S-Graphs+: Real-Time Localization and Mapping Leveraging Hierarchical Representations

Hriday Bavle ⬤, Jose Luis Sanchez-Lopez ⬤, *Member, IEEE,*
Muhammad Shaheer ⬤, *Graduate Student Member, IEEE,* Javier Civera ⬤, *Member, IEEE,*
and Holger Voos ⬤, *Member, IEEE*

Abstract—In this letter, we present an evolved version of Situational Graphs, which jointly models in a single optimizable factor graph (1) a pose graph, as a set of robot keyframes comprising associated measurements and robot poses, and (2) a 3D scene graph, as a high-level representation of the environment that encodes its different geometric elements with semantic attributes and the relational information between them. Specifically, our *S-Graphs+* is a novel four-layered factor graph that includes: (1) A keyframes layer with robot pose estimates, (2) a walls layer representing wall surfaces, (3) a rooms layer encompassing sets of wall planes, and (4) a floors layer gathering the rooms within a given floor level. The above graph is optimized in real-time to obtain a robust and accurate estimate of the robot's pose and its map, simultaneously constructing and leveraging high-level information of the environment. To extract this high-level information, we present novel room and floor segmentation algorithms utilizing the mapped wall planes and free-space clusters. We tested *S-Graphs+* on multiple datasets, including simulated and real data of indoor environments from varying construction sites, and on a real public dataset of several indoor office areas. On average over our datasets, *S-Graphs+* outperforms the accuracy of the second-best method by a margin of 10.67%, while extending the robot situational awareness by a richer scene model. Moreover, we make the software available as a docker file.

Index Terms—SLAM, localization, mapping.

Hriday Bavle, Jose Luis Sanchez-Lopez, and Muhammad Shaheer are with Automation and Robotics Research Group, Interdisciplinary Centre for Security, Reliability and Trust, University of Luxembourg, 1855 Luxembourg, Luxembourg (e-mail: hriday.bavle@uni.lu; joseluis.sanchezlopez@uni.lu; muhammad.shaheer@uni.lu).

Javier Civera is with I3A, Universidad de Zaragoza, 50018 Zaragoza, Spain (e-mail: jcivera@unizar.es).

Holger Voos is with Automation and Robotics Research Group, Interdisciplinary Centre for Security, Reliability and Trust, University of Luxembourg, 1855 Luxembourg, Luxembourg, and also with The Faculty of Science, Technology and Medicine, University of Luxembourg, 1855 Luxembourg, Luxembourg (e-mail: holger.voos@uni.lu).

I. INTRODUCTION

ROBOTS require a deep understanding of the situation for their autonomous and intelligent operations [1]. Works like [2], [3], [4], [5] generate 3D scene graphs modeling the environment with high-level semantic abstractions (such as chairs, tables, or walls) and their relationships (such as a set of walls forming a room or a corridor). While providing a rich understanding of the scene, they typically rely on separate SLAM methods, such as [6], [7], [8], that previously estimate the robot's pose and its map using metric/semantic representations without exploiting this hierarchical high-level information of the environment. Methods like [5] do optimize the full 3D scene graphs but only after detection of appropriate loop closures. Thus, in general, 3D scene graphs are not tightly and continuously optimized in a factor graph.

Our previous work *S-Graphs* [9] proposed for the first time a tight coupling of geometric LiDAR SLAM with 3D scene graphs in a single optimizable factor graph, demonstrating state-of-the-art metrics. However, it came with multiple limitations that we overcome in this work with our new *S-Graphs+*, with updated front-end and back-end relying on 3D LiDAR measurements.

Our new **front-end** (Section IV) contributes over *S-Graphs* with (1) a novel *room segmentation* algorithm using free-space clusters and wall planes, providing higher detection recall and removing most heuristics of the *S-Graphs* counterpart; (2) an additional *floor segmentation* algorithm extracting the floor centers using all the currently extracted wall planes.

The new **back-end** (Section V) consists of an improved real-time optimizable factor graph composed of four layers. A *keyframes layer* constraining a sub-set of robot poses at specific distance-time intervals. A *walls layer* constraining of the wall plane parameters and linked to the keyframes using pose-plane constraints. Both the layers are analogous to *S-Graphs*. A *rooms layer* modeling detected rooms to their corresponding wall planes constraining them in a single tightly coupled factor, rather than loosely coupled factors in *S-Graphs*. A *floors layer*, denoting the current floor level in the graph and constraining the rooms at that level, not present in *S-Graphs*. See Fig. 1 for an illustrative example of an *S-Graph+* of a real building.

Our main contributions are, therefore, summarized as:
- A novel real-time factor graph organized in four hierarchical layers and a specifically novel room-to-wall factor.
- A real-time extraction of high-level information, specifically novel room and floor segmentation algorithms.
- A thorough experimental evaluation in different simulated and real construction/office environments as well as software release for the research community.
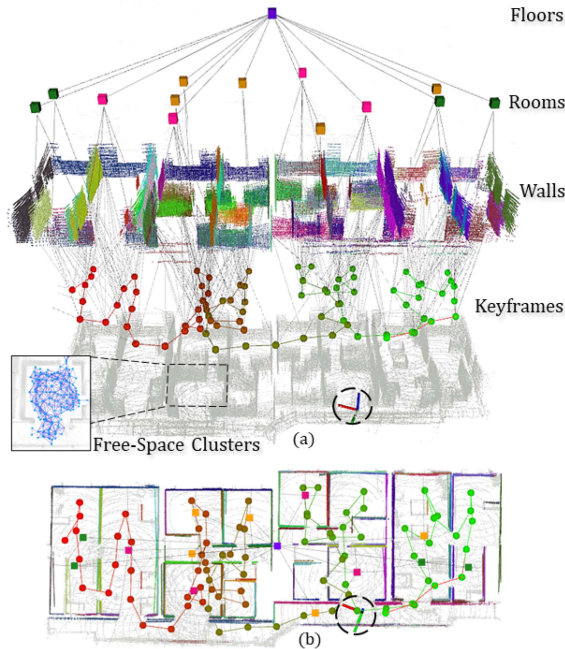
Fig. 1. *S-Graph+* built using a legged robot (circled in black) as it navigates a real construction site consisting of four adjacent houses. (a) 3D view of the four-layered hierarchical optimizable graph. The zoomed-in image shows a partial view of the free-space clusters utilized for room segmentation. (b) Top view of the graph.

II. RELATED WORKS

A. SLAM and Scene Graphs

The literature on LiDAR SLAM is huge, and there are several well-known geometric approaches like LOAM [6] and its variants [7], [10], [11], and also semantic ones like LeGO-LOAM [8], SegMap [12], SUMA++ [13] that provide robust and accurate localization and 3D maps of the environments. While geometric SLAM lacks meaning in the representation of the environments, causing failures in aliased environments and limitations for high-level tasks or human-robot interaction, its semantic SLAM counterparts lack in most occasions geometric accuracy and robustness, due to wrong matches between the semantic elements and the limited relational constraints between them.

Scene graphs, on the other hand, model scenes as structured representations, specifically in the form of a graph comprising objects, their attributes, and the inter-relationships among them. This high-level representation has the potential to boost several relevant challenges in SLAM, such as map compacity or understanding. Focusing on 3D scene graphs for understanding, the pioneering work [2] creates an offline semi-autonomous framework using object detections from RGB images, generating a multi-layered hierarchical representation of the environment and its components, divided mainly into layers of camera, objects, rooms, and building. [14] presents a framework for generating a 3D scene graph using a sequence of images to verify its applicability to visual questioning and answering and to task planning. 3D SSG (Semantic Scene Graph) [15] presents a learning method based on PointNet and Graph Convolutions Networks (GCN) to semi-automatically generate graphs for 3D scenes. SceneGraphFusion [4] on the other hand, generates a real-time incremental 3D scene graph using RGB-D sequences,

accurately handling partial and missed semantic data. 3D DSG (Dynamic Scene Graph) [3] extend the 3D scene graph concept to environments with static parts and dynamic agents in an offline manner, while Hydra [5], presents research in the direction of real-time 3D scene graph generation as well as its optimization using loop closure constraints. Though promising in terms of scene representation and higher-level understanding, a major drawback of these models is that they do not tightly couple the estimate of the scene graph with the SLAM state, in order to simultaneously optimize them. They thus in general generate a scene graph and a SLAM graph in an independent manner. Our previous work *S-Graphs* [9] bridged this gap showcasing the potential of tightly coupling SLAM graphs and scene graphs. However, for several reasons, it was limited to simple structured environments. Our current work *S-Graphs+* overcomes these limitations generating a four-layered hierarchical optimizable graph while simultaneously representing the environment as a 3D scene graph, able to provide an excellent performance even in complex environments.

B. Room Segmentation

For a robot to understand structured indoor environments, it is necessary to first understand their basic components, such as walls, and their composition into higher-level structures such as rooms. Hence, room identification and segmentation is one of the critical tasks in *S-Graphs+*. In the literature, different room segmentation techniques are presented over pre-generated maps using 2D LiDARs [16], [17], [18]. Their performance is, however, degraded in presence of clutter. While [19] presents a room segmentation approach based on pre-generated 2D occupancy maps in cluttered indoor environments, it still lacks real-time capabilities. Methods such as [20], [21], [22] perform segmentation of indoor spaces into meaningful rooms, although they require a pre-generated 3D map of the environment and cannot segment it in real-time. Authors in [5] present a real-time room segmentation approach to classifying different places into rooms but compared to our approach they do not utilize the walls in the environment to efficiently represent the rooms. Given the current state-of-the-art for room segmentation, there was a need to develop a room segmentation algorithm utilizing wall entities while capable of running in real-time as the robot explores its environment, to simultaneously incorporate this high-level information into the optimizable *S-Graphs+*.

III. OVERVIEW

The architecture of *S-Graphs+* is illustrated in Fig. 2. Its pipeline can be divided into six modules, and its estimates are referred to four frames: the LiDAR frame L_t, the robot frame R_t, the odometry frame O, and the map frame M. L_t and R_t are rigidly attached to the robot and depend on the time instant t, while O and M are fixed. The first module receives the 3D LiDAR point cloud in frame L_t, which is pre-filtered and downsampled. The second module estimates the robot odometry in frame O either from LiDAR measurements or the robot encoders. *S-Graphs+* is agnostic to the source of odometry, thus it can utilize odometry estimated either from sensor measurements like 3D LiDAR or directly generated from encoders of the robotic platforms. Four additional front-end modules generate the four-layered topological graph modeling the understanding of the environment, namely: 1) The plane
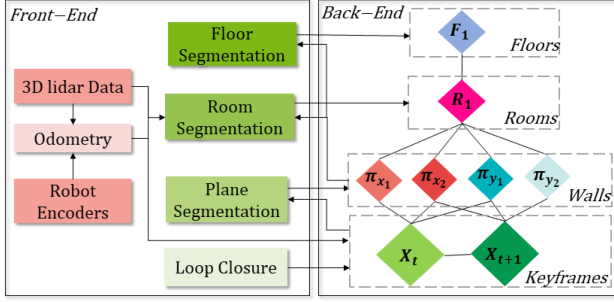
Fig. 2.   *S-Graphs+* overview. Our inputs are the 3D LiDAR measurements and robot odometry, which are pre-filtered and processed in the front-end to extract wall planes, rooms, floor, and loop closures. Note the four-layered *S-Graph+*, whose parameters are jointly optimized in the back-end.
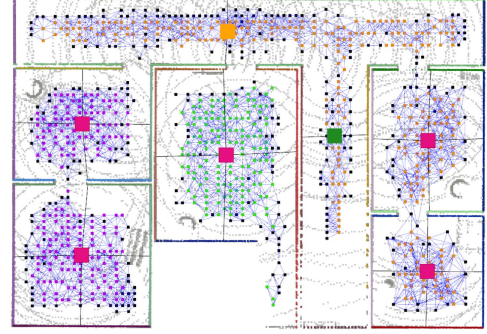

Fig. 3.   Free space clustering and rooms segmentation, obtained from the estimated wall planes surrounding each cluster. Pink colored squares represent a four-wall room, while yellow and green colored squares represent two-wall rooms in x and y directions respectively. Nodes colored in black are those that are closest to walls and vote for splitting the graph.

segmentation module, segmenting and initializing wall planes in the map frame M using the point clouds at each keyframe. 2) The room segmentation module, generating first free-space clusters from the robot poses and 3D LiDAR measurements, and then using such clusters along with the mapped planes to detect room centers in frame M. 3) The floor segmentation module, utilizes the information of all the walls in the map to extract the center of the current floor level in frame M. 4) Finally, the loop closure module as in [9], which utilizes a scan-matching algorithm to recognize revisited places and correct the drift.

We define the global state as:

$$\mathbf{s} = \begin{bmatrix} {}^{M}\mathbf{x}_{R_1}, \ \ldots, \ {}^{M}\mathbf{x}_{R_T}, \ {}^{M}\boldsymbol{\pi}_1, \ \ldots, \ {}^{M}\boldsymbol{\pi}_P, \\ {}^{M}\boldsymbol{\rho}_1, \ \ldots, \ {}^{M}\boldsymbol{\rho}_S, \ {}^{M}\boldsymbol{\kappa}_1, \ \ldots, \ {}^{M}\boldsymbol{\kappa}_K, \\ {}^{M}\boldsymbol{\xi}_1, \ \ldots, \ {}^{M}\boldsymbol{\xi}_F, \ {}^{M}\mathbf{x}_O \end{bmatrix}^{\top}, \quad (1)$$

where ${}^{M}\mathbf{x}_{R_t} \in SE(3)$, $t \in \{1, \ldots, T\}$ are the robot poses at T selected keyframes, ${}^{M}\boldsymbol{\pi}_i \in \mathbb{R}^3$, $i \in \{1, \ldots, P\}$ are the plane parameters of the P wall planes in the scene, ${}^{M}\boldsymbol{\rho}_j \in \mathbb{R}^2$, $j \in \{1, \ldots, S\}$ contains the parameters of the S four-wall rooms and ${}^{M}\boldsymbol{\kappa}_k \in \mathbb{R}^2$, $k \in \{1, \ldots, K\}$ the parameters of the K two-wall rooms, ${}^{M}\boldsymbol{\xi}_f \in \mathbb{R}^2$, $f \in \{1, \ldots F\}$ are the F floors levels, and ${}^{M}\mathbf{x}_O$ models the drift between the odometry frame O and the map frame M.

IV. Front-End

A. Wall Extraction

We use sequential RANSAC to detect and initialize wall planes. In *S-Graphs+*, we extract the wall planes from the 3D pointcloud snapshot for a newly registered keyframe, as opposed to our previous work [9] which extracted wall planes from a continuous stream of 3D pointcloud measurements. This results in efficient detection and mapping of all the wall planes at each keyframe level. Each wall plane extracted at time t, ${}^{L_t}\boldsymbol{\pi}$, is referred to the LiDAR frame L_t, we need to convert it to its Closest Point (CP) representation [9], and then to the map frame ${}^{M}\boldsymbol{\pi}$ using the estimated robot pose at time t. The wall plane normals with their ${}^{M}n_x$ or ${}^{M}n_y$ components greater than the ${}^{M}n_z$ component are classified as vertical planes. Furthermore, normals where ${}^{M}n_x$ is greater than ${}^{M}n_y$ are classified as x-plane normals, and otherwise they are classified as y-plane normals. Finally, planes whose normals' bigger component is ${}^{M}n_z$ are

classified as horizontal planes or ground surfaces. After initializing each plane in the global map, correspondences are searched for every subsequent plane observation. Data association is performed using the Mahalanobis distance between each mapped plane and the newly extracted ones.

B. Room Segmentation

In this work, we present a novel room segmentation strategy capable of segmenting different room configurations in a structured indoor environment improving the room extraction strategy proposed in [9] which only utilized plane-based heuristics to detect potential room candidates. Proposed room segmentation consists of two steps, **Free-Space Clustering** and **Room Extraction**, and the output are the parameters of **four-wall** and **two-wall rooms**.

Free-Space Clustering: Our free-space clustering algorithm divides the free-space graph of a scene into several clusters that should correspond to the rooms of that scene. Given a set of robot poses and a Euclidean Signed Distance Field (ESDF) representation [23] for these poses, we generate a sparse connected graph $\mathcal{G}$ of free spaces using [24]. The drift ${}^{M}\mathbf{x}_O$ estimated after the optimization step in the Back-End (Section V) is utilized to update the ESDF map also updating the graph $\mathcal{G}$. We only maintain an ESDF map and the graph $\mathcal{G}$ up to a certain radius t_r around the robot, clearing the map beyond the radius.

Given the graph $\mathcal{G}$, we cluster it into different free-space regions as follows. We create a filtered graph $\mathcal{G}_f$ removing the vertices $\boldsymbol{v}_d$ whose distance to obstacles is less than a given threshold t_λ. We also remove from $\mathcal{G}_f$ all the edges $\boldsymbol{e}_d$ that are connected to the node set $\boldsymbol{v}_d$. We then run the connected components method [25] on $\mathcal{G}_f$ to divide it into several connected sub-graphs $\mathcal{G}_{f_i}$, $i \in \{1, \ldots, N\}$. In order to re-connect the deleted vertices $\boldsymbol{v}_d$ and their edges $\boldsymbol{e}_d$ to the filtered sub-graphs $\mathcal{G}_{f_i}$, we check within the entire graph $\mathcal{G}$, each edge e_{d_i} that connects vertex v_{f_i} of a filtered sub-graph $\mathcal{G}_{f_i}$ to the deleted vertex v_{d_i}, thus inserting vertex v_{d_i} within $\mathcal{G}_{f_i}$. Using this technique we can obtain disconnected free-space clusters belonging to different rooms, as vertices close to room openings have distances closer to walls (obstacles) and thus vote for disconnecting the graph. Algorithm 1 and Fig. 3 give further details on this free-space clustering.

Algorithm 1: Free-Space Clustering.

Input: Free-space graph $\mathcal{G}$, generated using [24]
Output: Clustered sub-graphs $\mathcal{G}_{f_i}$, $i \in \{1, \ldots, N\}$

1: Filter nodes far from obstacles: $\mathcal{G}_f \leftarrow \mathcal{G}$

---

for $v_i \in \boldsymbol{v}$ & $\boldsymbol{v} \in \mathcal{G}$ **do**
$\quad$ **if** $v_i.distance < t_\lambda$ **then**
$\quad\quad\mid\quad$ $\boldsymbol{v}_d \leftarrow v_i$ & $\boldsymbol{e}_d \leftarrow v_i.edges$
$\quad$ **else**
$\quad\quad\mid\quad$ $\boldsymbol{v}_f \leftarrow v_i$ & $\boldsymbol{e}_f \leftarrow v_i.edges$
$\quad$ **end**
end
$\mathcal{G}_f \leftarrow (\boldsymbol{v}_f, \boldsymbol{e}_f)$

2: Graph clustering by connectivity: $\mathcal{G}_{f_1} \ldots \mathcal{G}_{f_n} \leftarrow \mathcal{G}_f$

$c \leftarrow 0$
for $v_{f_i} \in \boldsymbol{v}_f$ **do**
$\quad\mid\quad$ $v_{f_i}.visited \leftarrow False$ & $v_{f_i}.cluster \leftarrow 0$
end
Function visit(v_f):
$\quad$ **if** $v_{f_i}.visited = False$ **then**
$\quad\quad\mid\quad$ $v_{f_i}.visited \leftarrow True$
$\quad\quad\mid\quad$ **if** $v_{f_i}.cluster = 0$ **then**
$\quad\quad\quad\mid\quad$ $c \leftarrow c + 1$
$\quad\quad\quad\mid\quad$ $v_{f_i}.cluster \leftarrow c$ & $\mathcal{G}_c \leftarrow v_{f_i}$
$\quad\quad\mid\quad$ **end**
$\quad\quad\mid\quad$ **for** $v_{f_n} \in v_{f_i}.neighbours$ **do**
$\quad\quad\quad\mid\quad$ $v_{f_n}.cluster \leftarrow c$ & $visit(v_{f_n})$
$\quad\quad\mid\quad$ **end**
$\quad$ **end**
return
for $v_{f_i} \in \boldsymbol{v}_f$ **do**
$\quad\mid\quad$ $visit(v_{f_i})$
end

3: Inclusion of deleted vertices and edges in the cluster: $\mathcal{G}_{f_i} \leftarrow (\boldsymbol{v}_d, \boldsymbol{e}_d)$

---

for $v_{d_i} \in \boldsymbol{v}_d$ **do**
$\quad$ **for** $v_n \in v_{d_i}.neighbours$ **do**
$\quad\quad\mid\quad$ **if** $v_n \in \mathcal{G}_{f_i}$ **then**
$\quad\quad\quad\mid\quad$ $c \leftarrow v_n.cluster$ & $\mathcal{G}_{f_i} \leftarrow (v_{d_i}, e_{d_i})$
$\quad\quad\mid\quad$ **end**
$\quad$ **end**
end

Room Extraction: Room extraction uses the free-space clusters $\mathcal{G}_{f_i}$ and the wall planes from a keyframe at time t to detect different room configurations. Wall planes are represented in the map frame as $^M\boldsymbol{\Pi} = [^M\boldsymbol{\pi}_i, \ldots, ^M\boldsymbol{\pi}_j]$, where each plane $^M\boldsymbol{\pi}_i = [^M\boldsymbol{n}, ^Md]$ is defined by its normal $^M\boldsymbol{n} = [^Mn_x, ^Mn_y, ^Mn_z]$ and distance ^Md to the origin. All extracted wall planes are first categorized as x-direction planes $^M\boldsymbol{\Pi}_x$, for which their highest normal component is n_x, and y-direction planes $^M\boldsymbol{\Pi}_y$ for which the highest normal dimension is n_y. $^M\boldsymbol{\Pi}_x$ plane are further classified as $^M\boldsymbol{\Pi}_{x_a}$, with $n_x > 0$, and $^M\boldsymbol{\Pi}_{x_b}$ with $n_x < 0$. Analogously $^M\boldsymbol{\Pi}_{y_a}$ and $^M\boldsymbol{\Pi}_{y_b}$ represent y-planes with positive and negative n_y respectively.

Given each sub-category of the wall planes, our room extraction method first checks the $L2$ norm between the 3D points of each plane and the vertices of each cluster $\mathcal{G}_{f_i}$, to find the set of walls lying closer to each specific cluster.

Four-Wall Rooms: For a given cluster $\mathcal{G}_{f_i}$, if the room extraction module finds a set of four wall planes $^M\boldsymbol{\Pi}_s = [^M\boldsymbol{\pi}_{x_{a_1}}, ^M\boldsymbol{\pi}_{x_{b_1}}, ^M\boldsymbol{\pi}_{y_{a_1}}, ^M\boldsymbol{\pi}_{y_{b_2}}]$ close to the cluster vertices, it is considered as a four-wall room candidate and further tests are carried out. First, the widths w_x and w_y of $^M\boldsymbol{\Pi}_x = \{^M\boldsymbol{\pi}_{x_{a_1}}, ^M\boldsymbol{\pi}_{x_{b_1}}\}$ and $^M\boldsymbol{\Pi}_y = \{^M\boldsymbol{\pi}_{y_{a_1}}, ^M\boldsymbol{\pi}_{y_{b_1}}\}$ should be greater than a given threshold t_w, where:

$$w_x = \left[|^Md_{x_{a_1}}| \cdot {}^M\mathbf{n}_{x_{a_1}} - |^Md_{x_{b_1}}| \cdot {}^M\mathbf{n}_{x_{b_1}} \right]$$
$$w_y = \left[|^Md_{y_{a_1}}| \cdot {}^M\mathbf{n}_{y_{a_1}} - |^Md_{y_{b_1}}| \cdot {}^M\mathbf{n}_{y_{b_1}} \right] \quad (2)$$

$^Md_{x_{a_1}}$ and $^Md_{x_{b_1}}$ are the plane distances to the origin and $^M\mathbf{n}_{x_{a_1}}$ and $^M\mathbf{n}_{x_{b_1}}$ are the normals of x-planes. Similarly, $^Md_{y_{a_1}}$, $^Md_{yb_1}$, $^M\mathbf{n}_{y_{a_1}}$ and $^M\mathbf{n}_{y_{b_1}}$ are the distances and normals for y-planes. For (2) to hold true, $|^Md_{x_{a_1}}| > |^Md_{x_{b_1}}|$ and $|^Md_{y_{a_1}}| > |^Md_{y_{b_1}}|$. All plane normals are converted to point away from the map M frame as:

$$^M\mathbf{n} = \begin{cases} -1 \cdot {}^M\mathbf{n} & \text{if}\, {}^Md > 0 \\ {}^M\mathbf{n} & \text{otherwise} \end{cases} \quad (3)$$

If the above test is successful, the 3D points in each wall are checked to be enclosed within the two apposed walls. For example, in-plane points belonging to $^M\boldsymbol{\pi}_{x_{a_1}}$ are checked to lie within the points of $^M\boldsymbol{\pi}_{y_{a_1}}$ and $^M\boldsymbol{\pi}_{y_{b_1}}$. Given a room candidate with a planar set $^M\boldsymbol{\Pi}_s$ consisting of four walls, we first calculate the room center as follows:

$$^M\mathbf{r}_{x_i} = \frac{1}{2} \left[|^Md_{x_{a_1}}| \cdot {}^M\mathbf{n}_{x_{a_1}} - |^Md_{x_{b_1}}| \cdot {}^M\mathbf{n}_{x_{b_1}} \right]$$
$$+ |^Md_{x_{b_1}}| \cdot {}^M\mathbf{n}_{x_{b_1}}$$
$$^M\mathbf{r}_{y_i} = \frac{1}{2} \left[|^Md_{y_{a_1}}| \cdot {}^M\mathbf{n}_{y_{a_1}} - |^Md_{y_{b_1}}| \cdot {}^M\mathbf{n}_{y_{b_1}} \right]$$
$$+ |^Md_{y_{b_1}}| \cdot {}^M\mathbf{n}_{y_{b_1}}$$
$$^M\boldsymbol{\rho}_i = {}^M\boldsymbol{r}_{x_i} + {}^M\boldsymbol{r}_{y_i} \quad (4)$$

Equation (4) holds true when $|d_{x_1}| > |d_{x_2}|$. Again, all planes are converted to point away from the origin M using (3).

Data association for the room node follows two steps. First, the $L2$ norm between the positions of the mapped rooms with the newly detected ones is calculated. Second, the shortlisted rooms using the first step undergo id checks at each wall plane and cases with id mismatch further undergo Mahalanobis distance check. This process allows for the identification and merging of duplicate wall planes (id mismatch) for a given room, arising from inaccuracies in the wall plane matching step (Section IV-A) as the room and its respective wall plane matching thresholds can be safely tuned to be larger than the single wall plane matching threshold.

Two-Wall Rooms: The room extraction method is sometimes able to find only two walls that surround a free-space cluster $\mathcal{G}_{f_i}$. These two-wall rooms can be rooms with some undetected walls or corridor-like structures. If a wall plane set $^M\boldsymbol{\Pi}_s = [^M\boldsymbol{\pi}_{x_{a_1}}, ^M\boldsymbol{\pi}_{x_{b_1}}]$ contains two x-planes then it is a two-wall room in x direction. Analogously, two-wall rooms in the y direction are composed of opposed y-planes. Walls forming two-wall rooms undergo the same checks as four-wall rooms, shown in (3) and (2). Given the fact that two-wall rooms contain information in either x or y direction, the corresponding center $^M\boldsymbol{c}_i$ of the cluster $\mathcal{G}_{f_i}$ is also utilized to compute the two-wall room center as follows:

$$^M\mathbf{r}_{x_i} = \frac{1}{2} \left[|^Md_{x_{a_1}}| \cdot {}^M\mathbf{n}_{x_{a_1}} - |^Md_{x_{b_1}}| \cdot {}^M\mathbf{n}_{x_{b_1}} \right]$$
$$+ |^Md_{x_{b_1}}| \cdot {}^M\mathbf{n}_{x_{b_1}}$$
$$^M\boldsymbol{\kappa}_{x_i} = {}^M\mathbf{r}_{x_i} + \left[{}^M\mathbf{c}_i - \left[{}^M\mathbf{c}_i \cdot {}^M\hat{\mathbf{r}}_{x_i} \right] \cdot {}^M\hat{\mathbf{r}}_{x_i} \right] \quad (5)$$

where $^M\boldsymbol{\kappa}_{x_i}$ is the two-wall room center in x direction, $^M\hat{\mathbf{r}}_{x_i} = {}^M\mathbf{r}_{x_i}/\|^M\mathbf{r}_{x_i}\|$, and $^M\boldsymbol{c}_i$ is the cluster center obtained from the

endpoints of the cluster $\mathcal{G}_{f_i}$ as:

$$^M c_{x_i} = \frac{1}{2} \left[^M p_{x_1} - \, ^M p_{x_2} \right] + \, ^M p_{x_2}$$

$$^M c_{y_i} = \frac{1}{2} \left[^M p_{y_1} - \, ^M p_{y_2} \right] + \, ^M p_{y_2}$$

$$^M \mathbf{c}_i = \left[^M c_{x_i}, \, ^M c_{y_i} \right] \tag{6}$$

where $^M p_{x_1}$ to $^M p_{y_2}$ are the cluster endpoints. Two-wall room center in y direction can be calculated analogously.

Data association of two-wall rooms follows a similar concept as four-wall rooms. In the case of a two-wall room in the x direction, we first compute the $L2$ norm along the x-axis of the two-wall room center followed by the id check of individual wall planes. Cases with id mismatch further undergo $L2$ norm check of the planar points between the detected and the mapped wall planes.

Detected four and two-wall rooms are optimized along with their corresponding wall planes in the back-end explained in Section V.

C. Floor Segmentation

The floor segmentation module extracts the widest wall planes within the current explored floor level by the robot which can then be used to calculate the center of the current floor level. Our floor segmentation utilizes the information from all mapped walls to create a sub-category of wall planes as described in the room segmentation (Section IV-B) as, $^M \mathbf{\Pi}_{s_t}$ where $t = \{1, \ldots, T\}$. After receiving a complete plane set it computes the widths $\boldsymbol{w}_x$ between all x-direction planes and similarly $\boldsymbol{w}_y$ for y-direction planes using (2). 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 product 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 computes the floor center node using the obtained wall plane candidates following (4). Whenever the robot ascends or descends to a different floor level, the newly mapped wall planes are incorporated with the new floor, and the current floor center is computed only using the wall planes at that floor.

V. BACK-END

The back-end is responsible for creating and optimizing the four-layered *S-Graphs+* summing the individual cost functions of each layer, explained in detail as follows.

Keyframes: This layer creates a factor node $^M \boldsymbol{x}_{R_t} \in SE(3)$ with the robot keyframe pose at time t in the map frame M. The pose nodes are constrained by pairwise odometry readings between consecutive poses as in [9].

Walls: This layer creates the planar factor nodes for the wall planes extracted by the wall segmentation (Section IV-A). The planar nodes are factored as $^M \boldsymbol{\pi} = [^M \phi, \, ^M \theta, \, ^M d]$, where $^M \phi$ and $^M \theta$ stand for the azimuth and elevation of the plane in frame M. The planar nodes are constrained with their corresponding keyframes using pose-plane constraints as in [9]. The room segmentation module utilizes mapped walls at current keyframe k_t (Section IV-B) to identify different room candidates, whereas mapped walls from all the mapped keyframes $\boldsymbol{k} = \{k_1, \ldots, k_T\}$

are utilized by the floor segmentation module (Section IV-C) to identify the center of the floor level.

Rooms: The rooms layer receives the extracted room candidates and their corresponding wall planes from the room segmentation module (Section IV-B) to create appropriate constraints between them.

Four-Wall Rooms: We propose a novel edge formulation that minimizes in a single cost function the room node (generated from its center) and its four mapped wall planes, as opposed to [9] which comprised of individual cost functions for room and the wall planes. The proposed cost function can be written as:

$$c_{\boldsymbol{\rho}} \left(^M \boldsymbol{\rho}, \left[^M \boldsymbol{\pi}_{x_{a_i}}, \, ^M \boldsymbol{\pi}_{x_{b_i}}, \, ^M \boldsymbol{\pi}_{y_{a_i}}, \, ^M \boldsymbol{\pi}_{y_{b_i}} \right] \right)$$

$$= \sum_{t=1,i=1}^{T,S} \| ^M \hat{\boldsymbol{\rho}}_i - f(^M \tilde{\boldsymbol{\pi}}_{x_{a_i}}, \, ^M \tilde{\boldsymbol{\pi}}_{x_{b_i}}, \, ^M \tilde{\boldsymbol{\pi}}_{y_{a_i}}, \, ^M \tilde{\boldsymbol{\pi}}_{y_{b_i}}) \|_{\boldsymbol{\Lambda}_{\tilde{\rho}_{i,t}}}^2 \tag{7}$$

Where $^M \hat{\boldsymbol{\rho}}_i$ is the estimated four-wall room center obtained from Section IV-B and $f(^M \tilde{\boldsymbol{\pi}}_{x_{a_i}}, \, ^M \tilde{\boldsymbol{\pi}}_{x_{b_i}}, \, ^M \tilde{\boldsymbol{\pi}}_{y_{a_i}}, \, ^M \tilde{\boldsymbol{\pi}}_{y_{b_i}})$ is the function mapping the four wall planes estimated to a four-wall room center using (4). Compared to [9] which only included scalar values d in the room center computation, (4) now includes both the normal direction n and distance d. The goal of this cost function is to maintain the structural consistency between the four planes forming the room.

Two-Wall Rooms: We propose a similar improved cost function to minimize room nodes and their two corresponding wall planes as follows:

$$c_{\boldsymbol{\kappa}} \left(^M \boldsymbol{\kappa}_i, \left[^M \boldsymbol{\pi}_{x_{a_1}}, \, ^M \boldsymbol{\pi}_{x_{b_1}}, \, ^M \mathbf{c}_i \right] \right)$$

$$= \sum_{t=1,i=1}^{T,K} \| ^M \hat{\boldsymbol{\kappa}}_i - f \left(^M \tilde{\boldsymbol{\pi}}_{x_{a_1}}, \, ^M \tilde{\boldsymbol{\pi}}_{x_{b_1}}, \, ^M \mathbf{c}_i \right) \|_{\boldsymbol{\Lambda}_{\tilde{\kappa}_{i,t}}}^2 \tag{8}$$

$^M \mathbf{c}_i$ is the cluster center, which is kept constant during the optimization, and $^M \hat{\boldsymbol{\kappa}}_i$ is the estimated two-wall room center in x direction obtained from Section IV-B. $f(^M \tilde{\boldsymbol{\pi}}_{x_{a_1}}, \, ^M \tilde{\boldsymbol{\pi}}_{x_{b_1}}, \, ^M \mathbf{c}_i)$ maps the two wall planes along with its cluster center to a room center using (5). When comparing with [9], (5) now includes both orientation n, distance d, and the cluster center. The cost function to minimize two-wall rooms in y direction follows (8) for wall planes $(^M \boldsymbol{\pi}_{y_{a_j}}, \, ^M \boldsymbol{\pi}_{y_{b_j}})$ and cluster center $^M \mathbf{c}_j$. Duplicate wall plane nodes identified during the four-wall or two-wall room segmentation are constrained by a factor minimizing the difference between their respective parameters.

Floors: The floor node consists of the center of the current floor level calculated from the floor segmentation (Section IV-C). We add a cost function between the floor node and all the mapped four-wall rooms at that floor level as follows:

$$c_{\xi} \left(^M \boldsymbol{\xi}_i, \, ^M \boldsymbol{\rho}_i \right) = \sum_{t=1,i=1,j=1}^{T,F,S} \| ^M \hat{\boldsymbol{\delta}}_{\xi,\rho_j} - f \left(^M \boldsymbol{\xi}_i, \, ^M \boldsymbol{\rho}_j \right) \|_{\boldsymbol{\Lambda}_{\tilde{\xi}_{i,t}}}^2 \tag{9}$$

where $^M \hat{\boldsymbol{\delta}}_{\xi,\rho_j}$ stands for the relative distance between the floor i with center $\boldsymbol{\xi}_i$ and the four-wall room j with center $\boldsymbol{\rho}_j$, and $f(^M \boldsymbol{\xi}_i, \, ^M \boldsymbol{\rho}_j)$ maps the relative distance between the centers of floor node and four-wall room node. Two-wall room nodes are constrained with the floor node using the same (9). While

the robot navigates in the surroundings and discovers new wall planes, the estimate of the floor node might change due to the insertion of such planes into the map. If the current floor center calculated from the new wall planes gets updated beyond a threshold t_f, the estimate of the floor node is updated in the graph accordingly along with the relative distances between the floors and all the rooms.

VI. EXPERIMENTAL RESULTS

A. Methodology

S-Graphs+ is built on top of its baseline *S-Graphs* [9] and is validated over several construction sites and office spaces in both simulated and real-world scenarios, comparing it against several state-of-the-art LiDAR SLAM frameworks and its baseline. We utilize VLP-16 LiDAR data in all the datasets. To validate the presented novelty, we ablate *S-Graphs+* into *S-Graphs+* w. OR comprising older room detection algorithm from *S-Graphs* but newly proposed room-to-wall plane factors (Section V) and *S-Graphs+* w. OF with older factors from *S-Graphs* but the newly proposed room detection algorithm (Section IV-B). We do not ablate the proposed floor layer, as currently the floor level (Section IV-C) is mostly used to add semantic meaning to the map, without significantly improving the accuracy. Furthermore, we compare the room detection of *S-Graphs+* against the heuristics-based one in *S-Graphs*, reporting the precision and recall of the four-walled and two-walled room detections in the real-world scenarios, for which we have the ground truth number of rooms defined in the architectural plans.

In all the experiments, no fine-tuning of the mentioned thresholds was required and the same prior empirically selected thresholds sufficed for all. The ESDF map (Section IV-B) resolution depends on the LiDAR resolution, which in our case is computed as 0.18 m vertically and 0.03 m horizontally, while the map clearing threshold t_r is kept to 10 m, while t_λ and t_w are kept to 0.8 m and 0.5 m respectively. The plane matching threshold (Section IV-A) is kept to 0.35 m while the room matching threshold (Section IV-B) is 1 m. The dot product threshold t_n between the wall planes (Section IV-C) is kept to 0.9.

Simulated Data: We conduct a total of five simulated experiments. *CF1* and *CF2*, are generated from the 3D meshes of two floors of actual architectural plans, while *SE1*, *SE2*, and *SE3*, are performed in additional simulated environments resembling typical indoor environments with different room configurations. We report the ATE against the provided ground truth. Due to absence of odometry from robot encoders, in all simulated experiments the odometry is estimated only from LiDAR measurements. For a fair validation, *S-Graphs+* is run using two different odometry inputs, specifically VGICP [26] and FLOAM [7].

In-House Dataset: In all our in-house data we utilize the robot encoders for estimating the odometry. The first two experiments, *C1F1* and *C1F2*, are performed on two floors of a construction site consisting of a single house. Additionally, *C2F0*, *C2F1*, and *C2F2* consist of three floors of an ongoing construction site combining four individual houses. *C3F1*, and *C3F2* are two combined houses, while *C4F0* is a basement area with different storage rooms. To validate the accuracy of each method in all the real experiments we report the RMSE of the estimated 3D maps against the actual 3D map generated from the architectural plan

TABLE I
ABSOLUTE TRAJECTORY ERROR (ATE) [M], OF *S-GRAPH+* AND RELEVANT BASELINES ON SIMULATED DATA

Method		Dataset (m x 10^{-2})					
Mapping	Odometry	C1F0	C1F2	SE1	SE2	SE3	Avg
HDL-SLAM [11]	VGICP [26]	9.42	2.12	2.46	10.6	6.23	6.16
ALOAM [6]	ALOAM	9.90	8.70	15.7	40.2	19.7	18.8
MLOAM [10]	MLOAM	-	50.2	66.1	-	15.7	44.0
FLOAM [7]	FLOAM	11.7	14.5	14.6	30.5	27.6	19.8
LeGO-LOAM [8]	LeGO-LOAM	-	-	-	-	74.1	74.1
S-Graphs [9]	VGICP	5.09	2.57	2.18	9.10	3.86	4.56
S-Graphs+ w. OR	VGICP	4.95	2.52	**1.86**	**8.06**	4.00	4.28
S-Graphs+ w. OF	VGICP	5.31	3.00	2.18	8.40	3.93	4.56
S-Graphs+ (ours)	VGICP	**4.47**	**1.75**	1.91	9.31	3.37	**4.16**
S-Graphs+ (ours)	FLOAM	5.94	11.7	5.72	9.60	19.9	10.5

Best results are boldfaced, second best are underlined. '-' refers to an unsuccessful run.

TABLE II
POINT CLOUD RMSE [M] FOR OUR IN-HOUSE REAL SEQUENCES

	Dataset (m x 10^{-2})							
Method	Point Cloud RMSE							
Mapping	C1F1	C1F2	C2F0	C2F1	C2F2	C3F1	C3F2	Avg
HDL-SLAM [11]	33.5	19.8	18.5	21.1	19.5	22.9	19.4	22.1
ALOAM [6]	52.6	33.6	34.1	45.1	29.9	36.5	43.4	39.3
MLOAM [10]	45.0	27.6	40.6	32.4	23.6	-	-	33.8
FLOAM [7]	68.5	39.2	40.2	55.5	39.5	58.3	38.8	48.6
LeGO-LOAM [8]	-	-	39.2	45.5	-	52.9	50.3	47.0
S-Graphs [9]	33.1	**18.9**	18.4	21.8	**17.6**	22.8	22.6	22.2
S-Graphs+ w. OR	**32.4**	19.0	17.7	19.9	18.2	24.1	19.4	21.5
S-Graphs+ w. OF	32.8	19.0	17.0	20.1	**17.6**	23.3	19.3	21.3
S-Graphs+ (ours)	32.9	**18.9**	16.9	18.9	17.6	22.3	18.7	**20.9**

All methods use odometry from robot encoders. Best results are boldfaced, second best underlined. '-' refers to an unsuccessful run.

except for experiment *C4F0*, for which we provide qualitative results due to the absence of a ground truth plan.

TIERS LiDARs dataset: We also validate *S-Graphs+* on the public TIERS dataset [27], recorded by a moving platform in a variety of scenarios. Experiments *T6* to *T8* are done in a single small room in which the platform does several passes at increasing speeds. Experiments *T10* and *T11* are performed in a larger indoor hallway with longer trajectories. We report the ATE against the provided ground truth. Due to the absence of encoder readings in this dataset, each baseline method uses its own LiDAR-based odometry. As in the simulated datasets, we validate *S-Graphs+* with VGICP and FLOAM odometry.

B. Results and Discussion

Simulated Data: Table I showcases the ATE for the simulated experiments. *S-Graphs+* w. OR results in an average improvement in accuracy of 5.44% over *S-Graphs*, while *S-Graphs+* w. OF shows an average decrease in accuracy of 3.03% over *S-Graphs*. However, the full *S-Graphs+* with both the new room detector and newly proposed factors shows an improved average accuracy of 13.37% over the baseline. It can also be seen in Table I *S-Graphs+* is run using two different odometry methods, VGICP and FLOAM, and that it improves the respective odometries by 51.88% and 106.5%.

In-House Dataset: Table II presents the point cloud RMSE. As it can be observed in the table, *S-Graphs+* outperforms the second-best baseline by a margin of 5.93%. *S-Graphs+* w. OR

(a) *S-Graphs+* (b) HDL-SLAM

(c) ALOAM (d) FLOAM
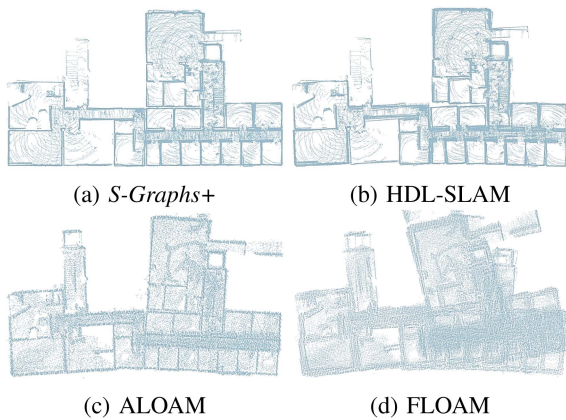
Fig. 4. Maps by *S-Graphs+* and baselines, in-house seq. *C4F0*.
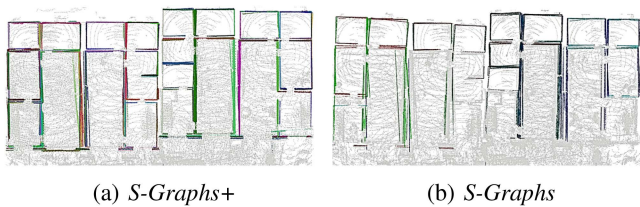
(a) *S-Graphs+* (b) *S-Graphs*

Fig. 5. *S-Graphs+* and *S-Graphs* maps, in-house seq. *C2F0*.
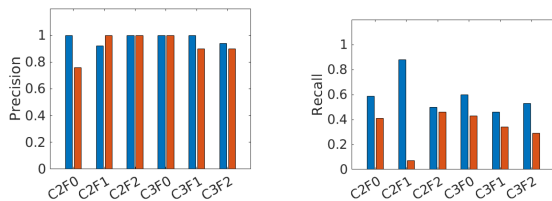
Fig. 6. Precision and recall for *S-Graphs+* (blue) and *S-Graphs* (red) on six different scenes of our in-house dataset.

TABLE III
COMPUTATION TIME [MS] OF *S-GRAPHS+* ALONG THE TOTAL LENGTH OF THE SEQUENCE [S] FOR IN-HOUSE DATASET

Module	Computation Time (mean) [ms]						
	C1F1	*C1F2*	*C2F0*	*C2F1*	*C2F2*	*C3F1*	*C3F2*
Plane Segmentation	91.8	47.4	68.9	45.3	82.2	57.6	44.8
Room Segmentation	17.6	9.8	9.6	5.3	10.7	2.9	4.2
Floor Segmentation	8.1	3.4	4.6	7.2	44.7	4.0	16.9
Back-End	74.0	105.7	87.3	169.0	263.1	124.5	173.2
Sequence Length [s]	487	657	238	672	1044	558	999

Dataset appears above the Computation Time header.

TABLE IV
ABSOLUTE TRAJECTORY ERROR (ATE) [M], OF *S-GRAPHS+* AND RELEVANT BASELINES ON THE TIERS DATASET [27]

Method		Dataset (m x 10^{-2})					
Mapping	Odometry	*T6*	*T7*	*T8*	*T10*	*T11*	*Avg*
HDL-SLAM [11]	VGICP [26]	25.6	27.3	**31.0**	148.9	287.1	104.0
ALOAM [6]	ALOAM	25.7	27.0	34.6	<u>68.1</u>	234.9	78.1
MLOAM [10]	MLOAM	25.7	**26.1**	33.9	263.4	**47.4**	79.3
FLOAM [7]	FLOAM	25.8	<u>26.3</u>	32.4	71.3	161.1	63.4
LeGO-LOAM [8]	LeGO-LOAM	27.3	33.5	36.3	140.9	68.2	<u>61.2</u>
S-Graphs [9]	VGICP	25.6	26.8	35.1	260.1	190.1	107.5
S-Graphs+ w. OR	VGICP	<u>25.3</u>	26.4	32.8	133.0	173.5	78.2
S-Graphs+ w. OF	VGICP	26.5	27.1	32.9	144.9	198.3	85.9
S-Graphs+ (ours)	VGICP	25.6	26.6	32.9	126.6	162.3	74.8
S-Graphs+ (ours)	FLOAM	**25.2**	26.5	<u>32.1</u>	**48.3**	<u>60.6</u>	**38.5**

Best results boldfaced, second best underlined.

and *S-Graphs+* w. OF individually outperform its baseline by 2.74% and 4.03% respectively. For experiment *C4F0*, Fig. 4 shows a top view of the final maps estimated by *S-Graphs+* and three other baselines. Observe the higher degree of accuracy and cleaner map elements in the *S-Graphs+* case, the latest indicating a better alignment for different robot passes. Similarly, observe the precise map generated by *S-Graphs+* in Fig. 5 for experiment *C2F0* when comparing with *S-Graphs*. Fig. 1, shows the entire four-layered *S-Graphs+* for *C2F2* along with its map accuracy.

Fig. 6 presents the precision/recall of the room detection in *S-Graphs+* and the one in *S-Graphs*. Note how the precision is slightly higher for *S-Graphs+* and, more importantly, the recall is substantially higher for *S-Graphs+*. In particular, the difference is notable for scenarios with complex layouts such as *C2F1* which also improves the final map accuracy (Table II). The latest is one of the main strengths of *S-Graphs+:* Extracting a higher number of rooms adds a higher number of constraints leading to more accurate estimates and a better representation.

Additionally, Table III provides a comprehensive overview of the computation time required by each module within *S-Graphs+*. Plane segmentation runtime can vary depending on the area of the wall planes (wall planes with a larger area have a higher number of points, increasing the computation time). The runtime of room segmentation can vary given the number of mapped wall planes in the environment at a given time instant around the robot, while floor segmentation runtime can vary given the current mapped wall planes in the environment (higher number of mapped wall planes increases the computation time). The back-end computation time increases with the length of the experiment, as the graph size typically increases with time, but even with sequence lengths of approximately 17 mins (*C2F2*), all the modules of *S-Graphs+* are able to maintain real-time performance.

TIERS LiDARs dataset: Table IV presents the ATE for all baseline methods and our *S-Graphs+* in the indoor sequences of the public TIERS dataset [27]. On an average *S-Graphs+* with FLOAM odometry gives the best results in all the experiments, improving by 43.2% over FLOAM. Individually *S-Graphs+* w. OR and *S-Graphs+* w. OF improve the average accuracy by 13.36% and 8.32% over *S-Graphs*, while *S-Graphs+* shows average improved accuracy by 14.59% over the second-best baseline. Note that all methods perform similarly for small scenes, but differ as scenes become larger. *S-Graphs+* presents significant error reductions for large environments. The strength of our hierarchical representation is particularly evident in scenarios like *T11*, in which *S-Graphs+* utilizing FLOAM odometry increases the FLOAM accuracy by 165.8% (Table IV and Fig. 7). Observe also in Fig. 7 the good performance of *S-Graphs+* in non-Manhattan worlds.
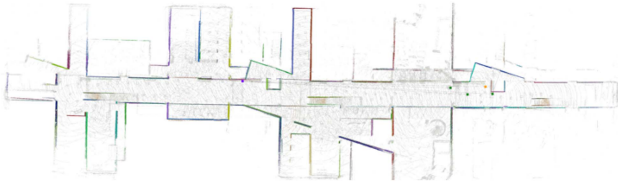
Fig. 7. Map estimated by *S-Graphs+* on TIERS sequence *T11*.

VII. CONCLUSION

In this work, we present *S-Graphs+*, a novel four-layered hierarchical factor graph composed of: A *keyframes layer* constraining a sub-set of robot poses at specific distance-time intervals. A *walls layer* constraining the wall plane parameters and linking it to the keyframes. A *rooms layer* modeling detected rooms to their corresponding wall planes and a *floors layer*, denoting the current floor level in the graph and constraining the rooms at that level. To extract this high-level information we also propose a novel room segmentation algorithm using free-space clusters and wall planes and a floor segmentation algorithm extracting the floor centers using all the currently extracted wall planes. We demonstrate an average improvement in the accuracy of 10.67% against the second-best method on our simulated and real experiments covering different indoor environments. In future work, we plan to exploit the hierarchical structure of the graph for efficient and faster optimization and validate it over buildings with several floors as well as enhance the reasoning over the graph for improving the detection of different relationship constraints between its semantic elements.

ACKNOWLEDGMENT

REFERENCES

[1] H. Bavle, J. L. Sanchez-Lopez, C. Cimarelli, A. Tourani, and H. Voos, "From SLAM to situational awareness: Challenges and survey," *Sensors*, vol. 23, no. 10, 2023, Art. no. 4849. [Online]. Available: https://www.mdpi.com/1424-8220/23/10/4849

[2] I. Armeni et al., "3D scene graph: A structure for unified semantics, 3D space, and camera," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2019, pp. 5664–5673.

[3] A. Rosinol, A. Gupta, M. Abate, J. Shi, and L. Carlone, "3D dynamic scene graphs: Actionable spatial perception with places, objects, and humans," in *Proc. Robot.: Sci. Syst.*, 2020.

[4] S.-C. Wu, J. Wald, K. Tateno, N. Navab, and F. Tombari, "Scenegraphfusion: Incremental 3D scene graph prediction from RGB-D sequences," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2021, pp. 7515–7525.

[5] N. Hughes, Y. Chang, and L. Carlone, "Hydra: A real-time spatial perception system for 3D scene graph construction and optimization," in *Proc. Robot.: Sci. Syst.*, 2022.

[6] J. Zhang and S. Singh, "LOAM: LiDAR odometry and mapping in real-time," in *Proc. Robot.: Sci. Syst.*, 2014.

[7] H. Wang, C. Wang, C. Chen, and L. Xie, "F-LOAM: Fast LiDAR odometry and mapping," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2021. pp. 4390–4396.

[8] T. Shan and B. Englot, "LeGO-LOAM: Lightweight and ground-optimized lidar odometry and mapping on variable terrain," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2018, pp. 4758–4765.

[9] H. Bavle, J. L. Sanchez-Lopez, M. Shaheer, J. Civera, and H. Voos, "Situational graphs for robot navigation in structured indoor environments," *IEEE Robot. Automat. Lett.*, vol. 7, no. 4, pp. 9107–9114, Oct. 2022.

[10] J. Jiao, H. Ye, Y. Zhu, and M. Liu, "Robust odometry and mapping for multi-LiDAR systems with online extrinsic calibration," *IEEE Trans. Robot.*, vol. 38, no. 1, pp. 351–371, Feb. 2022.

[11] K. Koide, J. Miura, and E. Menegatti, "A portable three-dimensional LIDAR-based system for long-term and wide-area people behavior measurement," *Int. J. Adv. Robot. Syst.*, vol. 16, no. 2, Mar. 2019, Art. no. 1729881419841532.

[12] R. Dubé et al., "SegMap: Segment-based mapping and localization using data-driven descriptors," *Int. J. Robot. Res.*, vol. 39, no. 2–3, pp. 339–355, Jul. 2019.

[13] X. Chen, A. Milioto, E. Palazzolo, P. Giguére, J. Behley, and C. Stachniss, "SuMa++ : Efficient LiDAR-based semantic SLAM," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2019, pp. 4530–4537.

[14] U.-H. Kim, J.-M. Park, T. jin Song, and J.-H. Kim, "3-D scene graph: A sparse and semantic representation of physical environments for intelligent agents," *IEEE Trans. Cybern.*, vol. 50, no. 12, pp. 4921–4933, Dec. 2020.

[15] J. Wald, H. Dhamo, N. Navab, and F. Tombari, "Learning 3D semantic scene graphs from 3D indoor reconstructions," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 3961–3970.

[16] R. Bormann, F. Jordan, W. Li, J. Hampp, and M. Hägele, "Room segmentation: Survey, implementation, and analysis," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2016, pp. 1019–1026.

[17] M. Mielle, M. Magnusson, and A. J. Lilienthal, "A method to segment maps from different modalities using free space layout maoris: Map of ripples segmentation," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2018, pp. 4993–4999.

[18] F. Foroughi, J. Wang, A. Nemati, Z. Chen, and H. Pei, "MapSegNet: A fully automated model based on the encoder-decoder architecture for indoor map segmentation," *IEEE Access*, vol. 9, pp. 101530–101542, 2021.

[19] M. Luperto, T. P. Kucner, A. Tassi, M. Magnusson, and F. Amigoni, "Robust structure identification and room segmentation of cluttered indoor environments from occupancy grid maps," *IEEE Robot. Automat. Lett.*, vol. 7, no. 3, pp. 7974–7981, Jul. 2022.

[20] I. Armeni et al., "3D semantic parsing of large-scale indoor spaces," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 1534–1543.

[21] R. Ambruş, S. Claici, and A. Wendt, "Automatic room segmentation from unstructured 3-D data of indoor environments," *IEEE Robot. Automat. Lett.*, vol. 2, no. 2, pp. 749–756, Apr. 2017.

[22] S. Ochmann, R. Vock, and R. Klein, "Automatic reconstruction of fully volumetric 3D building models from oriented point clouds," *ISPRS J. Photogrammetry Remote Sens.*, vol. 151, pp. 251–262, 2019.

[23] H. Oleynikova, Z. Taylor, M. Fehr, R. Siegwart, and J. Nieto, "Voxblox: Incremental 3D euclidean signed distance fields for on-board MAV planning," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2017, pp. 1366–1373.

[24] H. Oleynikova, Z. Taylor, R. Siegwart, and J. Nieto, "Sparse 3D topological graphs for micro-aerial vehicle planning," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2018, pp. 1–9.

[25] J. Clark and D. A. Holton, *A First Look At Graph Theory*. Singapore: World Scientific, 1991.

[26] K. Koide, M. Yokozuka, S. Oishi, and A. Banno, "Voxelized GICP for fast and accurate 3D point cloud registration," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2021, pp. 11054–11059.

[27] L. Qingqing, Y. Xianjia, J. P. Queralta, and T. Westerlund, "Multi-modal LiDAR dataset for benchmarking general-purpose localization and mapping algorithms," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2022, pp. 3837–3844.