GNN-based Metric-Semantic Factor Graph Generation

Jose Andres Millan-Romera[1], Hriday Bavle[1], Muhammad Shaheer[1],
Holger Voos[1], and Jose Luis Sanchez-Lopez[1]

Abstract— **Understanding the relationships between geometric structures and semantic concepts is crucial for building accurate models of complex environments. In indoors, certain spatial constraints, such as the relative positioning of planes, remain consistent despite variations in layout. This paper explores how these invariant relationships can be captured in a graph SLAM framework by representing high-level concepts like *rooms* and *walls*, linking them to geometric elements like planes through an optimizable factor graph. Several efforts have tackled this issue with add-hoc solutions for each concept generation and with manually-defined factors.**

This paper proposes a novel method for metric-semantic factor graph generation which includes defining a semantic scene graph, integrating geometric information, and learning the interconnecting factors, all based on Graph Neural Networks (GNNs). An edge classification network (G-GNN) sorts the edges between *planes* into *same room*, *same wall* or none types. The resulting relations are clustered, generating a *room* or *wall* for each cluster. A second family of networks (F-GNN) infers the geometrical origin of the new nodes. The definition of the factors employs the same F-GNN used for the metric attribute of the generated nodes. Furthermore, share the new factor graph with the *S-Graphs+* algorithm, extending its graph expressiveness and scene representation with the ultimate goal of improving the SLAM performance. The complexity of the environments is increased to N-*plane* rooms by training the networks on L-shaped *rooms*. The framework is evaluated in synthetic and simulated scenarios as no real datasets of the required complex layouts are available.

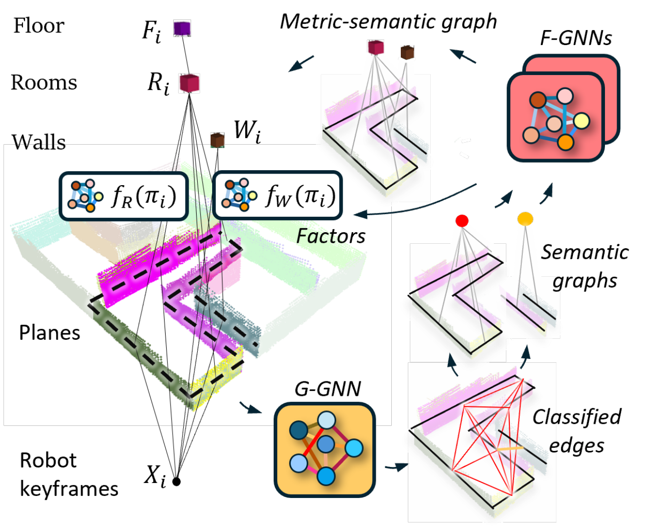

Fig. 1: **System Overview.** An initial graph by proximity is set from the *plane* nodes inside *S-Graphs+* [1]. G-GNN classifies the edges into *same room* or *same wall*. Those edges are clustered and a *room* or *wall* semantic nodes is generated for each cluster. The new nodes receive a geometric definition from its F-GNN depending on the concept. The metric-semantic nodes are incorporated into *S-Graphs+* [1] along with the factors for the know edges defined by the F-GNNs.

I. Introduction

Understanding the invariant relationships between high-level geometric and semantic concepts in complex scenes is essential for robots to accurately integrate them into a realistic model. In indoor, man-made environments, despite the complexity of the layout, certain constraints, such as the relative positions of planes, remain constant. In the context of graph SLAM, these invariant relationships are represented around a high-level concept such as a *room* and connected to the *planes* through factors within an optimizable graph.

The main steps to define the higher levels of a factor graph are: (1) the definition of the semantic graph or scene graph, (2) the inclusion of geometrical information of the nodes and (3) the constraints, or factors, amongst them.

Several efforts have been accomplished in the steps required for the generation of semantic graphs. [2] generates semantic relations between objects in an end-to-end manner. [3] presents an ad-hoc algorithm to generate the semantic nodes of the scene graph, such as *rooms*, with its edges to the objects. In [4], relations amongst objects (*planes*) are learned and further clustered to find the set of *planes* belonging to the same concepts (*rooms* or *walls*).

Fewer efforts have been tailored to the definition of continuous attributes, e.g. a *room* origin for generated nodes. NANG [5] employs adversarial learning to generate attributes that align with graph structure. [6] proposes a novel architecture that extends traditional GNNs by leveraging factor graphs to capture complex dependencies among multiple variables. However, no graph generation method jointly generated new nodes, edges and their continuous attributes.

Notably, *S-Graphs+* [1] proved that tightly coupling the semantics of the scene graph with the geometry as optimizable factors highly improves mapping and trajectory accuracy. The authors define type-wise manually-defined factors for

[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. {jose.millan, hriday.bavle, muhammad.shaheer, holger.voos, joseluis.sanchezlopez}@uni.lu

* 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.

room-plane and *wall-plane* edges. 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*.

In this work, we propose improvements on the three steps of factor graph generation to create a learnable end-to-end factor graph generation. First, edge classification in [4] is merged in a single GNN-based neural network (G-GNN). The classified edges are grouped into clusters of higher length for the *room* and *wall* semantic node generation, improving the graph expressiveness. Second, to define its continuous attributes, the inference of the origin of *rooms* and *walls* is defined by another GNN-based architecture (F-GNN) which also defines the constraint in the *room-plane* and *wall-plane* factors. Finally, to demonstrate the generalization capabilities of our proposed method, we break the constraint of *S-Graphs+* [1] and [4] of *rooms* composed by up to 4 *planes* and include *room-plane* factors which handle seamlessly from 2 to N *planes* by including L-shaped rooms in the training dataset. All of this is accomplished without a penalization in the overall computation time.

To summarize, the primary contributions of our paper are:

- GNN-based simultaneous generation of heterogeneous graph with *room* and *wall* node types, the origin as a continuous attribute and its adjacency with plane nodes.
- Definition of GNN-based factors for *room-plane* and *wall-plane* edge in *S-Graphs+* [1].
- Inclusion in *S-Graphs+* [1] of *rooms* related to N number of *planes*.

II. Related work

A. Semantic Scene Graphs for SLAM

Scene graphs are graph models that capture environments as structured representations, consisting of entities, their attributes, and the relationships between them. [7] introduced an offline, semi-autonomous framework based on object detections from RGB images, generating a hierarchical, multi-layered representation of environments, encompassing elements such as cameras, objects, rooms, and buildings. 3D DSG [8] further enhances this model by incorporating dynamic entities like humans into the scene. Additionally, [2] uses the semantic attributes of segmented instances to infer in real-time their relationships. Recent works such as [9], [10] employ open-vocabulary object detections and leverage large language models (LLMs) to generate open-vocabulary 3D scene graphs by querying the relationships between detected entities.

Hydra [3] addresses real-time 3D scene graph generation by performing real-time *room* segmentation and linking objects within *rooms*, using this information to enhance loop closure searches, thus optimizing the entire scene graph. The Hydra extension in [11] introduces *H-Tree* [12] to categorize rooms into specific building areas, such as kitchens and living rooms. However, both approaches fail to fully integrate scene graph elements into the SLAM state for simultaneous optimization, relying on an ad-hoc free-space voxel clustering method [13] for room identification, which can lead to misclassifications in complex environments.

S-Graphs [1], [14] constructs a four-layered hierarchical graph, facilitating real-time *room* and *floor* segmentation while representing the environment as a 3D scene graph. However, similar to previous methods, it employs an ad-hoc free-space clustering technique [13] to detect *room* entities, which limits its adaptability to diverse environments.

In contrast, [15] presents a 3D scene graph construction for outdoor environments. While using a panoptic detector for object instance detection, they apply similar heuristics to extract high-level information about roads and intersections, comparable to the rooms and corridors found in indoor 3D scene graphs.

B. Graph Generation with Continuous Node Attributes

Node Attribute Neural Generator (NANG) [5] employs adversarial learning to generate attributes that align with graph structure. Recent work has explored disentangled representations to separately model topology and node attributes, improving generation quality and interpretability [16]. Additionally, some methods focus on controlled generation, allowing users to specify relationships between node classes, attributes, and topology [5].

However, handling continuous attributes remains challenging, especially in maintaining coherence with graph topology at scale. Furthermore, the literature misses a framework capable of generating nodes and edges along with their semantic and continuous attributes in a learnable end-to-end manner.

C. Factor Graph Definition

Factor graphs have emerged as a powerful tool for modeling and solving large-scale inference problems in robotics, particularly in simultaneous localization and mapping (SLAM) [17]. They provide a structured representation that captures the relationships between variables and observations, facilitating efficient optimization and inference.

By integrating scene graph information into the factor graph, robots can interpret and interact with their surroundings more effectively [18]. The generation of factor graphs involves defining various factors that represent different types of measurements and constraints, such as prior factors, odometry factors, and visual factors [19]. Factor graphs have been extensively applied in various domains, including robotics for SLAM, localization, and mapping; computer vision for layout generation and scene understanding; and navigation for efficient estimation, such as the mentioned *S-Graphs+* [1], which employs a fully optimizable graph with manually defined factors.

[6] proposes a novel architecture that extends traditional GNNs by leveraging factor graphs to capture complex dependencies among multiple variables, thereby enabling efficient inference and representation learning in various applications, including probabilistic graphical models and belief propagation methods. However, we are not aware of any work about the definition of geometric factors as learnable neural networks for scene graphs in robotics.

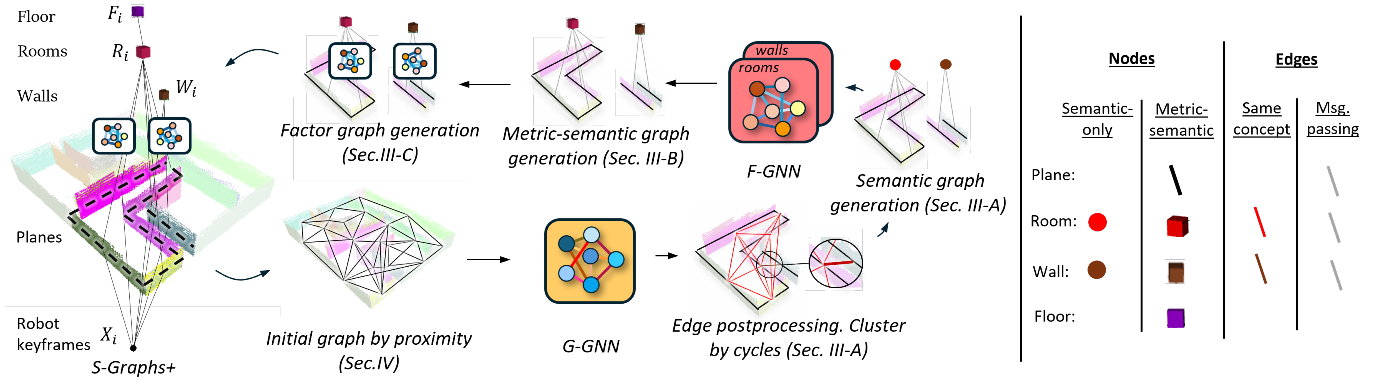
Floor
F_i
Rooms
R_i
Walls
W_i
Planes
Robot keyframes
X_i
S-Graphs+
Factor graph generation (Sec.III-C)
Metric-semantic graph generation (Sec. III-B)
walls
rooms
F-GNN
Semantic graph generation (Sec. III-A)
Initial graph by proximity (Sec.IV)
G-GNN
Edge postprocessing. Cluster by cycles (Sec. III-A)
Nodes
Semantic-only
Metric-semantic
Edges
Same concept
Msg. passing
Plane:
Room:
Wall:
Floor:

Fig. 2: **System architecture.** Every node in the *plane* layer of *S-Graphs+* [1] is connected with its K neighbours, building the initial graph by proximity. It is fed to the G-GNN which classifies the edges into *same room* or *same wall*. Those are separately clustered, leveraging cycles for *same room* ones, and generating a *room* or *wall* semantic nodes for each cluster. Afterwards, the geometric origin of the new nodes is defined by its F-GNN depending on the concept. A new factor is included for every new node.Every The metric-semantic factor graph is incorporated into *S-Graphs+* [1].

III. Methodology

Our metric-semantic factor graph generator (see Fig. 2) subdivides the process into three different steps. First, semantic graph generation defines new semantic-only nodes (*rooms* and *walls*) and the edges connecting them to other existing nodes (*planes*). Second, metric graph generation provides a continuous attribute definition to the semantic-only nodes. Third, every generated edge is defined in the form of a factor bounding the geometrical definition of the connecting nodes required for the graph optimization.

A. Semantic Graph Generation

The initial graph of *plane* nodes is defined by proximity as described in [4], along with its node and edge attributes. We merge the two models they use into a single GNN-based architecture predicting the class of all edges into *same room*, *same wall* or *none* (G-GNN). Its architecture is composed of two phases in a encoder-decoder fashion. The first one is a message-passing encoder which updates node and edge embeddings in two hops. The second stage is a fully-connected decoder with a neuron per class in the final layer.

The clustering algorithm for *rooms* remains the same as in [4] with the only difference that now the maximum cycle length is set to 10 as rooms with a higher number of *planes* is enabled. A new semantic node is crated for each *plane* cluster associated to a *room* and for each *plane* pair associated to a *wall*. Its geometric definition will be presented in the next subsection III-B.

B. Metric Graph Generation

The semantic graph is extended with the definition of the origins of *room* and *wall* nodes, completing it as a metric-semantic graph. That origin is inferred by another GNN-based architecture named F-GNN as depicts Fig. 2. Two different models, one for *room* origins and one for *wall* origins are trained with the same architecture but different hyperparameters.

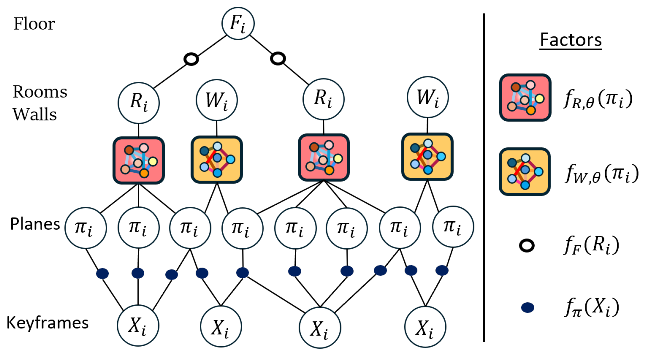

Fig. 3: **Factor Graph.** In the hierarchical structure, every connection between two nodes (circles) at subsequent levels is defined by a factor. *Floor-room* and *plane-keyframe* factors are manually defined as described in *S-Graphs+* [1]. However, *room-plane* and *wall-plane* factors are defined by a F-GNN depending on the node types. The full graph is jointly optimized as described in *S-Graphs+* [1].

For each *room* or *wall* semantic nodes, a new initial graph is defined by its neighboring *plane* nodes connected to the semantic node. In contrast to the input to G-GNN, the *plane* node attributes are its infinite plane definition with normal (observation side) along with its centroid, whereas no attribute is used for edges. The architecture is as well divided in an encoder-decoder manner. The decoder is a single-hop message passing that generates the embedding of the semantic node. That embedding is passed to a fully connected neural network with two neurons in the final layer that infer the x and y components of the origin.

The resultant metric-semantic graph is incorporated into the full *S-Graphs+* [1]. This graph generation is accomplished in an incremental manner as new *planes* are observed, which yield to new *room* and *wall* nodes which extend the already populated high-level layers of the hierarchical graph.

C. Factor Graph Generation

S-Graphs+ [1] is a fully factorized graph. This means that every single edge between nodes must be defined as a factor, i.e. a geometrical constrain bounding the the geometrical definitions of the connected nodes as depicted in Fig. 2. As it is what the F-GNNs did in III-B, it is reused for the definition of the factor, as shown in Fig. 3. While the definitions of *plane-keyframe* and *floor-room* factors remain as in [1], we redefine those for *room-plane* and *wall-plane* with its respective F-GNN.

The origin of a *room* node can be computed as follows:

$$\boldsymbol{\rho}_i = f_{R,\theta}(\boldsymbol{\pi}_1, ..., \boldsymbol{\pi}_j) \forall \boldsymbol{\pi}_j \in \boldsymbol{\mathcal{P}}_{R_i} \tag{1}$$

where ρ_i is the origin of *room* i, π_j is the definition of *plane* j, $\mathcal{P}_{R_i}$ is the set of *planes* connected to the *room* R_i, and $f_{R,\theta}(\cdot)$ is the graph neural network F-GNN for *room* origins.

The cost function $c_{\boldsymbol{\rho}}$ for *room* with origin $\boldsymbol{\rho_i}$ and its set of planes *planes* $\mathcal{P}_{R_i}$ can be given as:

$$c_{\boldsymbol{\rho}}(\boldsymbol{\rho}_i, [\boldsymbol{\pi}_1, ..., \boldsymbol{\pi}_j]) = \\ \|\hat{\boldsymbol{\rho}}_i - f_{R,\theta}(\hat{\boldsymbol{\pi}}_1, ..., \hat{\boldsymbol{\pi}}_j)\|^2_{\boldsymbol{\Lambda}} \forall \boldsymbol{\pi}_j, \hat{\boldsymbol{\pi}}_j \in \boldsymbol{\mathcal{P}}_{R_i} \tag{2}$$

The origin of a *wall* node can be computed as follows:

$$\boldsymbol{\kappa}_i = f_{R,\theta}(\boldsymbol{\pi}_1, ..., \boldsymbol{\pi}_j) \forall \boldsymbol{\pi}_j \in \boldsymbol{\mathcal{P}}_{W_i} \tag{3}$$

where κ_i is the origin of *wall* i, π_j is the definition of *plane* j, $\mathcal{P}_{W_i}$ is the set of *planes* connected to the *wall* R_i, and $f_{R,\theta}(\cdot)$ is the graph neural network F-GNN for *wall* origins.

The cost function $c_{\boldsymbol{\kappa}}$ for *wall* with origin $\boldsymbol{\kappa_i}$ and its set of planes *planes* $\mathcal{P}_{W_i}$ can be given as:

$$c_{\boldsymbol{\kappa}}(\boldsymbol{\kappa}_i, [\boldsymbol{\pi}_1, ..., \boldsymbol{\pi}_j]) = \\ \|\hat{\boldsymbol{\kappa}}_i - f_{W,\theta}(\hat{\boldsymbol{\pi}}_1, ..., \hat{\boldsymbol{\pi}}_j)\|^2_{\boldsymbol{\Lambda}} \forall \boldsymbol{\pi}_j, \hat{\boldsymbol{\pi}}_j \in \boldsymbol{\mathcal{P}}_{W_i} \tag{4}$$

IV. Training with Synthetic Dataset

To train G-GNN and the two F-GNNs, we employ a generator mimicking common building layouts in the form of graphs, called synthetic dataset generator. It contains all the required nodes, edges and their semantic and geometric definitions of all the layers of *S-Graphs+* [1] but the keyframes. To make it as realistic as possible, wall thickness, wall length, *plane* dropout, number of planes of a room amongst other variables can be tuned as summarized in Tab. I. Furthermore, we postprocess it with noise in orientation and position of all the geometric definitions. With it, a wide range of different but realistic buildings can be automatically generated. From this same source, different subgraphs are extracted and post-processed according to the inputs and ground truth required for the trainings.

Edge classification. As depicted in Fig. 4, to train G-GNN, one subgraph containing all the *planes* along with the *same room* and *same wall* edges is extracted from each full layout, forming the ground truth graph. To define the initial graph by proximity, all existing edges are removed and each node is then connected with the K nearest neighbour. The optimization process is guided by the cross-entropy criterion, with the loss minimized using the Adam optimizer.

Origin inference. As shown in Fig. 5, in each layout, one subgraph is formed for each higher-level node, it is *room* or *wall* nodes. The high-level node, its adjacent *plane* nodes and the edges connecting both are included in the graph. The origin definition of the high-level node is removed and used as the ground truth while the remaining graph is used as input to train the corresponding F-GNN. The optimization process is guided by the mean squared error (MSE) criterion, with the loss minimized using the Adam optimizer.

TABLE I: **Synthetic dataset parameters.** All the tuned parameters to generate the widest and more realistic range of building layouts in the training datasets.

Parameter	Value [range]	Unit
Number of voxels (x and y)	[25, 70]	
Voxels per room (x and y)	[10,60]	
Maximum size of building (x and y)	[60,100]	meter
Voxel size (x and y)	[0.1, 0.2]	meter
Number of building in dataset	2000	
Wall thickness	[0.05, 0.15]	meter
Plane dropout	10%	
L-shape room formation	40%	
# connected neighbours in proximity graph	10	
Noise - global rotation	[0,360]	degree
Noise - *plane* rotation	[0,5]	degree
Noise - *room* translation	[0,0.1]	meter
Noise - *room* rotation	[0,3]	degree

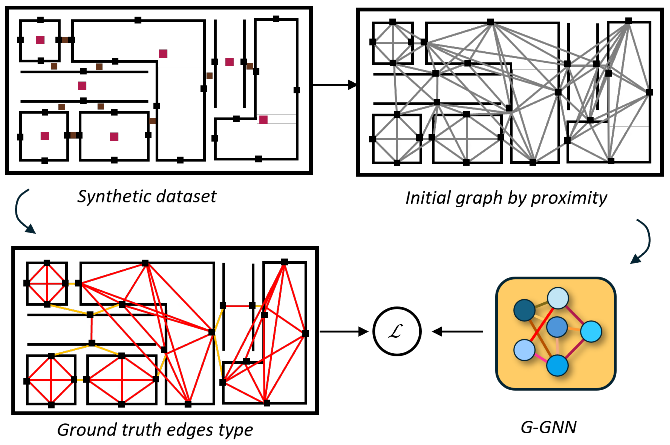

Fig. 4: **Edge classification training.** From the synthetic dataset (left, up), only *plane* nodes are extracted and linked by proximity (right, up) and fed to the G-GNN which infers the edge type (right, down). The loss is computed against the ground truth in the synthetic dataset (left, down. red and orange lines).

V. Experimental Results

A. Methodology

In this section, we evaluate and discuss the performance of the two neural architectures of our factor graph generation algorithm, G-GNN and the F-GNNs. Different metrics are

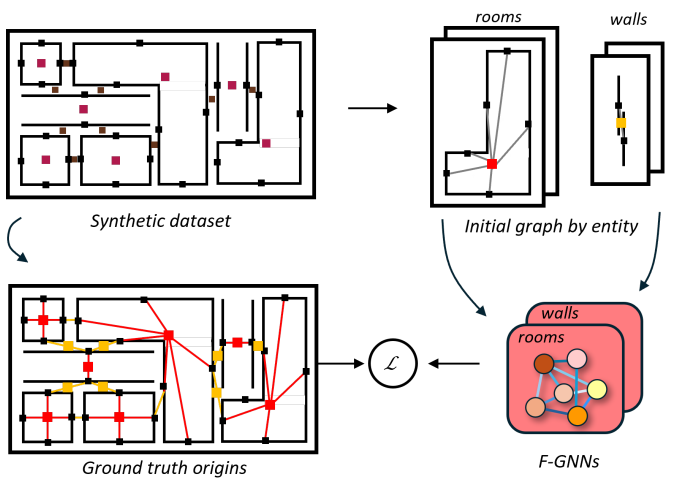

Fig. 5: **Origin inference training.** From the synthetic dataset (left, up), subgraphs containing *room* or *wall* and its adjacent *planes* are extracted (right, up) and fed independently to the corresponding F-GNN which infers the origin (right, down). The loss is computed against the ground truth origins in the synthetic dataset (left, down. red and orange squares).

compared over a set of synthetically generated buildings as it is the only ground truth available in the literature containing the required information. As the synthetic dataset (D) is divided into train, validation and test datasets, we use only the test dataset to provide the metrics. Furthermore, we run *S-Graphs+* [1] in simulated datasets (S) to test the edge classification of G-GNN and the posterior *room* and *wall* semantic node generation. Unfortunately, we are not aware of any real dataset containing *rooms* with more than four *planes* which can be suitable to be included in our validation.

For the edge-classification task of G-GNN, we evaluate the average precision, recall and Area Over Curve (AUC) averaged over the test building graphs to assess the graph expressiveness. In the case of the origin inference task of the F-GNNs, the Mean Squared Error (MSE) is the selected metric, expressed as an average over the buildings. For all of them, we include the inference time, assessing its validity in a real-time graph SLAM application. Unfortunately, the implementation of the node association module of *S-Graphs+* [1] is beyond the scope of this work so we cannot provide SLAM performance metrics such as ATE or MMA.

B. Results and Discussion

Semantic graph generation. The classification of the edges into *same room*, *same wall* or *none* by the G-GNN results in 82.4% precision, 86.3% recall and 90.3% AUC averaged over all the buildings in the test synthetic dataset. Three examples are shown in the first three rows of the left column of Fig. 6. We observe that most of *same room* edges (blue) are properly included although some of them are missing in L-shaped rooms and some of them are misclassified. Regarding the *same wall* edges (small red lines), almost all of them are included.

Using as input the results of the edge type classification, the results of the *same room* edge clustering algorithm are shown in the central column of Fig. 6. Most of the *planes* are properly clustered as the same room thanks to the longer cycle prioritisation (depicted with the same colors). However, there are examples in which very close *planes* of other rooms are misclassified.

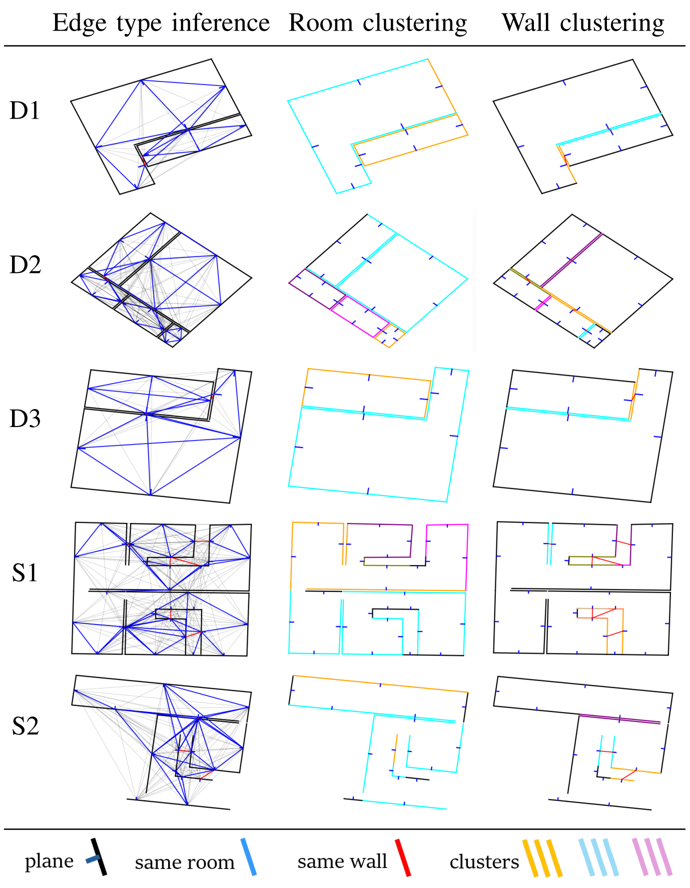

Fig. 6: **Edge classification by the G-GNN.** Top view. From the initial graph by proximity of *planes* and its geometrical definitions, the G-GNN classifies each edge into *same room* and *same wall* or *none* type (first column, include). Clustering algorithm for *rooms* and single pair association for *walls*. Tested in the synthetic dataset (D) and simulated dataset (S).

In the case of *same wall edges* (right column of Fig. 6), we can notice how all the pairs of *planes* which were connected by *same wall* inferred edges and now have the same color, are correct.

The test in simulated environments (S1 and S2) shows that, although *wall* generation is only slightly affected by difference in data distribution, *room* generation notably decreases the performance in around half of the inferred *rooms* due to *same room* edge missclassification. The average generation time of the semantic graph over the two simulated environments is $0.68s$, making it suitable for a real-time application in which the graph is constantly updated with new observations.

Origin inference. The MSE performance of the F-GNN which infers *room* origins is $0.89m$ and $0.17m$ for the F-GNN which infers *wall* origins. As depicted in the three examples of Fig. 7, the origins (blue dot) of *walls* (right column) is

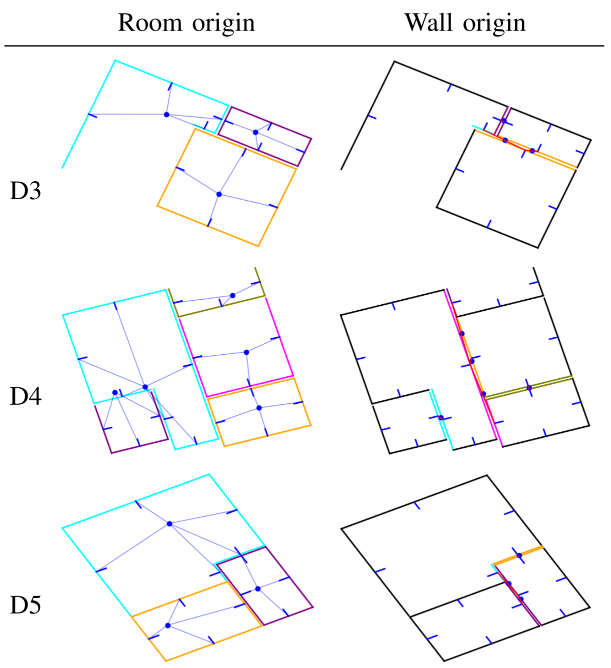

Fig. 7: **Origin inference by the F-GNNs.** Provided clusters of *planes* (shown with the same color) and its geometrical definitions, both F-GNNs infer the *room* and *wall* origins (blue dots) respectively. Tested in the synthetic dataset (D).

precisely placed between the *wall* origins. On its side, *room* origins are always placed near the geometric origin of the *planes* in most of the cases. The average inference time for the *room-plane* factor (the most computationally expensive) is $468\mu s$, which yields within the acceptable range for the graph optimization process.

VI. Conclusion

In this paper we present a framework for the generation of metric-semantic factor graphs. With it, we augment the factor graph of *S-Graphs+* [1] by receiving the *plane* layer and populating *room* and *wall* layers along with the interconnecting factors.

This novel approach is divided into three main steps. First, a semantic graph generation is accomplished by an edge type classification inferred by a GNN-based architecture called G-GNN. The inferred types are then clustered and a new *room* or *wall* node is created for each cluster. The second step is the inclusion of a geometric definition for each newly generated node. It is inferred by another GNN-based architecture, called F-GNN, different for each inferred node type. The third step is the inclusion of the factor on every newly created link *room-plane* and *wall-plane*.

The performance of the two neural networks and the full process has been trained and tested in a synthetic dataset that imitates building layouts. For the edge classification task (G-GNN), we obtained 82.4% precision, 86.3% recall and 90.3% AUC. For the origin inference (F-GNN), the MSE achieved is $0.89m$ for *rooms* and $0.17m$ for *walls*. The metric-semantic graph generation time is $0.68s$ on average and the *room-plane* factor average computation time is $468\mu s$, meaning that both of them remain on the average times of the previous version of *S-Graphs+* [1]. Furthermore, the tests in simulated environments show that while *wall* generation is only slightly affected by difference in data distribution, *room* generation is notably more impacted.

It remains as future work to implement the inclusion of the generated graphs into *S-Graphs+* [1], improve the generation performance and validate SLAM metrics in real scenarios, when we will be able to compare with the state of the art.

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] 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.

[3] 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.

[4] J. A. Millan-Romera, H. Bavle, M. Shaheer, M. R. Oswald, H. Voos, and J. L. Sanchez-Lopez, "Learning high-level semantic-relational concepts for slam," *arXiv preprint arXiv:2310.00401*, 2023.

[5] X. Chen, S. Chen, H. Zheng, J. Yao, K. Cui, Y. Zhang, and I. W. Tsang, "Node attribute generation on graphs," *arXiv preprint arXiv:1907.09708*, 2019.

[6] Z. Fang, Z. Zhang, G. Song, Y. Zhang, D. Li, J. Hao, and X. Wang, "Invariant factor graph neural networks," in *2022 IEEE International Conference on Data Mining (ICDM)*. IEEE, 2022, pp. 933–938.

[7] 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.

[8] 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.

[9] 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.

[10] 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.

[11] 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.

[12] 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.

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

[14] 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.

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

[16] W. Zhang, L. Zhang, D. Pfoser, and L. Zhao, "Disentangled dynamic graph deep generation. arxiv 2021," *arXiv preprint arXiv:2010.07276*, 2021.

[17] F. Dellaert, M. Kaess, *et al.*, "Factor graphs for robot perception," *Foundations and Trends® in Robotics*, vol. 6, no. 1-2, pp. 1–139, 2017.

[18] M. Haroon Dupty, Y. Dong, S. Leng, G. Fu, Y. L. Goh, W. Lu, and W. S. Lee, "Constrained layout generation with factor graphs," *arXiv e-prints*, pp. arXiv–2404, 2024.

[19] C. Taylor and J. Gross, "Factor graphs for navigation applications: A tutorial," *NAVIGATION: Journal of the Institute of Navigation*, vol. 71, no. 3, 2024.