2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 14-18, 2024. Abu Dhabi, UAE

Learning High-level Semantic-Relational Concepts for SLAM

Jose Andres Millan-Romera[1], Hriday Bavle[1], Muhammad Shaheer[1],
Martin R. Oswald[2], Holger Voos[1], and Jose Luis Sanchez-Lopez[1]

***Abstract*— Recent works on SLAM extend their pose graphs with higher-level semantic concepts like *Rooms* exploiting relationships between them, to provide, not only a richer representation of the situation/environment but also to improve the accuracy of its estimation. Concretely, our previous work, Situational Graphs (*S-Graphs+*), a pioneer in jointly leveraging semantic relationships in the factor optimization process, relies on semantic entities such as *Planes* and *Rooms*, whose relationship is mathematically defined. Nevertheless, there is no unique approach to finding all the hidden patterns in lower-level factor-graphs that correspond to high-level concepts of different natures. It is currently tackled with ad-hoc algorithms, which limits its graph expressiveness.**

To overcome this limitation, in this work, we propose an algorithm based on Graph Neural Networks for learning high-level semantic-relational concepts that can be inferred from the low-level factor graph. Given a set of mapped *Planes* our algorithm is capable of inferring *Room* entities relating to the *Planes*. Additionally, to demonstrate the versatility of our method, our algorithm can infer an additional semantic-relational concept, i.e. *Wall*, and its relationship with its *Planes*. We validate our method in both simulated and real datasets demonstrating improved performance over two baseline approaches. Furthermore, we integrate our method into the *S-Graphs+* algorithm providing improved pose and map accuracy compared to the baseline while further enhancing the scene representation.

I. Introduction

High-level semantic-relational entities enhance a robot's situational awareness [2] and enrich the built world model for improved scene understanding. It further provides advantageous information for successive tasks such as planning [3] or robot navigation [4].

During recent years, 3D Scene Graphs [5], [6] have emerged as a promising framework to model the scene using semantic-relational concepts. Notably, [7] takes a step further by generating the relations between observed objects in real time, although they do not include new entities. Hydra [8] constructs and optimizes the 3D scene graph in real time leveraging loop closures.

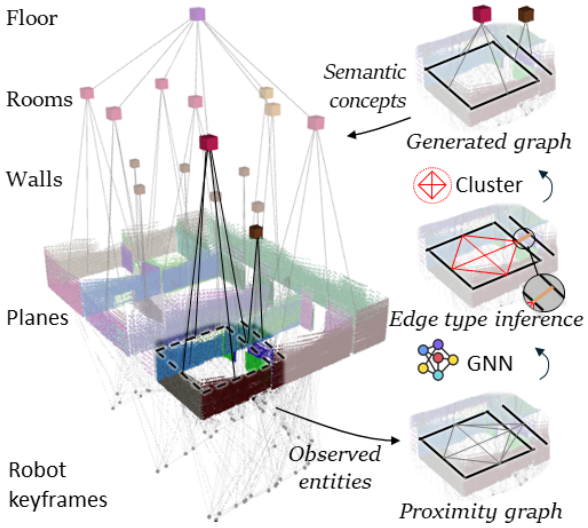


Fig. 1: **System Overview.** We learn how to generate high-level semantic concepts, such as *Rooms* and *Walls* from low-level observed entities, such as *Planes*. In this example, the *Plane* information is retrieved from the low-level layer of *S-Graphs+* [1] and transformed in a *proximity graph* in which a GNN classifies each edge as *same Room* or *same Wall*. The newly inferred edges are further clustered leveraging the existence of cycles, obtaining a new *Room* or *Wall* node for each cluster. By augmenting the *S-Graphs+* [1] with the new nodes and edges, we improve the quality of the map and the estimated camera trajectory.

Coupling them more tightly, *S-Graphs+* [1] generates a four-layered optimizable factor graph comprising a SLAM graph and a 3D scene graph as depicted in Fig. 1. The lowest layer of the graph contains the robot *Keyframes* connected to the second layer, composed of directly observed raw geometric entities i.e. *Planes* (vertical planes named *Walls* in [1]). The upper two layers represent a scene graph, containing semantic *Room* entities relating with the underlying *Planes* and semantic *Floor* entities connecting with the respective *Rooms*. However, *Hydra* [8] and *S-Graphs+* [1] extracts these semantic *Room* and *Floor* entities using ad-hoc hand-tuned algorithms, thus not generalizable to complex and irregularly shaped indoor environments.

To address these limitations, we present a framework to enhance the relational and generalization capabilities of 3D

[1]Authors are with the Automation and Robotics Research Group, Interdisciplinary Centre for Security, Reliability and Trust (SnT), University of Luxembourg. Holger Voos is also associated with the Faculty of Science, Technology and Medicine, University of Luxembourg, Luxembourg. {jose.millan, hriday.bavle, muhammad.shaheer, holger.voos, joseluis.sanchezlopez}@uni.lu

[2]Author is with the University of Amsterdam. m.r.oswald@uva.nl

* This work was partially funded by the Fonds National de la Recherche of Luxembourg (FNR) under the projects 17097684/RoboSAUR and C22/IS/17387634/DEUS.

scene graphs by learning to generate semantic entities based on GNNs [9], [10]. With it, we not only improve the state of the art of *Room* generation but also generate new entities such as *Walls* (two parallel vertical planes), all by a common architecture. As shown in Fig. 1, the observed entities are transformed into a proximity graph used for message passing. The GNN classifies the type of each node into concepts of *same Room*, *same Wall* or none, which are further processed and clustered leveraging the existence of cycles. For each cluster, a new node of *Room* or *Wall* is generated along with the factors that tightly couple them with the underlying *Planes*. Nodes and factors are incorporated back to the *S-Graphs+* [1] to validate its usefulness in semantic-relational SLAM.

As a ground truth graph for training the underlying GNN model, we first generate a synthetic dataset comprising of low-level entities i.e. *Planes* and the higher-level semantic entities i.e. *Rooms* and *Walls* along with their relationships. With no further fine-tuning of the trained GNN model, the results over several simulated and real structured indoor environments demonstrate that our method improves the baselines in detection time, expressiveness, and the number of entities detected.

To summarize, the primary contributions of our paper are:

- A GNN-based framework to generate high-level semantic entities (i.e. *Rooms* and *Walls*) and their relationships with the low-level entities (i.e. *Planes*) in a precise, fast, and versatile manner.
- Integration of the algorithm within the four-layered optimizable *S-Graphs+* framework [1] along with validation in simulated and real datasets with relevant ablations.

II. Related work

A. Semantic Scene Graphs for SLAM

Scene graphs serve as graph models that encapsulate the environment as structured representations. This graph comprises entities, their associated attributes, and the interrelationships among them. In the context of 3D scene graphs, [5] has pioneered the development of an offline, semi-autonomous framework. This framework relies upon object detections derived from RGB images, creating a multi-layered hierarchical representation of the environment and its constituent elements such as cameras, objects, rooms, and buildings. 3D DSG [11] extends this model to account for dynamic entities as humans in the scene. Furthermore, [7] segment instances, their semantic attributes, and the concurrent inference of relationships, in real time. [12], [13] generate open-vocabulary 3D scene graphs by using open-vocabulary object detections and querying their relationships to suitable Large Language Models (LLMs). On the one hand, while the above 3D scene graph frameworks run SLAM/pose estimation backend, they do not utilize the generated scene graph to enhance the SLAM process and on the other hand, they can only infer nearby object relationships and are unable to estimate higher-level entities like *Rooms* and its interconnections with the objects inside.

Hydra [8] focuses on real-time 3D scene graph generation performing a real time room segmentation and interconnecting the objects lying within the rooms while utilizing this information to enhance the loop closure search to finally optimize the entire scene graph. The extension of Hydra in [14] introduces *H-Tree* [15] to characterize the room detection to specific building areas, like kitchens, living rooms, etc. Both the above approaches do not completely integrate the scene graph elements within the SLAM state for simultaneous optimization and utilize an ad-hoc free-space voxel [16] based clustering for room identification, leading to misclassification of room entities in the presence of complex environmental setups.

S-Graphs [1], [17], creates a four-layered hierarchical optimizable graph performing real time room and floor segmentation while concurrently representing the environment as a 3D scene graph. However, the detection of the *Room* entities is also performed using an ad-hoc free-space clustering approach based on [16] limiting its generalizability in different environments. [18] present a 3D scene graph construction for outdoor environment. Although using panoptic detector for detecting object instances, they utilize similar heuristics to extract high-level information about roads and intersections corresponding to rooms and corridors in the indoor 3D scene graphs.

Analyzing the state-of-the-art regarding 3D scene graphs necessitates the requirement of a generic framework for the identification of higher-level semantics like *Rooms*. Thus, to augment the reasoning capability of 3D scene graphs through efficient extraction of high-level semantic concepts and relating them to their low-level counterparts, we present a GNN-based framework integrated within the *S-Graphs+* [1], to infer high-level semantic concepts (*Rooms* and *Walls*) for a given set of low-level entities (*Planes*).

B. Room and Wall Detection

The first step in the generation of higher-level concepts resides in comprehending the interrelations among fundamental geometric entities. The identification of structural configurations corresponding to *Planes* which collectively form *Rooms* and *Walls*, is crucial. Various methods have been explored to address this challenge, encompassing the utilization of pre-existing 2D LiDAR maps [19]–[21], the utilization of 2D occupancy maps within complex indoor environments [22], and pre-established 3D maps [23]–[25]. It should be noted, however, that these approaches exhibit inherent performance constraints and lack real-time operational capabilities. [8] introduce a real-time *Room* segmentation approach using free-space clusters [16] designed to classify different places into *Rooms*. [1] leverages the *Planes* surrounding a given free-space cluster to instantaneously define *Rooms* in real-time. To the best of our knowledge, no analogous methodologies exist based on GNNs to identify both *Room* and *Wall* entities for a given set of *Planes*.

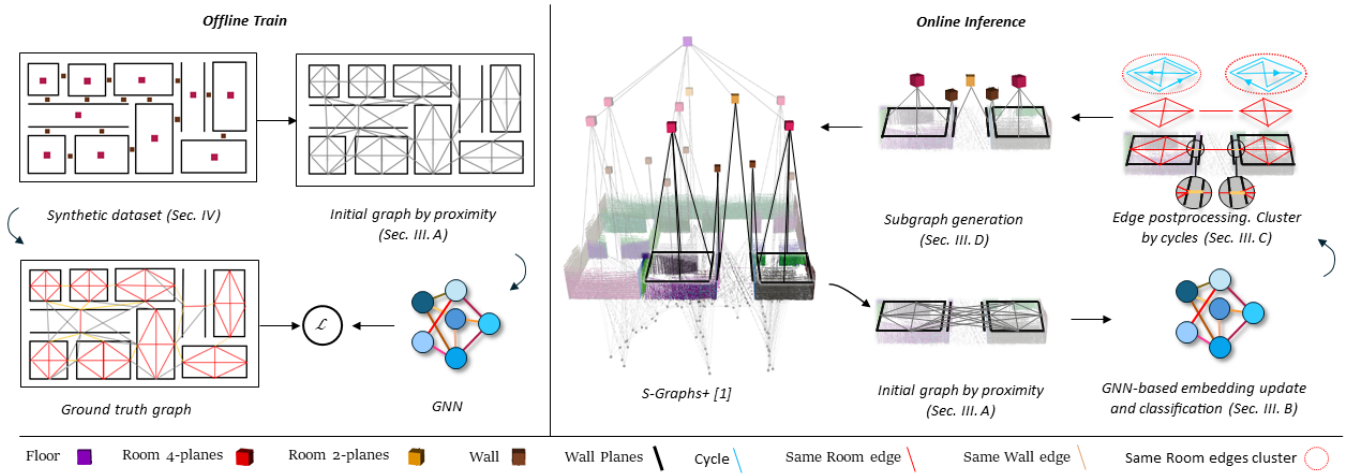


Fig. 2: **System Architecture.** This illustrates the entire process from geometric entities reception to the inclusion of new higher-level entities to *S-Graph*. First, the GNN is trained off-line to update the embedding of nodes and edges and classify the edges. During the SLAM process, the raw low-level nodes are retrieved from *S-Graphs* and connected with a proximity graph. The GNN infers edge classification to *"same Room"*, *"same Wall"*, or nothing. *"same Room"* edges are clustered leveraging cycles. A subgraph is generated for clusters or standalone edges and finally included in *S-Graph*.

III. Methodology

The pipeline of our method is illustrated in Fig. 2. It can be mainly divided into two parts, offline training and online inference. Offline training utilizes a synthetically generated dataset (see Sec. IV) which contains an initial connected graph (see Sec. III-A) based on proximity and the ground truth graph labels for sets of *Planes* forming either *"same Room"* or *"same Wall"* relations. Both the initial graph and the ground truth labels are utilized to train a GNN model.

In the part of online inference, first, the mapped *Plane* features from the low-level layer of *S-Graphs+* [1] are received. These *Plane* features are preprocessed to build a proximity graph and define the initial embedding of nodes and edges (see Sec. III-A) which serve as an input to the trained GNN model. The trained GNN model updates these initial edge embeddings between the nodes, classifying them appropriately into either *"same Room"* or *"same Wall"* relations, further explained in Sec. III-B. Furthermore, these inferred edges are appropriately clustered to generate new nodes of either *Wall* or *Room* entities appropriately connected with the underlying *Planes* (Sec. III-D). Finally, these new nodes and their relationships are integrated into the high-level *Rooms* and *Walls* layer of *S-Graphs+* [1].

A. Initial Graph by Proximity

The *Plane* features before being input to the GNN are preprocessed to generate an initial graph based on the proximity of these features. This module is utilized in both offline training and online inference (see Fig. 2). In case of offline training, the initial *Plane* features in the synthetic dataset are defined as $\boldsymbol{\pi'}_i = [\boldsymbol{n}_i, w_i, c_i]$, where $\boldsymbol{n}_i$ is a normal orientation of the plane defined in closest point form as in [1], w_i and c_i are the width and the centroid of the plane.

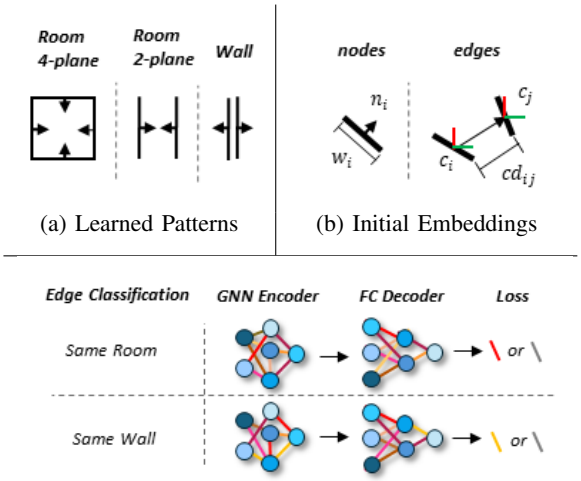
In online inference, *Plane* features are received from *S-Graphs+* [1] in the closest point form as $\boldsymbol{\pi}_i = [\boldsymbol{n}_i, d_i]$, $\boldsymbol{n}_i$ being the normal orientation and d_i being the perpendicular distance to the origin. Furthermore, each *Plane* feature also includes the set of 3D points $\boldsymbol{p}_i \in \mathbb{R}^3$ from the observed point cloud. To obtain the width and centroid of these *Planes* we first flatten and assimilate all the 3D points to a 2D line segment. Subsequently, due to noise in the plane mapping step of *S-Graphs+* [1], there could be the presence of duplicate *Planes* and same *Planes* could be shared between different *Rooms*. To overcome this, we first filter out duplicate planes and then split the *Planes* based on their 2D line intersections with the neighboring *Planes*.

At this point, we have *Planes* represented as $\boldsymbol{\pi'}_i$ in both offline training and online inference. Finally, as described in Fig. 3b, the initial *Plane* embedding $\boldsymbol{v}_i^0$ for the GNN is defined as $[\boldsymbol{n}_i, w_i]$. Given the centroid information for each *Plane* feature new directed edges are generated based on their proximity to the other *Planes*. The embedding for these edges can be defined as $e_{ij}^0 = [\delta(c_j, c_i), cd_{ij}]$, being $\delta(c_j, c_i)$ the relative position of the centroids of i and j *Planes* and cd_{ij} being the closest distance between their segment extremes.

B. GNN-based Embedding Update and Classification

All the edges contained in the initial graph are classified into relations of either *"same Room"*, *"same Wall"*, or none by the GNN-based model trained in the offline training step. For each relation of *"same Room"* and *"same Wall"*, the classification is performed by two separate GNN models. As shown in Fig. 3c and inspired by [7], both models have the same encoder-decoder architecture but have different hyperparameters.

The GNN-based encoder updates the node and edge embeddings separately but interleaved using the latest updates



(a) Learned Patterns (b) Initial Embeddings

(c) Neural Architecture

Fig. 3: a) **Learned Patterns.** Example of the distribution of *Planes* that belong to the same high-level concept. b) **Initial Embeddings.** The definition of the nodes features of the *Planes* is composed by the width (w) and the normal (n) from which it was observed. The edge features are defined by the relative position of the centroids (c) and the closest distance (cd). c) **Neural Architecture.** The classification of both *"same Room"* and *"same Wall"* relations is accomplished by two independent neural networks with similar architecture.

as below:

$$v_i^{l+1} = g_v\big([v_i^l, \max_{j\in\mathcal{N}(i)}(\mathrm{GAT}(v_i^l, e_{ij}^l, v_j^l))]\big) \quad (1)$$

$$e_{ij}^{l+1} = g_e\big([v_i^l, e_{ij}^l, v_j^l]\big) \quad (2)$$

where $g_v(\cdot)$ and $g_e(\cdot)$ are linear layers [26], $\mathcal{N}(i)$ are the neighbors of i_{th} node and $\mathrm{GAT}(\cdot)$ is a Graph Attention Network [9], [10]. Encoder hyperparameters are maintained across the classification of both relations. Eq. (1) and Eq. (2) utilize two hidden layers.

The latest embeddings from the encoder are passed through a multi-layer perceptron decoder as follows:

$$c_{ij} = g_d([v_i^L, e_{ij}^L, v_j^L]) \quad (3)$$

where $g_d(\cdot)$ are three linear layers and L is the last layer of the encoder and c_{ij} is the final binary classification of a specific edge.

C. Edge Postprocessing

We further post-process the binary classification of the edges using appropriate clustering to account for misclassifications from the GNN and select *Planes* that belong to the same *Wall* and *Room* entities. Since a *Wall* node is only related to two *Planes* (see Fig. 3a), only one *"same Wall"* edge is required, avoiding the need for further clustering.

For *"same Room"* relation, the existence of cycles is leveraged. We assume all *Plane* nodes forming a *"same Room"* relationship are connected through at least one cycle and a *Plane* only relates to one *Room*. For each cycle, a *Plane* set is obtained. We prioritize those sets by the following criteria: (1) sets with largest number of *Planes* and (2) highest repetitions of the same *Plane* set. These criteria overcome the issue of the existence of false positives that may lead to the classification of the same *Planes* for two different *Rooms*. In our current implementation, we extract cycles of either two or four *Planes* relating to a *"same Room"* relation (i.e. 2-Plane or 4-Plane Rooms).

D. Subgraph Generation

Finally, after performing the edge post-processing step and filtering out unwanted/misclassified edges, we are left with edge sets relating *Planes* as either *"same Room"* or *"same Wall"*. Based on the inferred relation type we can generate either *Room* or *Wall* nodes. The center of a *Room* node can be computed as follows:

$$\boldsymbol{\rho}_i = \frac{\sum_{j=1}^{\mathcal{N}_i} c_j}{\mathcal{N}_i} \quad (4)$$

where centroid c_j is the centroid of a given *Plane* and $\mathcal{N}_i$ is the set of all planes connected to the *Room* node. *Wall* node center can be computed in the same manner following Eq. (4).

These newly generated nodes along with centers and their connected *Planes* are incorporated into the *Rooms* and *Walls* layer of the optimizable factor graph of *S-Graphs+* [1]. The cost function for *Room* with center $\boldsymbol{\rho_i}$ and its four *Planes* $\boldsymbol{\pi}_n$ can be given as:

$$c_{\boldsymbol{\rho}}(\boldsymbol{\rho}_i, [\boldsymbol{\pi}_1, \boldsymbol{\pi}_2, \boldsymbol{\pi}_3, \boldsymbol{\pi}_4]) = \|\hat{\boldsymbol{\rho}}_i - f(\hat{\boldsymbol{\pi}}_1, \hat{\boldsymbol{\pi}}_2, \hat{\boldsymbol{\pi}}_3, \hat{\boldsymbol{\pi}}_4)\|^2_{\boldsymbol{\Lambda}} \quad (5)$$

$f(\cdot)$ is the function that maps the room center using the four *Planes*.

The cost function for a *Room* $\boldsymbol{\kappa}_i$ with its two *Planes* $\boldsymbol{\pi}_n$ can be given as:

$$c_{\boldsymbol{\kappa}_i}(\boldsymbol{\kappa}_i, [\boldsymbol{\pi}_1, \boldsymbol{\pi}_2, \mathbf{p}_i]) = \|\hat{\boldsymbol{\kappa}}_i - f(\hat{\boldsymbol{\pi}}_1, \hat{\boldsymbol{\pi}}_2, \mathbf{p}_i)\|^2_{\boldsymbol{\Lambda}} \quad (6)$$

where $\mathbf{p}_i \in \mathbb{R}^3$ is the average of the centroid of the two planes and $f(\cdot)$ is the function that maps the room center using the two *Planes* and points $\mathbf{p}_i$. *Wall* node is incorporated into the factor graph using cost $c_{\boldsymbol{w}_i}$ which follows the same Eq. 6. In both Eq. 5 and Eq. 6, $\hat{(\cdot)}$ refers to the estimated values of the variables in the optimizable graph.

IV. Training with Synthetic Dataset

We generate synthetic dataset that provides ground truth labels of relationships between target entities, avoiding the need for extracting and labelling real world datasets. We developed the synthetic dataset focusing on replicating common *Plane* structure of usual indoor environments. As explained in Sec. III-A, the *Planes* $\boldsymbol{\pi'}_i$ is defined in closest point form with its width and centroid. The dataset also contains the center of the *Rooms* $\boldsymbol{\rho}_i$ and *Walls* $\boldsymbol{w}_i$ appropriately relating to the underlying *Planes*. Furthermore this data is augmented with several layers of randomization in size, position, and orientation when creating *Rooms* and *Planes*. Ground truth edges between *Planes* are automatically tagged as *same Room* or *same Wall* concepts and included along with negative tagged edges with the 15 closest neighbor *Plane* nodes for a given *Plane* node. During the training

process, 800 different layouts are used for backpropagation during each one of the 35 epochs. Xavier uniform initialization [27] is used for the learnable parameters.

One of the advantages of our approach is that, after the initial training with the synthetic dataset, the trained GNN model is used on real data without further retraining, with no additional tuning of parameters and the same applied normalization. This gives the flexibility of training our model for different kinds of complex environment without the need for tedious data collection and labelling process.

V. Experimental Results

A. Methodology

Our work is validated in both simulated and real datasets presented in [1] collected using a Velodyne VLP-16 3D LiDAR over different indoor environments comprising office spaces and constructions sites detailed in Tab. I. We compare the graph expressiveness through precision and recall of our algorithm with the ad-hoc *Room* detection algorithms presented in Hydra [8] and *S-Graphs+* [1], which we call *Hydra RS* and *S-Graphs+ RS* respectively. In addition, our algorithm is tested for its precision and recall in both Conservative (C) or Greedy (G) scenarios for *Room* generation. In the *Ours G* scenario lower threshold value is applied in the *same Room* edge classification type when compared to the *Ours C* scenario. We also compare the First Detection Time of our approach to the room segmentation of the baselines.

Furthermore, we integrate our algorithm within the *S-Graphs+* [1] framework replacing its ad-hoc room segmentation algorithm while naming it *Ours (Int.)*. We compare the pose and map accuracy of *Ours (Int.)* with the baseline *S-Graphs+* [1] algorithm. Additionally, we ablate *Ours (Int.)* method incorporating only *Rooms* without *walls* calling it *Ours (Int.) (rooms only)*. Given the lower variance in precision/recall for *Ours C* approach, we choose it for in the *Ours (Int.)* approach for further validations of ATE and MMA. The example scenarios are preceded with *S* and *R* to differentiate simulation and real datasets respectively.

In all the experiments, no fine-tuning of the specified network hyper-parameters is applied, as the empirically chosen ones during the training (Sec. IV) suffice for all cases.

Simulated Data. We performed a total of five experiments on simulated datasets denoted as *SC1F1*, *SC1F2*, *SE1*, *SE2*, and *SE3*. *SC1F1* and *SC1F2* are generated from the 3D meshes of two floors of actual architectural plans, while *SE1*, *SE2* and *SE3* simulate typical indoor environments with varying *Room* configurations. In these experiments we compute the graph expressiveness of our approach and baselines in terms of precision/recall for *Room* and *Wall* detection given the availability of ground truth *Rooms* and *Walls*.

Additionally, In order to assess the pose and map accuracy of *Ours (Int.)* with the *S-Graphs+* [1] baseline we report Average Trajectory Error (ATE) and Map Matching Accuracy (MMA). The ATE is calculated against the ground truth provided by the simulator and the MMA is computed utilizing the ground truth map available for each scenario.

TABLE I: **Scenes Description.** Enumeration of all simulated and real scenes included in the validation. *Room* shapes can be squared, L-shaped, elongated, or corridors.

Scene	World	Description
SC1F1	Simulated	1 L-shaped, 1 squared rooms and 2 corridors.
SC1F2	Simulated	5 squared, 2 elongated rooms and 1 corridor.
SE1	Simulated	6 squared rooms and 3 corridors.
SE2	Simulated	5 squared and 2 elongated rooms.
SE3	Simulated	22 squared rooms and 4 corridors.
RC1F1	Real	1 L-shaped, 1 squared rooms and 2 corridors.
RC1F2	Real	5 squared, 2 elongated rooms and 1 corridor.
RC2F2	Real	7 squared, 2 L-shaped and 3 elongated rooms.
RC3F2	Real	9 squared, 3 L-shaped rooms and 2 corridors.

Real Dataset. We conducted four real experiments in two different construction sites. *RC1F1* and *RC1F2* are conducted on two floors of a small construction site. *RC2F2* and *RC3F2* are conducted in two other construction sites with larger areas. First, we validate the graph expressiveness of *Rooms* and *Walls* detection of our approach and baselines with the ground truth *Rooms* and *Walls*. Second, to validate the accuracy of each method in all real-world experiments, we report the MMA of the estimated 3D maps in comparison to the ground truth 3D map generated from the architectural plans.

B. Results and Discussion

Graph Expressiveness. Fig. 4 showcases the precision/recall performance on the detection of *Rooms* for simulated and real scenarios. Room segmentation algorithms used in *Hydra* [8] and *S-Graphs+* [1] are compared with our ablated module *(rooms only)*. Our ablation also assesses the relaxation of the GNN threshold to classify *same Room* edges i.e Greedy (G) and Conservative (C) approach.

For simulated/real scenarios, *Ours C* approach improves the average precision over *Hydra* [8] by $37\%/9.5\%$ and the recall by $0\%/16\%$ in simulated and real scenarios respectively. With respect to *S-Graphs+* [1], *Ours C* provides the precision in simulated/real scenarios as $16\%/0\%$. In terms of recall, *Ours C* maintains the same average recall in simulated scenarios while is in real scenarios the average recall deteriorates by 11%. Finally, although *Ours G* with respect to *Ours C* provides average improvements in recall by $21\%/37\%$, *Ours G* degrades in average precision by $-8\%/2\%$.

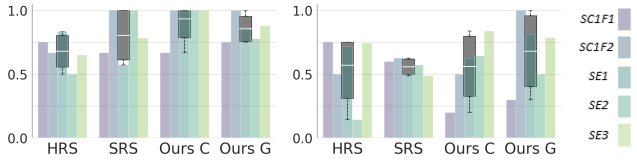
Fig. 5 presents qualitative results of graph expressiveness for experiments SE3 and RC2F2. As can be seen from the figure, although in SE3 the performance of *Hydra RS* [8] is maintained on the average including 4 false positives, in RC2F2, most of the rooms present points misplaced over the whole area given the complexity of the real environment and the noise in the LiDAR measurements. Note from the figure that while *S-Graphs+ RS* [1] is able to include *2-Plane Rooms* (orange squares) in both simulated and real scenarios, *Ours C* is able to provide segment higher quantity of *4-Plane Rooms* (pink squares) in both simulated and real datasets

TABLE II: **Graph Expressiveness.** measured by precision and recall for *Wall* detection of our approach in different simulated and real scenes.

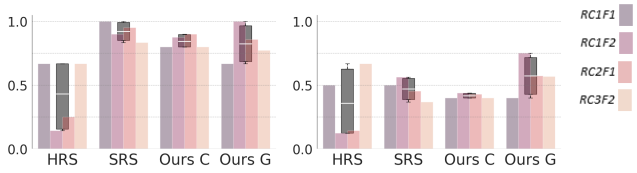
	Dataset					
Metric	*SC1F1*	*SC1F2*	*SE1*	*SE2*	*SE3*	Avg
Precision	1.00	1.00	1.00	1.00	1.00	1.00
Recall	0.80	1.00	0.87	0.43	0.89	0.80
	C1F1r	*C1F2r*	*C2F2r*	*C3F2r*		Avg
Precision	1.00	1.00	1.00	1.00		1.0
Recall	0.80	0.75	0.82	0.84		0.80

with higher precision, additionally it is able to identity and segment the *Wall* entities.

On its side, *Wall* segmentation can not be compared to these baselines as they do not segmented by them. Thus in case *Wall* segmentation we present Tab. II providing precision/recall results compared to the ground truth data. It is worth mentioning that precision is always maintained at 1.0 across simulated and real scenarios. Recall is over 75% in all scenarios but in a simulated one. On the contrary to *Rooms*, *Walls* present a similar structure, which simplifies the task of finding the patterns by the GNN thus providing better results.



(a) Simulation, precision (b) Simulation, recall



(c) Real, precision (d) Real, recall

Fig. 4: **Graph Expressiveness.** measured by precision and recall in *Room* detection for the in *Hydra RS* [8] (HRS) and in *S-Graphs+ RS* [1] (SRS), as baselines. Those are compared with *Our* Conservative *(C)* and Greedy *(G)* approach in five simulated and four real scenes. For each approach, experiments are aggregated and the metrics are presented by mean, std, maximum, and minimum.

First Detection Time. Tab. III provides a comprehensive overview of the time required by each module to accomplish detection of the first *Room*. Every experiment is started inside the construction site, that is inside of a *Room*. *Ours C* is compared with *S-Graphs+ RS* [1], demonstrating a drastic average improvement of 84.3% in the simulated datasets and 62.7% in the real datasets. This is due to the fact that the room segmentation of *S-Graphs+* needs to observe more

TABLE III: **First Detection Time (FDT)** [s] of *Rooms* on simulated and real data. Our method is substantially faster than the baseline. The best results are boldfaced.

	Dataset					
	Computation Time (mean) [ms]					
Module	*SC1F1*	*SC1F2*	*SE1*	*SE2*	*SE3*	Avg
S-Graphs+ RS [1] (baseline)	76.0	19.0	19.0	115.0	160.0	77.8
Ours C	**2.7**	**2.8**	**10.2**	**8.2**	**37.1**	**12.2**
	RC1F1	*RC1F2*	*RC2F2*	*RC3F2*		Avg
S-Graphs+ RS [1] (baseline)	19.0	25.5	367.0	308.0		179.9
Ours C	**1.7**	**2.6**	**54.0**	**210.2**		**67.1**

TABLE IV: **Absolute Trajectory Error (ATE)** [m], of *S-Graph+* with different detection modules on simulated data. The best results are boldfaced. Both of our approaches improve the ATE of the baseline in the most complex scenes Considering semantic relations between both walls and room is always better or equal than only rooms.

	Dataset [m $\times 10^{-2}$]					
Module	*SC1F1*	*SC1F2*	*SE1*	*SE2*	*SE3*	Avg
S-Graphs+ [1] (baseline)	2.72	6.93	**1.47**	**1.36**	2.98	3.09
Ours (Int.) (rooms only)	2.72	6.58	1.55	1.57	2.23	2.93
Ours (Int.)	**2.71**	**6.35**	1.54	1.56	**2.23**	**2.88**

map points until a free-space cluster can be inferred and then associate the cluster with the mapped *Planes*, while our method utilizing only the mapped *Planes* as input succeeds in finding the rooms faster.

Average Trajectory Error. The ATE for the simulated experiments is presented in Tab. IV. *Ours (Int.)* approach for *Room* and *Wall* detection demonstrates an improvement of 6.8% with respect to *S-Graphs+* [1] baseline. The ablation of *walls* in *Ours (Int.) (Rooms only)* shows an improvement of 5.2% even though no *Wall* entities are leveraged. Note that in simulated data resembling to real construction sites i.e., *SC1F1*, *SC1F2* and *SE3*, our approach can of robustly detect *Rooms* improving the final ATE. However, when complexity and size decrease, *S-Graphs+* presents a similar or better performance due to the same number of detected rooms.

Map Matching Accuracy. Tab. V presents the MMA for simulated and real experiments. *Our (Int.)* presents an average improvement of 1.8% with respect to *S-Graphs+* [1] baseline. The ablation of *Walls* in *Ours (Int.) (Rooms only)* still represents an improvement of 0.3%. The results demonstrate that, even with an already low MMA in the baseline, the inclusion of better representations in *Our (Int.)* still presents a notable improvement while enhancing its expressiveness.

Limitations. The number of higher-level entities in the synthetic dataset is limited to *Rooms* and *Walls*, which limits the graph expressiveness of the generation. In addition, the layouts only contain a limited variety of shapes, which leads

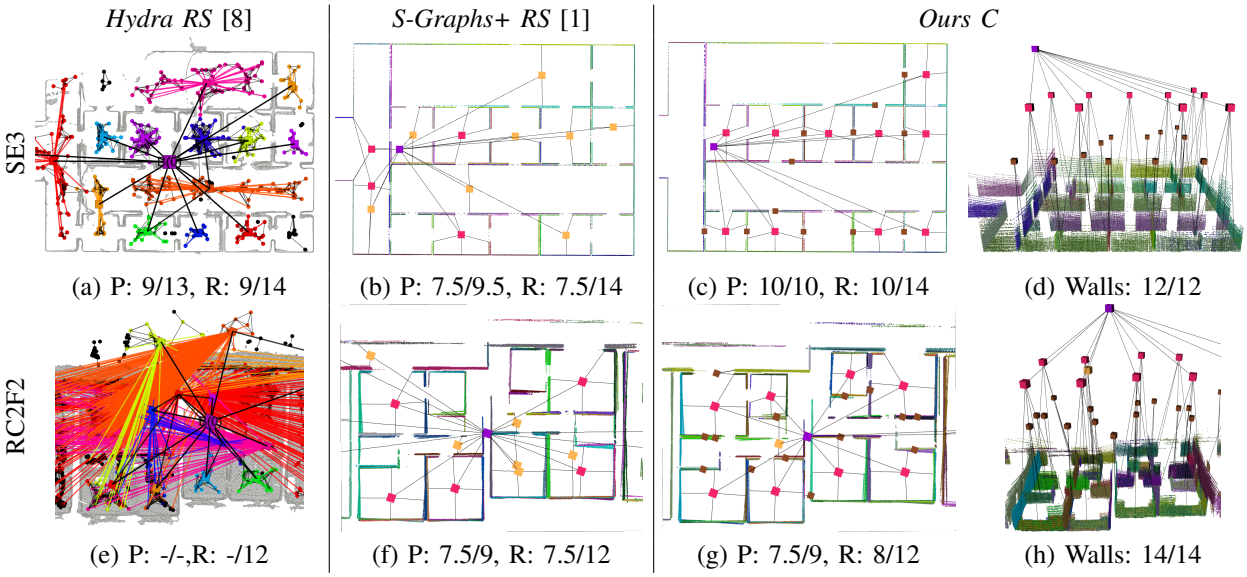


Fig. 5: **Graph Expressiveness.** Compared qualitatively over *Hydra RS* [8], *S-Graphs+ RS* [1] and *Ours C* on two example datasets, SE3 simulated and RC2F2 real. *Room* generation is presented in the first three columns while indicating precision (P) and recall (R) ratios. The fourth column, presents a 3D perspective to demonstrate the *Rooms* as well as *Wall* generation but *Ours C*.

TABLE V: **Map Matching Accuracy (MMA)** [m] of *S-Graphs+* with different detection modules on simulated and real data. The best results are boldfaced. In all real scenes and all but one simulated, the MMA is outperformed by our approach, including the ablated method.

	Dataset [m $\times 10^{-2}$]							
Module	*SE2*	*SC1F1*	*SC1F2*	*RC1F1*	*RC1F2*	*RC2F2*	*RC3F2*	Avg
S-Graphs+ [1]	27.53	**7.40**	7.55	32.60	18.75	17.8	44.86	22.35
Ours (Int.) (rooms only)	27.52	7.61	7.53	**32.54**	18.64	17.8	44.35	22.28
Ours (Int.)	**27.51**	7.60	**7.51**	32.67	**17.79**	**17.27**	**43.31**	**21.95**

to a lower performance on edge cases in real scenarios. Furthermore, edge postprocessing in Sec. III-D requires the maximum cluster size for each entity of 1 for *Walls* and any for *Rooms* to be set in advance.

VI. Conclusion

We presented a novel approach based on Graph Neural Networks for inferring high-level semantic-relational concepts such as *Rooms* and *Walls* to enrich the 3D scene graph for a given environment. Our method unfolds in several steps: (a) GNN-based Edge Inference: Initially, we infer *"same Room"* and *"same Wall"* edges among the observed low-level *Planes*. (b) Clustering: Subsequently, we process these inferred edges to cluster *Planes* corresponding to each higher-level concept. (c) Subgraph Generation: Finally, we represent these clusters in the form of a subgraph to form new *Room* and *Wall* nodes, finally incorporating them into the existing factor graph within the *S-Graphs+* [1] framework.

In comparison to the current baselines for *Room* segmentation, our approach exhibits a notable reduction of 67% of detection time, expressiveness, and generalization attributes given the fact that *Walls* entities are not yet automatically detected. Importantly these enhancements contribute to a better final pose (6.8%) and map accuracy (1.8%).

In future research, we expect to expand the expressiveness of our dataset with new entities useful for the SLAM such as *Floors* and more complex scenes. Furthermore, we envision the GNN-based generation of entities (*Rooms* and *Walls*) and their relationships in an end-to-end manner without the need of a postprocessing step.

References

[1] 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*, 2023.

[2] 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. [Online]. Available: https://www.mdpi.com/1424-8220/23/10/4849

[3] P. Kremer, H. Bavle, J. L. Sanchez-Lopez, and H. Voos, "S-nav: Semantic-geometric planning for mobile robots," *arXiv preprint arXiv:2307.01613*, 2023.

[4] C. Huang, O. Mees, A. Zeng, and W. Burgard, "Visual language maps for robot navigation," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, London, UK, 2023.

[5] I. Armeni, Z.-Y. He, J. Gwak, A. R. Zamir, M. Fischer, J. Malik, and S. Savarese, "3D Scene Graph: A structure for unified semantics, 3D space, and camera," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 5664–5673.

[6] 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 Transactions on Cybernetics*, vol. 50, no. 12, pp. 4921–4933, dec 2020.

[7] S.-C. Wu, J. Wald, K. Tateno, N. Navab, and F. Tombari, "Scenegraphfusion: Incremental 3d scene graph prediction from rgb-d sequences," in *IEEEConference on Computer Vision and Pattern Recognition*, 2021.

[8] N. Hughes, Y. Chang, and L. Carlone, "Hydra: A real-time spatial perception system for 3d scene graph construction and optimization," in *Robotics: Science and Systems*, 2022.

[9] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and S. Y. Philip, "A comprehensive survey on graph neural networks," *IEEE transactions on neural networks and learning systems*, vol. 32, no. 1, pp. 4–24, 2020.

[10] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph attention networks," *arXiv preprint arXiv:1710.10903*, 2017.

[11] A. Rosinol, A. Gupta, M. Abate, J. Shi, and L. Carlone, "3D dynamic scene graphs: Actionable spatial perception with places, objects, and humans," in *Robotics: Science and Systems (RSS)*, 2020.

[12] Q. Gu, A. Kuwajerwala, S. Morin, K. M. Jatavallabhula, B. Sen, A. Agarwal, C. Rivera, W. Paul, K. Ellis, R. Chellappa, C. Gan, C. M. de Melo, J. B. Tenenbaum, A. Torralba, F. Shkurti, and L. Paull, "Conceptgraphs: Open-vocabulary 3d scene graphs for perception and planning," 2023.

[13] S. Koch, N. Vaskevicius, M. Colosi, P. Hermosilla, and T. Ropinski, "Open3dsg: Open-vocabulary 3d scene graphs from point clouds with queryable objects and open-set relationships," 2024.

[14] 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," *arXiv preprint arXiv:2305.07154*, 2023.

[15] R. Talak, S. Hu, L. Peng, and L. Carlone, "Neural trees for learning on graphs," *Advances in Neural Information Processing Systems*, vol. 34, pp. 26 395–26 408, 2021.

[16] H. Oleynikova, Z. Taylor, R. Siegwart, and J. Nieto, "Sparse 3d topological graphs for micro-aerial vehicle planning," 2018.

[17] 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.

[18] E. Greve, M. Büchner, N. Vödisch, W. Burgard, and A. Valada, "Collaborative dynamic 3d scene graphs for automated driving," 2023.

[19] R. Bormann, F. Jordan, W. Li, J. Hampp, and M. Hägele, "Room segmentation: Survey, implementation, and analysis," *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1019–1026, 2016.

[20] 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 *2018 IEEE International Conference on Robotics and Automation (ICRA)*, 2018, pp. 4993–4999.

[21] 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. 101 530–101 542, 2021.

[22] 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 Robotics and Automation Letters*, vol. 7, no. 3, pp. 7974–7981, 2022.

[23] I. Armeni, O. Sener, A. R. Zamir, H. Jiang, I. Brilakis, M. Fischer, and S. Savarese, "3D Semantic Parsing of Large-Scale Indoor Spaces," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 1534–1543.

[24] R. Ambruş, S. Claici, and A. Wendt, "Automatic Room Segmentation From Unstructured 3-D Data of Indoor Environments," *IEEE Robotics and Automation Letters*, vol. 2, no. 2, pp. 749–756, 2017.

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

[26] F. Murtagh, "Multilayer perceptrons for classification and regression," *Neurocomputing*, vol. 2, no. 5-6, pp. 183–197, 1991.

[27] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proceedings of the thirteenth international conference on artificial intelligence and statistics*. JMLR Workshop and Conference Proceedings, 2010, pp. 249–256.