduction of our *S-Graphs 2.0* against baselines, as most baselines accumulate significant drift or incur in false loop closures for multi-floor scenes. The first two experiments, *C1F1* and *C1F2*, are performed on two different floors of a construction site. Additionally, *C2F0*, *C2F1*, and *C2F2* consist of three different floors of an ongoing construction site combining four individual houses. *C3F1*, and *C3F2* are two different floors

two combined houses. For validation, due to the absence of ground truth trajectory metrics, we use ground truth point clouds, extracted from available Building Information Models (BIM), to report pointcloud RMSE. Furthermore, we also report the computational time of our approach against the most accurate baseline *S-Graphs+* [4].

**Multi-Floor Dataset.** The multi-floor dataset is aimed at validating the accuracy of our algorithm in the presence of larger trajectories covering multiple floors. It consists of five datasets, *BE1* and *BE2* are university buildings where the robot navigates the three and two floors, respectively. *LC1* is a university building with long corridors and a robot traversing two floors. *CS1* and *CS2* are buildings under construction with a robot traversing all three floors. Due to the absence of ground truth, to validate the accuracy of each approach, we use the Mean Map Entropy (MME) presented in [28] to measure the sharpness of the generated map. Additionally, to assess algorithmic accuracy in multi-floor segmentation, we evaluate the Intersection over Union (IoU) between the histogram of z-value occurrences per floor level in the generated map and the percentage of ground truth time spent per floor level. In this evaluation, we recorded the time the robot spent on each floor level, which was subsequently converted into percentage data for that floor. We also documented the maximum and minimum heights for each floor level, establishing the ground truth bins for the datasets. We then utilized the z-values from the 3D maps generated by our algorithms and baselines to compute the IoU of these bins against the ground truth bins. While IoU is an indirect metric, it captures if the SLAM algorithm correctly tracks the robot's position. The vertical distribution of mapped points should match the time spent on each floor. When the algorithm incorrectly identifies floors, points are mapped at incorrect heights, resulting in clear mismatches in the z-distribution. Thus, high IoU indicates good floor-level tracking throughout the mapping process, providing meaningful validation despite the lack of ground truth poses.

### B. Results and Discussions

**Single-Floor Dataset.** The results of the single-floor experiments, in Tab. I, show that our *S-Graphs 2.0* is able to provide final map accuracy similar to the second best baseline *S-Graphs+* with an average accuracy improvement of 0.48%. However, the benefits of the hierarchical optimization of our *S-Graphs 2.0* clearly stand out in Tab. V, where it has an average cost reduction of 79.87% to the closest baseline. The benefits of the hierarchical optimization become even more clear for longer sequences. See, for example, how in *C2F2*, *C3F2* there is slight decrease in the accuracy with respect to *S-Graphs+* (2.27% and 2.67% respectively), but a large reduction the computation time (85.56% and 82.66% respectively). Summing up, in these single-floor datasets, we show that the optimization strategy in *S-Graphs 2.0* maintains the estimation accuracy while significantly reducing the computational cost. We also compared our algorithm and other baselines on the publicly available Hilti SLAM datasets [29]. For fair comparison, we use the same odometry source *fast lio* [9] for all. Tab. II reports the ATE for the easy (E04, E05) and medium (E06, E14) difficulty categories. We used Our *S-Graphs 2.0* consistently outperforms the baseline in all sequences. Note that none of

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

the methods converged on the hard category sequences due to their extreme difficulty.

TABLE I: Point cloud RMSE [cm] for our single-floor dataset. Best results are **boldfaced**, second best are underlined.

Method	Point Cloud RMSE [cm] ↓							Avg
	C1F1	C1F2	C2F0	C2F1	C2F2	C3F1	C3F2	
HDL-SLAM [30]	33.5	19.8	18.5	21.1	19.5	22.9	19.4	22.1
ALOAM [5]	52.6	33.6	34.1	45.1	29.9	36.5	43.4	39.3
SCA-LOAM [31]	45.6	26.1	22.5	25.4	20.2	26.9	19.2	26.6
BALM [7]	34.9	22.5	18.2	19.7	99.0	26.9	22.6	34.8
S-Graphs+ [4]	<u>32.9</u>	<b>18.9</b>	<b>16.9</b>	<b>18.9</b>	<b>17.6</b>	22.3	<b>18.7</b>	<u>20.9</u>
<i>Ours</i>	<b>31.3</b>	<u>19.0</u>	<u>17.0</u>	<u>19.3</u>	<u>18.0</u>	<b>22.0</b>	<u>19.2</u>	<b>20.8</b>

TABLE II: Absolute Trajectory Error (ATE) in *cm* for Hilti datasets [29]. Best results are **boldfaced**, second best are underlined.

Method	Absolute Trajectory Error ↓				Avg
	E04	E05	E06	E14	
HDL-SLAM [30]	21.9	15.4	120.4	14.2	42.9
ALOAM [5]	28.2	22.5	<u>130.1</u>	24.9	51.4
SCA-LOAM [31]	28.6	23.1	130.4	25.3	51.9
BALM [7]	27.9	18.8	129.4	16.7	48.2
S-Graphs+ [4]	<u>20.5</u>	<u>13.8</u>	123.2	<u>13.1</u>	<u>42.6</u>
<i>Ours</i>	<b>19.8</b>	<b>13.0</b>	<b>50.3</b>	<b>11.5</b>	<b>23.6</b>

TABLE III: Point cloud Mean Map Entropy (MME) for our multi floor dataset. Best results are **boldfaced**, second best are underlined.

Method	Point Cloud MME ↓					Avg
	BE1	BE2	LC1	CS1	CS2	
HDL-SLAM [30]	-1.23	-1.38	-1.31	-0.97	-1.14	-1.21
ALOAM [5]	<u>-1.37</u>	-1.23	-1.32	-0.73	-1.07	-1.14
SCA-LOAM [31]	-0.85	-1.04	-0.96	-0.66	-1.04	-0.91
BALM [7]	-1.35	-1.27	-1.62	-0.77	<b>-1.28</b>	-1.26
S-Graphs+ [4]	-1.23	-1.28	<u>-1.37</u>	<u>-1.06</u>	-1.11	-1.21
<i>Ours</i>	<b>-1.58</b>	<b>-1.41</b>	<b>-1.67</b>	<b>-1.24</b>	<b>-1.28</b>	<b>-1.44</b>

TABLE IV: Intersection over Union (IoU) for our multi-floor dataset. Best results are **boldfaced**, second best underlined.

Method	IoU ↑					Avg
	BE1	BE2	LC1	CS1	CS2	
HDL-SLAM [30]	0.76	<b>0.90</b>	0.55	0.37	0.35	0.59
ALOAM [5]	0.50	0.78	0.69	0.69	0.31	0.59
SCA-LOAM [31]	0.56	0.81	0.78	0.51	0.31	0.59
BALM [7]	0.67	0.71	<u>0.77</u>	0.79	0.80	0.75
S-Graphs+ [4]	<u>0.83</u>	0.72	0.46	0.33	0.30	0.53
<i>Ours</i>	<b>0.90</b>	<b>0.91</b>	<b>0.93</b>	<b>0.90</b>	<b>0.91</b>	<b>0.91</b>

**Multi-Floor Dataset.** Tab. III shows the Mean Map Entropy (MME) for the multi-floor dataset. Our method outperforms all the baselines by an average of 25.67% when comparing the MME. Given the large area covered in these multi-floor experiments, all baselines fail at giving reasonable estimates, being unable to maintain the separation between the floor levels as well as accumulating significant drift. This can be assessed in Fig. 8, which shows the maps estimated by *S-Graphs 2.0* and *S-Graphs+* for *LC1* (Top row (a-b) respectively), and in one of the multifloor Hilti datasets [29], bottom row (c-d). As these environments are highly aliased between floors, *S-Graphs+*'s loop closure gives a high number of false positives

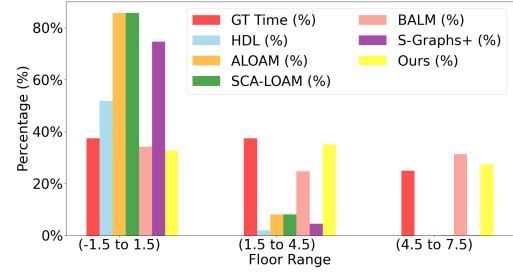


Fig. 7: **Floor Comparison.** CS2 dataset comparing the distribution of time spent per floor (ground truth) in percentage with the distribution of map-derived z-values converted to percentages, organized per floor.

TABLE V: Computation time in milliseconds (ms) for our method compared against the baseline *S-Graphs+*. Sequence lengths [s] are indicated for each dataset.

Dataset	Seq. Length [s]	Computation Time [ms]	
		S-Graphs+	Ours
C1F1	487	74.0	<b>36.0</b>
C1F2	657	106	<b>38.0</b>
C2F0	238	87.3	<b>4.00</b>
C2F1	672	169	<b>23.0</b>
C2F2	1044	263	<b>37.0</b>
C3F1	558	125	<b>8.00</b>
C3F2	999	173	<b>30.0</b>
BE1	1378	1106	<b>88.0</b>
BE2	1032	479	<b>101</b>
LC1	490	180	<b>21.0</b>
CS1	3000	1085	<b>10.0</b>
CS2	690	126	<b>13.0</b>
<b>Avg</b>	-	331.1	<b>34.08</b>

between levels. Our approach, given the floor segmentation, stairway detection, and floor-based loop closure, is able to maintain the accurate map while segmenting the keyframes and their connected layers based on the given floor-level. Fig. 9 shows qualitative results for CS2 demonstrating that our method is capable of segmenting the three floor-levels while maintaining a good map accuracy compared to *S-Graphs+* (which again performs inaccurate loop closures between floor levels) and BALM. Tab. IV and Fig. 7 also validate the proper floor segmentation by comparing the IoU percentage of  $z$  (height) values per floor-level from the maps generated by the algorithms versus the percentage ground truth time spend per floor-level. Observe that the IoU of our *S-Graphs 2.0* correlates perfectly with the ground truth time, *BALM* presents a weaker correlation and the rest of baselines offer very poor results.

Remarkably, as reported in Tab. V, the average reduction of computation time in multi-floor dataset of *S-Graphs 2.0* compared to *S-Graphs+* is 92.2%. As detailed before, this comes as a result of the hierarchical optimization and floor-level loop closure strategies of our *S-Graphs 2.0*.

## VII. CONCLUSION

In this work, we present *S-Graphs 2.0*, a multi-layered hierarchical SLAM for large, multi-floor indoor environments. Our approach leverages semantic relationships and a robust floor detection module to assign floor-level information, enabling effective floor-level global and room-level local optimizations as well as reliable loop closures across visually similar areas. We demonstrate a 19.48% increase in accuracy and a 92.2% reduction in computation time compared to the second-best

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

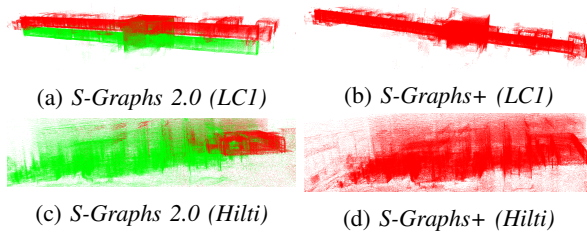


Fig. 8: **Qualitative Results** comparing *S-Graphs 2.0* and *S-Graphs+* Top row (a-b): multi-floor dataset *LC1*, bottom row (c-d): *Hilti* dataset [29]. Note how the two floor levels collapse for *S-Graphs+*, while our *S-Graphs 2.0* shows correct results.

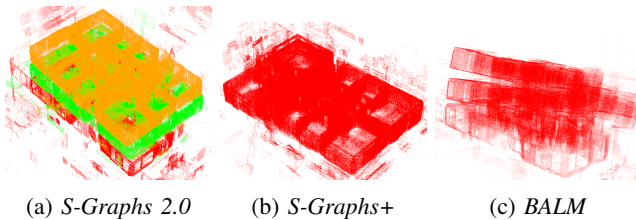


Fig. 9: **Qualitative Results** comparing *S-Graphs 2.0*, *S-Graphs+* and *BALM* in the multi-floor dataset *CS2*. Note the collapse of floor levels for *S-Graphs+*, deviations for *BALM* and accurate estimation results for our *S-Graphs 2.0*.

method. While our current floor detection relies on trajectory slope analysis and thus does not handle purely vertical transitions such as elevators, it effectively manages the more common case of stair-based floor transitions. Future work will integrate additional sensing modalities to handle elevator transitions, as well as further semantic-relational concepts to enhance floor-based loop closure and optimization.

#### REFERENCES

- [1] C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, I. Reid, and J. J. Leonard, "Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age," *IEEE Transactions on robotics*, vol. 32, no. 6, pp. 1309–1332, 2016.
- [2] N. Hughes, Y. Chang, S. Hu, R. Talak, R. Abdulhai, J. Strader, and L. Carlone, "Foundations of spatial perception for robotics: Hierarchical representations and real-time systems," *The International Journal of Robotics Research*, vol. 43, no. 10, pp. 1457–1505, 2024.
- [3] E. Greve, M. Büchner, N. Vödisch, W. Burgard, and A. Valada, "Collaborative dynamic 3d scene graphs for automated driving," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*, 2024, pp. 11 118–11 124.
- [4] H. Bavle, J. L. Sanchez-Lopez, M. Shaheer, J. Civera, and H. Voos, "S-graphs+: Real-time localization and mapping leveraging hierarchical representations," *IEEE Robotics and Automation Letters*, vol. 8, no. 8, pp. 4927–4934, 2023.
- [5] J. Zhang and S. Singh, "LOAM: Lidar Odometry and Mapping in Real-time," in *Robotics: Science and Systems*, 2014.
- [6] T. Shan, B. Englot, D. Meyers, W. Wang, C. Ratti, and D. Rus, "Lio-sam: Tightly-coupled lidar inertial odometry via smoothing and mapping," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020, pp. 5135–5142.
- [7] Z. Liu and F. Zhang, "Balm: Bundle adjustment for lidar mapping," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 3184–3191, 2021.
- [8] W. Xu and F. Zhang, "Fast-lio: A fast, robust lidar-inertial odometry package by tightly-coupled iterated kalman filter," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 3317–3324, 2021.
- [9] W. Xu, Y. Cai, D. He, J. Lin, and F. Zhang, "Fast-lio2: Fast direct lidar-inertial odometry," *IEEE Transactions on Robotics*, vol. 38, no. 4, pp. 2053–2073, 2022.
- [10] T. Shan and B. Englot, "LeGO-LOAM: Lightweight and Ground-Optimized Lidar Odometry and Mapping on Variable Terrain," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018, pp. 4758–4765.
- [11] L. Li, X. Kong, X. Zhao, W. Li, F. Wen, H. Zhang, and Y. Liu, "Sa-loam: Semantic-aided lidar slam with loop closure," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, 2021, pp. 7627–7634.
- [12] R. Dubé, A. Cramariuc, D. Dugas, H. Sommer, M. Dymczyk, J. Nieto, R. Siegwart, and C. Cadena, "SegMap: Segment-based mapping and localization using data-driven descriptors," *The International Journal of Robotics Research*, vol. 39, no. 2-3, pp. 339–355, jul 2019.
- [13] X. Chen, A. Milioto, E. Palazzolo, P. Giguère, J. Behley, and C. Stachniss, "Suma++: Efficient lidar-based semantic slam," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2019, pp. 4530–4537.
- [14] G. Grisetti, R. Kümmerle, C. Stachniss, U. Frese, and C. Hertzberg, "Hierarchical optimization on manifolds for online 2d and 3d mapping," in *2010 IEEE International Conference on Robotics and Automation*, 2010, pp. 273–278.
- [15] H. Kretzschmar and C. Stachniss, "Information-theoretic compression of pose graphs for laser-based slam," *The International Journal of Robotics Research*, vol. 31, no. 11, pp. 1219–1230, 2012.
- [16] D. Droschel and S. Behnke, "Efficient continuous-time slam for 3d lidar-based online mapping," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, 2018, pp. 5000–5007.
- [17] K. Koide, M. Yokozuka, S. Oishi, and A. Banno, "Globally consistent and tightly coupled 3d lidar inertial mapping," in *2022 International Conference on Robotics and Automation (ICRA)*, 2022, pp. 5622–5628.
- [18] W. Hess, D. Kohler, H. Rapp, and D. Andor, "Real-time loop closure in 2d lidar slam," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, 2016, pp. 1271–1278.
- [19] I. Armeni, Z.-Y. He, A. Zamir, J. Gwak, J. Malik, M. Fischer, and S. Savarese, "3d scene graph: A structure for unified semantics, 3d space, and camera," in *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019, pp. 5663–5672.
- [20] J. Wald, H. Dhano, N. Navab, and F. Tombari, "Learning 3d semantic scene graphs from 3d indoor reconstructions," in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 3960–3969.
- [21] U.-H. Kim, J.-M. Park, T.-j. Song, and J.-H. Kim, "3-d scene graph: A sparse and semantic representation of physical environments for intelligent agents," *IEEE Transactions on Cybernetics*, vol. 50, no. 12, pp. 4921–4933, 2020.
- [22] S.-C. Wu, J. Wald, K. Tateno, N. Navab, and F. Tombari, "Scenegrph-fusion: Incremental 3d scene graph prediction from rgb-d sequences," in *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021, pp. 7511–7521.
- [23] A. Rosinol, A. Gupta, M. Abate, J. Shi, and L. Carlone, "3d dynamic scene graphs: Actionable spatial perception with places, objects, and humans," *arXiv preprint arXiv:2002.06289*, 2020.
- [24] H. Bavle, J. L. Sanchez-Lopez, M. Shaheer, J. Civera, and H. Voos, "Situational graphs for robot navigation in structured indoor environments," *IEEE Robotics and Automation Letters*, vol. 7, no. 4, pp. 9107–9114, 2022.
- [25] J. A. Millan-Romera, H. Bavle, M. Shaheer, M. R. Oswald, H. Voos, and J. L. Sanchez-Lopez, "Learning high-level semantic-relational concepts for slam," in *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2024, pp. 9803–9810.
- [26] C. Kassab, M. Mattamala, L. Zhang, and M. Fallon, "Language-extended indoor slam (lexis): A versatile system for real-time visual scene understanding," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*, 2024, pp. 15 988–15 994.
- [27] C. Campos, R. Elvira, J. J. G. Rodríguez, J. M. M. Montiel, and J. D. Tardós, "Orb-slam3: An accurate open-source library for visual, visual-inertial, and multimap slam," *IEEE Transactions on Robotics*, vol. 37, no. 6, pp. 1874–1890, 2021.
- [28] D. Droschel and S. Behnke, "Efficient continuous-time slam for 3d lidar-based online mapping," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, 2018, pp. 5000–5007.
- [29] L. Zhang, M. Helmlinger, L. F. T. Fu, D. Wisth, M. Camurri, D. Scaramuzza, and M. Fallon, "Hilti-oxford dataset: A millimeter-accurate benchmark for simultaneous localization and mapping," *IEEE Robotics and Automation Letters*, vol. 8, no. 1, pp. 408–415, 2023.
- [30] K. Koide, J. Miura, and E. Menegatti, "A portable three-dimensional lidar-based system for long-term and wide-area people behavior measurement," *International Journal of Advanced Robotic Systems*, vol. 16, no. 2, p. 1729881419841532, 2019.
- [31] G. Kim, S. Choi, and A. Kim, "Scan context++: Structural place recognition robust to rotation and lateral variations in urban environments," *IEEE Transactions on Robotics*, vol. 38, no. 3, pp. 1856–1874, 2022.