# vS-Graphs: Tightly Coupling Visual SLAM and 3D Scene Graphs Exploiting Hierarchical Scene Understanding

Ali Tourani[1], Saad Ejaz[1], Hriday Bavle[1], Miguel Fernandez-Cortizas[1], David Morilla-Cabello[2], Jose Luis Sanchez-Lopez[1], and Holger Voos[1] [*†‡§]

November 13, 2025

**Abstract**

Current Visual Simultaneous Localization and Mapping (VSLAM) systems often struggle to create maps that are both semantically rich and easily interpretable. While incorporating semantic scene knowledge aids in building richer maps with contextual associations among mapped objects, representing them in structured formats, such as scene graphs, has not been widely addressed, resulting in complex map comprehension and limited scalability. This paper introduces **vS-Graphs**, a novel real-time VSLAM framework that integrates vision-based scene understanding with map reconstruction and comprehensible graph-based representation. The framework infers structural elements (i.e., rooms and floors) from detected building components (*i.e.,* walls and ground surfaces) and incorporates them into optimizable 3D scene graphs. This solution enhances the reconstructed map's semantic richness, comprehensibility, and localization accuracy. Extensive experiments on standard benchmarks and real-world datasets demonstrate that vS-Graphs achieves an average of **15.22**% accuracy gain across all tested datasets compared to state-of-the-art VSLAM methods. Furthermore, the proposed framework achieves environment-driven semantic entity detection accuracy comparable to that of precise LiDAR-based frameworks, using only visual features.

The code is publicly available at `https://github.com/snt-arg/visual_sgraphs` and is actively being improved. Moreover, a web page containing more media and evaluation outcomes is available on https://snt-arg.github.io/vsgraphs-results/.

## 1 Introduction

Robust environment understanding, a core foundation of robots' situational awareness [1] in the context of Simultaneous Localization and Mapping (SLAM), relies heavily on sensor quality and modality. While diverse sensors, *e.g.,* Light Detection And Ranging (LiDAR) and cameras, have been employed in SLAM, vision sensors offer a cost-effective solution for rich map reconstruction, forming a distinct category titled Visual SLAM (VSLAM) [2]. Among vision sensors, RGB-D cameras provide complementary visual and depth cues, overcoming monocular limitations by producing dense point clouds that capture fine spatial

[*1] Authors are with the Automation and Robotics Research Group, Interdisciplinary Centre for Security, Reliability, and Trust (SnT), University of Luxembourg, Luxembourg. Ali Tourani is also associated with the Institute for Advanced Studies (IAS), University of Luxembourg, Luxembourg. Holger Voos is also associated with the Faculty of Science, Technology, and Medicine, University of Luxembourg, Luxembourg. {`ali.tourani, saad.ejaz, hriday.bavle, miguel.fernandez, joseluis.sanchezlopez, holger.voos`}`@uni.lu`

[†2] Author is with the Instituto de Investigación en Ingeniería de Aragón (I3A), Universidad de Zaragoza, Spain. `davidmc@unizar.es`

[‡*] This research was funded, in whole or part, by the Luxembourg National Research Fund (FNR), DEUS Project, ref. C22/IS/17387634/DEUS. It was also partially funded by the Institute of Advanced Studies (IAS) of the University of Luxembourg through an "Audacity" grant (project TRANSCEND).

[§*] For the purpose of open access, and in fulfillment of the obligations arising from the grant agreement, the author has applied a Creative Commons Attribution 4.0 International (CC BY 4.0) license to any Author Accepted Manuscript version arising from this submission.

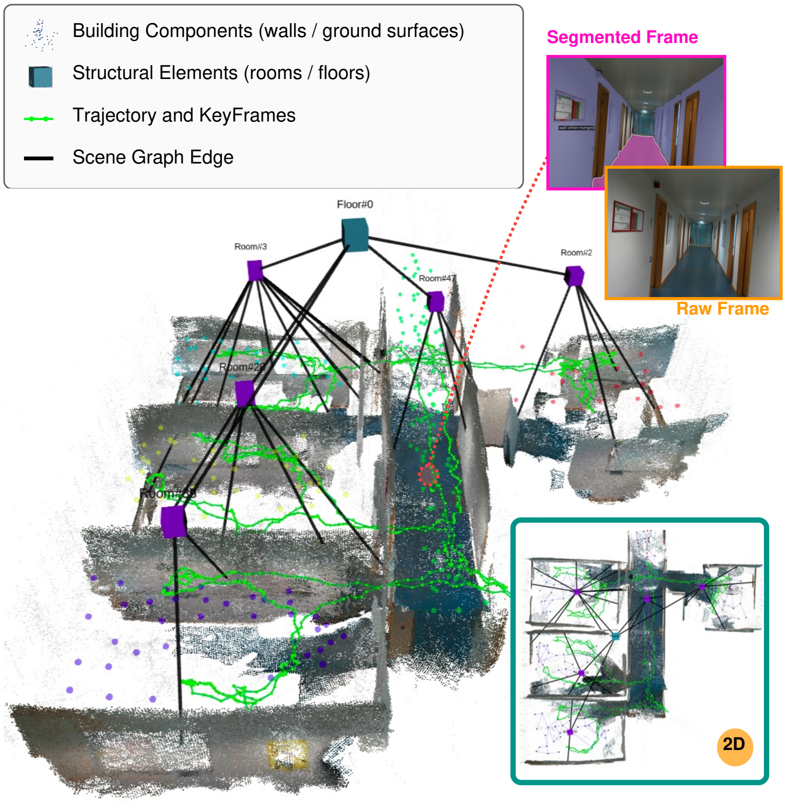


Figure 1: A reconstructed map tailored to the optimizable 3D scene graph generated by the proposed vS-Graphs, enriched with environment-driven semantic entities. Distinct color point clouds represent different building components (sequence *MR03* of the *AutoSense* dataset).

details, enabling precise detection, localization, and mapping of environmental elements [3]. To further enhance VSLAM performance, visual perception techniques are integrated, ranging from semantic scene understanding to the incorporation of artificial landmarks such as fiducial markers [4].

Beyond enriching maps with visual and depth information, various methodologies aim to organize this data into interpretable and structured representations. Among them, *scene graphs* represent environments as hierarchical structures that capture the presence of objects, their attributes, and inter-relationships. These solutions provide a higher level of abstraction for scene understanding by outlining spatial associations among observed entities [5]. While scene graph-driven works like [6, 7] focus on tailoring geometric and semantic cues for reliable interpretation, approaches such as Hydra [8] and HOV-SG [9] serve as 3D scene graph builders (rather than SLAM systems), representing spatial hierarchies without estimating camera trajectories. Moreover, HOV-SG requires ground-truth semantics and an entire-environment map to construct the scene graph offline. In contrast, works like *S-Graphs* [10] push the boundaries by directly incorporating scene graphs into the SLAM pipeline, relying on LiDAR odometry with planar surface extraction within a unified optimization system, though without support for visual input.

Inspired by LiDAR-based *S-Graphs* [10], this paper proposes a VSLAM framework titled **visual S-Graphs** (vS-Graphs), which seamlessly integrates scene graph generation within the SLAM process. vS-Graphs operates in real time and employs both visual and depth data to enhance map reconstruction and camera pose estimation. It reliably incorporates “building components” (*i.e.,* wall and ground surfaces), “structural elements” (*i.e.,* rooms and floors), and their spatial associations to produce a more structured and semantically coherent environmental representation. Unlike LiDAR-based S-Graphs, which derive semantic–relational structures from geometric data, vS-Graphs exploits the richer semantic and appearance information available in visual data to both validate and contextualize structural elements. This not only strengthens category verification but also establishes a scalable foundation for incorporating additional components in future extensions. The framework also generates interpretable 3D scene graphs with hierarchical optimization capabilities that pair robot poses from the underlying SLAM with detected entities, as depicted in Fig. 1. With this, the contributions of the paper are summarized below:

- A real-time multi-threaded VSLAM framework that constructs optimizable 3D scene graphs during map reconstruction,
- A vision-based method for recognizing and mapping building components (*i.e.,* wall and ground surfaces), enhancing map richness and trajectory estimation,
- A mechanism for extracting high-level structural elements from the localized building components for advancing scene understanding; and
- Publicly available source code to facilitate reproducibility and further research in the field.

# 2 Related Works

Recent advances in computer vision, combined with the development of reliable VSLAM frameworks such as ORB-SLAM 3.0 [11], have enabled more robust localization and mapping systems. As noted by [12], depth data information is crucial for enhancing scene understanding and supporting downstream tasks, such as environment modeling. In this regard, ElasticFusion [13] constructs globally consistent surfel maps for precise photometric tracking, but struggles with scalability and limited integration of environment-level cues. BAD SLAM [14] improves map and trajectory optimization via direct RGB-D Bundle Adjustment (BA), yet remains sensitive to initialization errors. More recent systems, like GS-SLAM [15], RTG-SLAM [16], and SplaTAM [17], incorporate 3D Gaussian scene representations to enhance tracking and map reconstruction. Nonetheless, they overlook the semantic context of the scene.

Building on these advances, RGB-D VSLAM frameworks have begun integrating semantic awareness into their mapping pipelines. Tools like Voxblox++ [18] enhance online scanning by incorporating volumetric, object-centric mapping, facilitating improved recognition of scene elements. NICE-SLAM [19] combines neural implicit and hierarchical representations with pre-trained geometric priors to improve dense scene reconstruction at the cost of heavy computation and increased localization errors. To address semantic awareness, works such as SaD-SLAM [20], OVD-SLAM [21], and YDD-SLAM [22] utilize Convolutional Neural Networks (CNNs) to filter feature points associated with specific semantic objects, thereby refining pose estimation and trajectory accuracy. Similarly, [23] introduces a binary CNN-based descriptor for robust feature matching while improving the initial pose measurements. Despite these efforts, most systems remain limited to object-level reasoning rather than constructing comprehensive scene representations enriched with environment-driven entities.

Another research direction explores fiducial markers as reliable landmarks, offering an alternative to purely vision-based scene interpretation. Approaches like [24,25] detect and map environment-driven entities labeled with markers that contribute to a more comprehensive understanding of the environment layout. Nevertheless, their reliance on pre-placed markers restricts their applicability to controlled environments, limiting flexibility in unprepared or dynamic settings.

In contrast, our approach introduces a real-time VSLAM framework that unifies scene graph construction with the VSLAM process. Relying on RGB-D data, it more effectively exploits visual and depth cues to enhance mapping accuracy and trajectory estimation, while generating semantically enriched representations of the environment without the need for external landmarks.

# 3 Proposed Method

## 3.1 System Overview

Building upon ORB-SLAM 3.0, vS-Graphs introduces substantial modifications to its baseline's core modules and adds new threads for robust scene analysis and reconstruction. The system architecture, shown in Fig. 2, details the individual threads, components, and their interconnections. The current version supports RGB-D input, utilizing depth data primarily to generate point clouds of surfaces, which are then validated

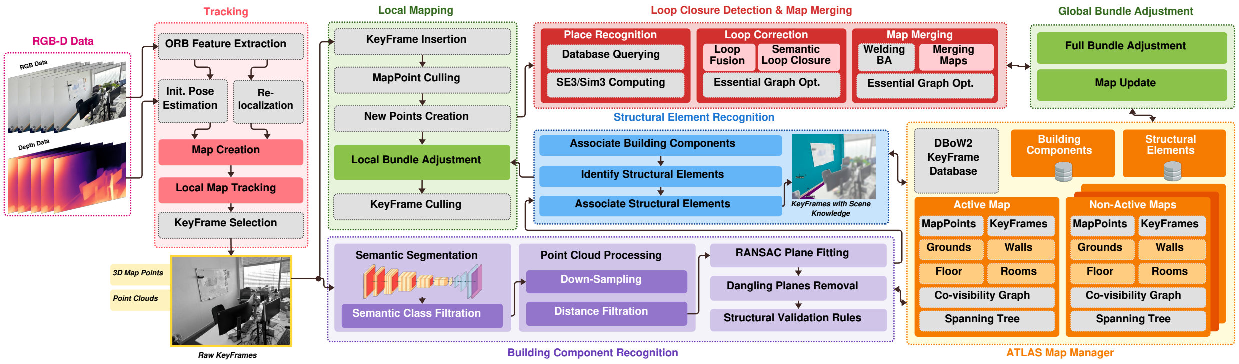


Figure 2: The multi-thread architecture of vS-Graphs. Modules with dashed borders and a light gray background are inherited directly from the baseline (*i.e.,* ORB-SLAM 3.0), while the remaining components are newly added or modified modules.

for robust scene understanding. The core contribution lies in seamlessly integrating two novel threads: *Building Component Recognition* (§3.2) and *Structural Element Recognition* (§3.3). These threads are deeply embedded within vS-Graphs, working tightly with other modules to enrich the reconstructed maps and ensure optimal performance.

In brief, RGB-D data is processed in real-time, supplying integrated visual and depth information for the subsequent modules. Visual features are extracted and tracked across frames in the *Tracking* thread, where pose information is either initialized or refined, depending on the map reconstruction stage, creating a 3D map with tracked features across frames. KeyFrame selection, a critical step following feature extraction, is performed within the *Tracking* thread by analyzing the visual data. These KeyFrames contain 3D map points and point clouds, forming the foundation for subsequent processes.

KeyFrames are then sent to the *Local Mapping* thread for map integration and optimization, with inaccurately posed KeyFrames culled for enhanced accuracy. Simultaneously, the *Building Component Recognition* thread identifies and localizes walls and ground surfaces by processing the KeyFrame-level visual-spatial data. These components are precisely detected by filtering the data through a CNN, followed by RANSAC plane fitting for global pose estimation. Concurrently, the *Structural Element Recognition* thread runs at fixed intervals, extracting higher-level entities, including rooms and floors, from the building components of the current (active) map. Eventually, the *local bundle adjustment* in the *Local Mapping* thread optimizes KeyFrame poses and map points for local consistency, as well as refining building components and structural elements, jointly optimizing geometric and semantic entities for accurate map reconstruction.

In the baseline, the *Atlas* map manager serves as the core of the multi-map management procedure, maintaining mapped instances and their associations in an active map, while also supporting operations such as map merging. In vS-Graphs, we extend this core to additionally store higher-level semantic entities, including building components and structural elements, enriching the map representation storage and retrieval beyond purely geometric information. Furthermore, the *Loop Closure Detection* checks for revisited locations and triggers *Global Bundle Adjustment* for optimization and map merging when a loop is detected. While the loop closure mechanism remains fundamentally geometric, the inclusion of environment-driven entities enables the system to verify and refine loop candidates. During loop closure and global optimization, redundant building components and structural elements detected across overlapping KeyFrames are automatically merged and their associations updated, ensuring a consistent map representation. Although the *semantic loop closure* module provides an additional consistency check that complements geometric matching, a fully semantic-driven loop closure triggered solely by the association of the detected building components and structural elements is an area for future investigation.

## 3.2 Building Component Recognition

This thread processes the point cloud and visual data from KeyFrames to extract fundamental environment-driven elements beneficial for scene understanding. The current version of vS-Graphs defines building components $\mathbf{\Psi}$ as walls and ground surfaces. Accordingly, each KeyFrame $K = \{\mathbf{P}, L, \varepsilon\}$ passes through the *building component recognition* module $B$, generating identified structural element set $\mathbf{\Pi}_{\mathbf{\Psi}}^{K} = B(\mathbf{K})$ for $K$. In the KeyFrame level, $\mathbf{P} = \{p_i \mid p \in \mathbb{R}^3, i \in \mathbb{N}\}$ is the point cloud, $L$ is the matrix of RGB data, and $\varepsilon$ represents additional metadata for mapping (such as camera pose).

Building component extraction is achieved by processing $L$ through a semantic segmentation function $S$, which labels each pixel in the RGB data, retaining only the relevant classes and discarding irrelevant entities. In contrast with the conventional object detection methods, panoptic scene segmentation provides *pixel-level class labels* and *instance differentiation*, enabling sharper boundary delineation and more precise object recognition. In principle, any reliable real-time panoptic segmentation framework can be integrated into vS-Graphs. However, the current version has integrated Panoptic-FCN (pFCN) [26] and YOSO [27] by default, due to their efficiency and strong balance between accuracy and real-time performance. In this regard, $\widehat{\mathbf{P}}_{\mathbf{\Psi}}^{K} = S(\mathbf{P}, L)$ refers to the segmented visual-spatial data of $K$, where $L_{(u,v)}$ refers to the semantic class label of a pixel at $(u, v) \in \mathbb{N}^2$. Applying semantic class filtration to filter building components' semantic classes $\mathbf{\Psi}$ (*i.e.,* wall and ground surfaces) takes place as follows:

$$\widehat{\mathbf{P}}_{\psi}^{K} = \{p_j \mid p_j \in \mathbb{R}^3, L_{p_j} \in \mathbf{\Psi}\} \tag{1}$$

$$\widehat{\mathbf{P}}_{\mathbf{\Psi}}^{K} = \{\widehat{\mathbf{P}}_{\psi}^{K}\} = \{\widehat{\mathbf{P}}_{wall}^{K}, \widehat{\mathbf{P}}_{ground}^{K}\} \tag{2}$$

where $\widehat{\mathbf{P}}_{\mathbf{\Psi}}^{K} \subset \mathbf{P}$ represents the set of semantically segmented point clouds, including the point clouds of $\widehat{\mathbf{P}}_{wall}^{K}$ and $\widehat{\mathbf{P}}_{ground}^{K}$. It should be noted that the classification confidence $\lambda_{(u,v)} \in \mathbb{R}$ for each pixel $L_{\Psi(u,v)}$ is set for potential classification errors.

The next stage is to optimize $\widehat{\mathbf{P}}_{\mathbf{\Psi}}^{K}$, as it may contain noisy or low-resolution points that negatively impact subsequent steps. In this regard, each segmented visual-spatial point cloud $\widehat{\mathbf{P}}_{\psi}^{K}$ undergoes a two-stage preprocessing procedure. The first step is applying down-sampling to fetch $\widehat{\mathbf{P}}_{\psi_d}^{K}$, a refined point cloud with reduced redundancy and noise, where $\widehat{\mathbf{P}}_{\psi_d}^{K} \subset \widehat{\mathbf{P}}_{\psi}^{K}$. As depth sensors exhibit increased noise at extreme ranges, the semantic classification of structural elements is most reliable within an optimal depth range. Thus, applying distance filtration based on the recommended sensor range results in retaining a subset of the point cloud as $\widehat{\mathbf{P}}_{\psi_\varsigma}^{K} \subset \widehat{\mathbf{P}}_{\psi_d}^{K}$. The final processed point cloud, referred to as $\widehat{\mathbf{P}}_{\mathbf{\Psi}_\varsigma}^{K} = \{\widehat{\mathbf{P}}_{\psi_\varsigma}^{K}\}$, is then forwarded to successive stages for further analysis.

Processing each $\widehat{\mathbf{P}}_{\psi_\varsigma}^{K}$ through RANdom SAmple Consensus (RANSAC) [28] plane fitting algorithm results in detecting semantically-validated building components with their geometric equations. Thus, sets of random points $\mathbf{p_r} \in \widehat{\mathbf{P}}_{\psi_\varsigma}^{K}$ are iteratively selected to calculate the normal vectors $\mathbf{n} \in \mathbb{R}^3$ representing validated planar components $\pi$ with the pre-defined distance $d(\mathbf{p_r}, \pi) \leq \epsilon$, where $\epsilon$ is the inlier threshold. Accordingly, the final output representing all defined building components is as follows:

$$\mathbf{\Pi}_{\psi}^{K} = \{\pi_{\psi_i}^{\ K} \mid \pi_{\psi_i}^{\ K} \in \mathbb{R}^3, i \in \mathbb{N}\} \tag{3}$$

$$\mathbf{\Pi}_{\mathbf{\Psi}}^{K} = \{\mathbf{\Pi}_{\psi}^{K}\} = \{\mathbf{\Pi}_{wall}^{K}, \mathbf{\Pi}_{ground}^{K}\} \tag{4}$$

The remaining reliable elements are checked against structural validation rules $\mathbf{\Pi}_{\mathbf{\Psi}}^{K} = V(\mathbf{\Pi}_{\mathbf{\Psi}}^{K})$, ensuring that they satisfy reasonable geometric constraints for the environment. For instance, walls should be represented as vertical planes, while ground surfaces should be defined as horizontal planes. The thread concludes the processing of KeyFrame $K$ by storing its detected building components $\mathbf{\Pi}_{\mathbf{\Psi}}^{K}$ within the current map in the *Atlas*, ensuring they contribute to the ongoing map reconstruction.

## 3.3 Structural Element Recognition

This thread repeatedly runs at constant time intervals (*i.e.,* every two seconds) to detect potential higher-level semantic entities that characterize the environment's layout, including rooms and floors. Structural elements comprise multiple building components, with their topological associations taken into account. The thread actively searches for layouts that form rooms, where a room is a spatially enclosed area bounded by at least two walls that surround a confined free-space cluster on the ground plane. Additionally, a floor represents a higher-order structural element that encompasses a collection of rooms within a single building level.

In the first step, the map's existing building components $\mathbf{\Pi_{\Psi}} = \{\mathbf{\Pi}_{\mathbf{\Psi}}^{K_i} \mid i \in \mathbb{N}\}$ are fetched from *Atlas*. Associating and merging redundant or conflicting building components is crucial to ensure consistency in the map reconstruction. In this context, the primary factors to assess are the spatial proximity and alignment of the mapped building components, requiring evaluating both the Euclidean distance and the angular difference between their normal vectors. Accordingly, the building component association condition is presented below:

$$\|\mathbf{\Pi}_{\mathbf{\Psi}}^{K_i} - \mathbf{\Pi}_{\mathbf{\Psi}}^{K_j}\| \leq \rho \quad \wedge \quad \cos^{-1}\left(\frac{\mathbf{n}_i \cdot \mathbf{n}_j}{\|\mathbf{n}_i\|\|\mathbf{n}_j\|}\right) \leq \eta \tag{5}$$

where $\mathbf{n}$ is the normal vector of a particular surface and $\rho$ and $\eta$ refer to spatial proximity and angular alignment thresholds, respectively. The associated building components $\mathbf{\Pi_{\Psi}}$ are subsequently utilized to detect structural elements $\mathbf{\Delta_{\Phi}}$, defined as below:

$$\mathbf{\Delta}_{\phi}^{K} = \{\delta_{\phi_m}^{\ K} \mid \delta_{\phi_m}^{\ K} \in \mathbb{R}^3, m \in \mathbb{N}\} \tag{6}$$

$$\mathbf{\Delta}_{\mathbf{\Phi}}^{K} = \{\mathbf{\Delta}_{\phi}^{K}\} = \{\mathbf{\Delta}_{room}^{K}, \mathbf{\Delta}_{floor}^{K}\} \tag{7}$$

where $\mathbf{\Delta}_{\mathbf{\Phi}}^{K} \in \mathbf{\Delta_{\Phi}}$ is the set of structural elements belong to semantic classes $\mathbf{\Phi}$ found in KeyFrame $K$. The procedure for detecting structural elements is outlined below.

**Rooms.** A room $\delta_r^K = \{\mathbf{\Pi}_{wall}, \pi_{ground}^{K_q}, \nu_r\}$ is a convex, spatially enclosed area bounded by at least two walls, each oriented toward a distinct free-space cluster $\mathbf{\Upsilon} = \{\upsilon_j \mid \upsilon_j \in \mathbb{R}^3, j \in \mathbb{N}\}$ in the global frame. The free-space cluster is calculated from the depth camera points using the Voxblox tool [29]. The associated set of walls is denoted as $\mathbf{\Pi}_{wall} = \{\pi_{wall}^{K_1}, \ldots, \pi_{wall}^{K_n}\}$, where $n \geq 2$. Each wall $\pi_{wall}^{K_i}$ is validated using *proximity* and *directionality* constraints, ensuring geometric consistency with its corresponding free-space. *Proximity* is calculated as:

$$\forall\, \pi_{wall}^{K_i} \in \mathbf{\Pi}_{wall},\ \exists\, \upsilon_j \in \mathbf{\Upsilon} :\ \left|\mathbf{n}_{wall}^{K_i \top} \cdot \upsilon_j + d_{wall}^{K_i}\right| \leq \omega \tag{8}$$

where $\mathbf{n}_{wall}^{K_i} \in \mathbb{R}^3$ denotes the wall's normal vector, $d_{wall}^{K_i}$ is the plane offset, and $\omega$ is a tunable threshold controlling the allowable distance between the wall and nearby free-space points. *Directionality* is computed as follows:

$$\forall \pi_{wall}^{K_i} \in \mathbf{\Pi}_{wall} :\ \mathbf{n}_{wall}^{K_i \top}\left(\nu_{\Upsilon} - \nu_{wall}^{K_i}\right) < 0 \tag{9}$$

where $\nu_{wall}^{K_i} \in \mathbb{R}^3$ is the wall centroid and $\nu_{\Upsilon} \in \mathbb{R}^3$ is the centroid of the associated free-space cluster $\Upsilon$. These formulations enable the detection of convex $n$-wall rooms with arbitrary wall orientations, beyond Manhattan layouts, by enforcing that adjacent walls face the enclosed cluster. The ground plane $\pi_{ground}^{K_q}$ is associated with a room if it is approximately orthogonal to all wall normals and spatially enclosed by them. Accordingly:

$$\left|\mathbf{n}_{ground}^{\top} \cdot \mathbf{n}_{wall}^{K_i}\right| \leq \sin(\vartheta) \quad \wedge \quad d_{\pi_{wall}^{K_i}} \leq \nu_g \leq d_{\pi_{wall}^{K_j}} \tag{10}$$

where $\vartheta$ defines the angular tolerance and $\nu_g$ refers to the ground centroid lying between the nearest and farthest walls.

The optimization incorporates a geometry-aware constraint mechanism that updates each room's centroid ($\nu r_i$) based on the centroids of its associated walls, while simultaneously identifying *parallel* and *perpendicular* wall pairs within the same room. Such patterns are naturally common in man-made environments, where walls are commonly constructed to align along dominant structural axes. These relationships introduce geometric constraints that minimize angular deviations, encouraging walls to align closer to 0° or 90°, respectively. In the absence of such relations, only the centroid consistency term remains active, maintaining spatial coherence without enforcing unnecessary structure. This flexible design enables the system to accommodate rooms of arbitrary or asymmetric geometry without relying on prior layout assumptions, unlike rectangular-shaped formulations commonly used in previous works [8,9,30]. In this context, if a room contains parallel walls facing each other toward $\nu_{\Upsilon}$, a parallelism constraint penalizes deviation through a cost given by the angle between their normals:

$$c^{\parallel}_{\delta_r^K\,(i,j)} = 1 - \left|\mathbf{n}_i^{\top} \cdot \mathbf{n}_j\right|, \quad \text{where } |\mathbf{n}_i| = |\mathbf{n}_j| = 1. \tag{11}$$

where $\mathbf{n}_i$ and $\mathbf{n}_j$ are the normalized values of normal vectors of the two wall planes. The cost reaches zero for perfectly parallel walls ($\mathbf{n}_i \parallel \mathbf{n}_j$) and increases proportionally with angular deviation, providing a numerically stable measure of parallelism. Similarly, perpendicularity is calculated as:

$$c^{\perp}_{\delta_r^K (i,j)} = \left| \mathbf{n}_i^\top \cdot \mathbf{n}_j \right|, \quad \text{where } |\mathbf{n}_i| = |\mathbf{n}_j| = 1. \tag{12}$$

Finally, a centroid consistency term $c_{\nu_r}$ keeps the room centroid $\nu_r$ aligned with the mean of its wall centroids. The overall cost function for room-level constraints is:

$$c_{\nu_r} = \| \hat{\nu_r} - \frac{1}{n} \sum_{j=1}^{n} \nu_{wall}^{K_j} \|^2 \tag{13}$$

$$c_{\delta_r^K} = \frac{1}{N_{\|}} \sum_{i,j=0}^{N_{\|}} c^{\|}_{\delta_r^K (i,j)} + \frac{1}{N_{\perp}} \sum_{i,j=0}^{N_{\perp}} c^{\perp}_{\delta_r^K (i,j)} + c_{\nu_{r_i}} \tag{14}$$

were $N_{\|}$ and $N_{\perp}$ denote the number of parallel and perpendicular walls. Hence, when no parallel or perpendicular relations are detected, only the Euclidean-norm–based centroid update is activated, preserving localization accuracy while enhancing the structural coherence of the 3D scene graph.

**Floors.** A floor $\delta_f^K = \{\mathbf{\Delta}_r, \nu_f, \pi_{floor}^{K_i}\}$ is defined by a set of rooms $\mathbf{\Delta}_r$, all sharing a common horizontal reference plane $\pi_{floor}^{K_i}$ and the floor centroid $\nu_f$. The floor plane is associated with the constituent rooms if it satisfies the co-planarity condition within a vertical tolerance. In practice, this validation step is ignored in the current vS-Graphs implementation due to its single-floor design constraint. Additionally, its spatial extent is bounded by the union of all constituent room boundaries. The floor centroid $\nu_f$ is computed as the weighted arithmetic mean of all constituent room centroids:

$$\nu_f = \frac{\sum_{j=1}^{N_r} \nu_{r_j}}{N_r} \tag{15}$$

where $N_r = |\mathbf{\Delta}_{room}|$. The cost function to optimize the floor's vertex node is calculated as follows:

$$c_{\delta_f^K} = \sum_{t=1}^{T} \left\| \hat{\nu}_f - h(\mathbf{\Delta}_r^K) \right\|^2_{\mathbf{\Lambda}_{\tilde{\delta}_{f,t}}} \tag{16}$$

where $h(\dots)$ is a hierarchical mapping function that associates the floor's structural components with its centroid.

### 3.4 Scene Graph Structure

Fig. 3 illustrates the 3D scene graph structure generated using vS-Graphs, along with its corresponding modules shown in Fig. 2. In contrast to the traditional SLAM reconstructed maps, the geometric replicas of vS-Graphs are augmented with hierarchical, rich semantic data, enabling meaningful interaction with the environment. The generated scene graph fills the contextual understanding gap and provides better scene interpretation, enabling complementary missions such as scalable map-building and improved decision-making.

## 4 Experimental Results

### 4.1 Evaluation Criteria

**Setup.** Evaluations were conducted on a system with an Intel® Core™ i9-11950H processor (2.60 GHz), a 4GB NVIDIA T600 Mobile GPU, and 32GB of RAM.

**Datasets.** We evaluated vS-Graphs on standard benchmarks and in-house datasets to assess its performance across diverse environments and scene complexities. The standard datasets include ICL [31] photorealistic synthetic scenes and real-world data from OpenLORIS [32], ScanNet [33], and TUM-RGBD [34]. The in-house dataset, *AutoSense*, was collected using a custom-built device that records RGB-D video and LiDAR

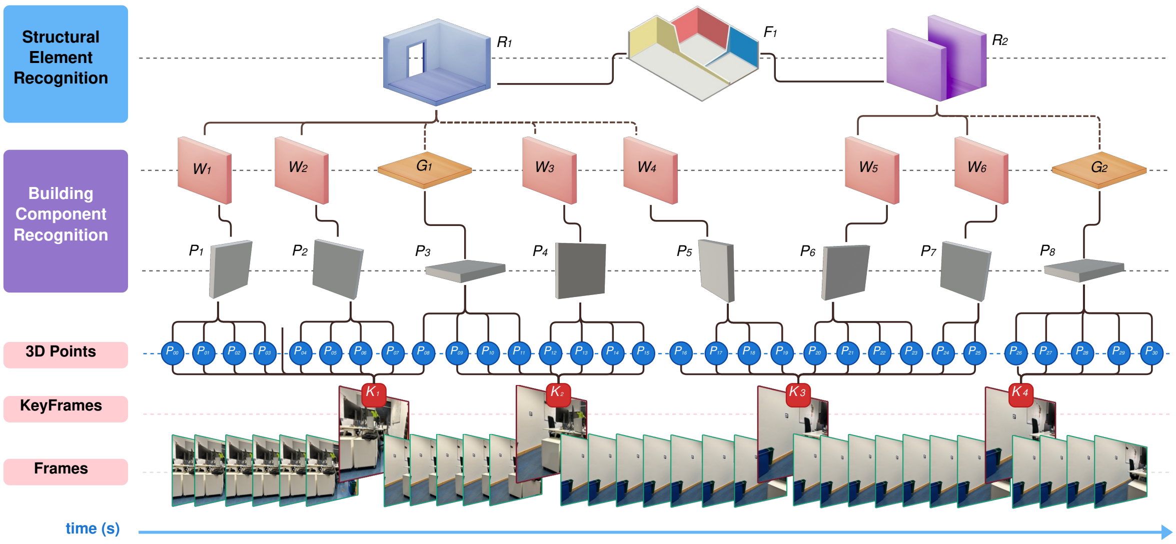


Figure 3: Scene graph structure generated using vS-Graphs, creating a hierarchical representation of the environment.

Table 1: Characteristics of the *AutoSense* in-house dataset. Each sequence includes one ground and one floor surface by design. *BC* denotes building components (*ground/walls*), and *SE* denotes structural elements (*rooms/floor*).

| **Seq.** | **#BC**<br>*(Ground+Walls)* | **#SE**<br>*(Room+Floor)* | **Description** |
|---|---|---|---|
| SR01 | 4 (1+3) | 2 (1+1) | partial view of a room |
| SR02 | 4 (1+3) | 2 (1+1) | room corner with intersecting walls |
| SR03 | 6 (1+5) | 2 (1+1) | a single rectangular room |
| MR01 | 14 (1+13) | 4 (3+1) | across rooms connected to a corridor |
| MR02 | 13 (1+12) | 4 (3+1) | adjoining rooms linked by a corridor |
| MR03 | 22 (1+21) | 6 (5+1) | a suite of interconnected rooms |

point clouds. It features diverse real-world indoor environments with varied architectural layouts and multi-room scenarios. The ground truth data in the dataset was derived from reliable LiDAR poses and point clouds generated by *S-Graphs*. Table 1 outlines the dataset characteristics, and Fig. 4 showcases the device and sample sequences. Due to space constraints, the complete evaluation results and figures are available at https://snt-arg.github.io/vsgraphs-results/.

## 4.2 Evaluations and Discussions

### 4.2.1 Trajectory Estimation Performance

To assess trajectory estimation accuracy, vS-Graphs was benchmarked against established and robust VSLAM systems, including ORB-SLAM 3.0 (baseline) [11], ElasticFusion [13], and BAD SLAM [14]. Marker-based methods (*e.g.,* [24]) were excluded due to their reliance on artificial landmarks and external pose constraints, which limit their applicability in marker-free environments. Similarly, neural field-based approaches (such as [19]) were omitted, as they depend on learned scene priors rather than explicit geometric–semantic representations. Additionally, vS-Graphs was evaluated with different segmentation backbones (pFCN and YOSO) to study the effect of segmentation quality and efficiency on trajectory robustness.

Table 2 presents the evaluation results, where each system is evaluated over eight runs per sequence, and performance is measured using Absolute Trajectory Error (ATE) reported in centimeters. Accordingly,

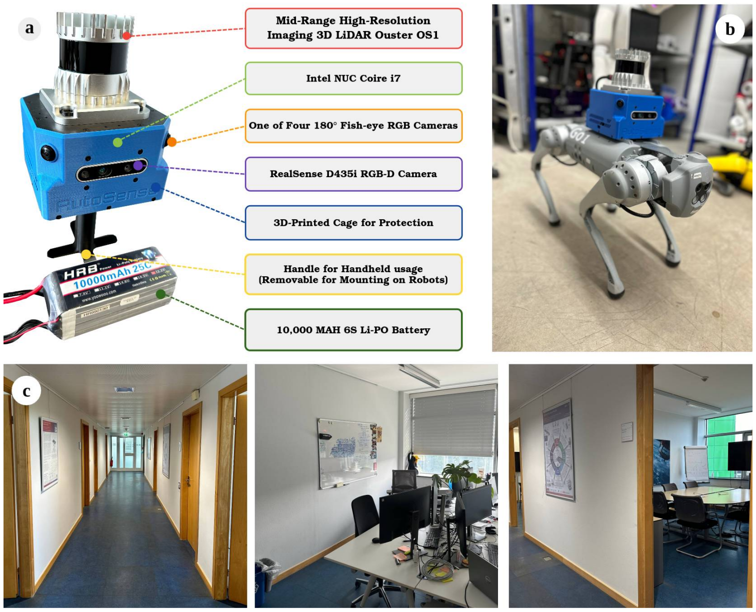

a
Mid-Range High-Resolution Imaging 3D LiDAR Ouster OS1
Intel NUC Coire i7
One of Four 180° Fish-eye RGB Cameras
RealSense D435i RGB-D Camera
3D-Printed Cage for Protection
Handle for Handheld usage (Removable for Mounting on Robots)
10,000 MAH 6S Li-PO Battery
b
c

Figure 4: In-house dataset collection using the *AutoSense* device: **a)** the setup overview, **b)** the device mounted on a legged robot, and **c)** some instances of the collected data.

vS-Graphs variants consistently achieve state-of-the-art performance, ranking first or second in nearly all evaluations across segmentation backbones and module configurations. This improvement stems from the additional geometric and semantic constraints imposed by accurately localized building components and structural elements. When combined with loop closures, these constraints yield even larger gains, *e.g.,* 75.38% for "*deer-w*" with the YOSO-based configuration. However, rapid motion and noisy depth data can negatively affect performance (*e.g.,* "*SR01*", BC-only). Another notable observation is that incorporating room entities further improves trajectory accuracy, particularly when room–wall constraints are enforced. This effect is most evident in looped ("*deer-w*") and multi-room environments ("*MR01*"), where maintaining geometric consistency is crucial, reducing ATE by 13.82% and 15.22% w.r.t. the baseline. It can be seen that the choice of panoptic backbone (YOSO or pFCN) has only a marginal effect on the trajectory accuracy of vS-Graphs, with higher segmentation quality naturally yielding more precise structural mapping.

It should be noted that a primary challenge shared by all tested VSLAMs arises in low-texture scenes (*e.g.,* corridors with uniformly painted walls), negatively impacting feature-matching and tracking procedures. In vS-Graphs, this can affect pose estimation and localization of building components and propagate the error through the structural element recognition stage. Additionally, treating building components as planar surfaces might limit the framework's applicability in environments with irregular geometries (*e.g.,* curved walls). Furthermore, overly or loosely permissive association thresholds can degrade building component identification performance, making careful threshold selection crucial.

### 4.2.2 Mapping Performance

Additionally, analyzing the accuracy of the reconstructed maps against *AutoSense*'s ground truth data revealed that *vS-Graphs* performs more robustly compared to ORB-SLAM 3.0 in terms of Root Mean Square Error (RMSE). As shown in Fig. 5, the median RMSE is consistently lower in *vS-Graphs*, indicating a higher level of overall mapping precision. *vS-Graphs* achieves superior mapping accuracy despite generating maps with $\sim$ 10.15% fewer points on average than the baseline, owing to its environment-driven constraints that enable a more coherent reconstruction.

Table 2: Absolute Trajectory Error (ATE) in centimeters (cm) for various VSLAM algorithms across eight iterations. The best and second-best results are **boldfaced** and underlined, respectively. vS-Graphs is evaluated with multiple semantic segmentation backbones (YOSO and pFCN) and with different levels of entity integration, including only building component cues (*vS-Graphs BC*) and the full pipeline with structural element recognition (*vS-Graphs*). The minimum completion rate of each sequence is set to 50%, and dashes indicate unavailable data due to tracking failures. Additionally, the "*%Gain*" rows quantify the percentage improvement achieved by vS-Graphs compared to its baseline (ORB-SLAM 3.0).

| **Dataset** | **Sequence** | **Time** | **Methodology** | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | **ORB-SLAM3** | **BAD SLAM** | **ElasticFusion** | **vS-Graphs** | | | | **vS-Graphs vs. ORB-SLAM3** | | | |
| | | | | | | *BC Only* | | *BC+SE* | | *BC Only* | | *BC+SE* | |
| | | | | | | YOSO | pFCN | YOSO | pFCN | YOSO | pFCN | YOSO | pFCN |
| **ICL** | deer-ground | 79.9 | **0.74** | 147.60 | 14.50 | 0.81 | 0.91 | 0.78 | 0.89 | -8.89 | -23.40 | -5.25 | -20.09 |
| | deer-walkh | 65.3 | 6.96 | – | 82.50 | 6.08 | 6.23 | **5.96** | 6.14 | 12.64 | 10.54 | 14.38 | 11.80 |
| | deer-walk | 64.0 | 9.91 | 147.40 | 62.00 | 8.82 | 12.19 | **2.44** | 4.02 | 10.97 | -22.98 | 75.38 | 59.43 |
| | deer-running | 28.4 | 6.92 | – | 78.70 | 5.34 | 5.47 | **2.83** | 3.34 | 22.82 | 20.98 | 59.12 | 51.75 |
| | deer-mav-fast | 102.5 | 2.67 | 4.60 | – | 2.65 | 2.07 | 2.23 | **1.26** | 0.96 | 22.57 | 16.61 | 52.88 |
| | **Total** | 340 | 5.44 | 99.87 | 59.43 | 4.74 | 5.37 | **2.85** | 3.13 | 7.70 | 1.54 | 32.05 | 31.15 |
| **OpenLORIS** | office1-1 | 27.0 | 6.37 | 12.30 | 6.88 | 6.41 | **6.33** | 6.42 | 6.37 | -0.66 | 0.50 | -0.85 | -0.06 |
| | office1-2 | 30.0 | 10.05 | 14.00 | 11.26 | 9.91 | 9.93 | **9.28** | 9.68 | 1.39 | 1.24 | 7.67 | 3.69 |
| | office1-3 | 12.0 | 14.73 | 19.50 | – | 14.57 | 16.05 | **14.44** | 15.01 | 1.08 | -9.00 | 1.94 | -1.93 |
| | office1-4 | 29.0 | 12.51 | 34.30 | 18.03 | 12.68 | 12.55 | 11.54 | **11.36** | -1.37 | -0.33 | 7.77 | 9.21 |
| | office1-5 | 53.0 | 11.57 | 30.20 | 22.15 | 11.11 | 11.29 | 10.71 | **10.59** | 4.03 | 2.48 | 7.45 | 8.49 |
| | office1-6 | 36.5 | 5.86 | 10.70 | – | 5.90 | 5.89 | 5.70 | **5.67** | -0.68 | -0.45 | 2.76 | 3.27 |
| | office1-7 | 38.6 | 17.17 | 20.30 | – | 17.55 | 17.50 | **13.00** | 13.25 | -2.17 | -1.92 | 24.30 | 22.84 |
| | **Total** | 226.1 | 11.18 | 20.19 | 14.58 | 11.16 | 11.36 | **10.16** | 10.28 | 0.23 | -1.07 | 7.29 | 6.50 |
| **ScanNet** | scn0041_01 | 75.0 | 14.26 | 22.20 | 21.96 | 14.29 | 14.06 | **13.56** | 13.78 | -0.27 | 1.37 | 4.88 | 3.34 |
| | scn0200_00 | 37.0 | 4.88 | – | – | 4.67 | 4.65 | 4.55 | **4.42** | 4.36 | 4.79 | 6.85 | 9.52 |
| | scn0614_01 | 36.2 | 13.55 | 13.90 | 20.23 | 12.50 | 13.38 | **12.36** | 12.45 | 7.75 | 1.26 | 8.76 | 8.10 |
| | scn0626_00 | 21.1 | 9.60 | 27.30 | 18.91 | 9.50 | 9.22 | 8.73 | **8.71** | 1.05 | 4.02 | 9.07 | 9.28 |
| | **Total** | 169.3 | 10.57 | 21.13 | 20.36 | 10.24 | 10.33 | **9.80** | 9.84 | 3.22 | 2.86 | 7.39 | 7.56 |
| **TUM-RGBD** | frb1-desk | 23.8 | 2.07 | 2.20 | 2.55 | 1.98 | **1.96** | 2.04 | 2.03 | 4.25 | 5.18 | 1.44 | 1.93 |
| | frb1-desk2 | 25.1 | 3.23 | 2.90 | 7.87 | 3.19 | 3.19 | 2.89 | **2.85** | 1.17 | 1.21 | 10.50 | 11.74 |
| | frb1-room | 49.1 | 13.25 | 20.90 | 16.75 | 13.02 | 13.29 | **5.13** | 6.77 | 1.75 | -0.26 | 61.29 | 48.91 |
| | frb2-desk-prs | 142.1 | 1.90 | 9.60 | 4.51 | 1.79 | **1.78** | 1.79 | 1.81 | 5.69 | 6.07 | 5.80 | 4.75 |
| | frb3-strct | 31.9 | 1.62 | 2.10 | 1.75 | 1.54 | 1.53 | **1.52** | 1.54 | 5.04 | 5.64 | 6.32 | 5.08 |
| | **Total** | 272 | 4.41 | 7.54 | 6.68 | 4.31 | 4.35 | **2.67** | 3.00 | 3.58 | 3.57 | 17.07 | 14.48 |
| **AutoSense** | SR01 | 61 | 7.98 | 13.00 | **6.62** | 8.10 | 8.16 | 7.34 | 7.63 | -1.58 | -2.28 | 7.99 | 4.36 |
| | SR02 | 80 | 8.75 | 13.50 | – | **8.21** | 8.29 | 8.23 | 8.26 | 6.15 | 5.32 | 5.95 | 5.60 |
| | SR03 | 170 | 10.90 | 55.70 | – | 10.70 | 10.71 | **10.51** | 11.04 | 1.85 | 1.71 | 3.56 | -1.30 |
| | MR01 | 305 | 15.26 | 41.30 | – | 15.44 | 14.98 | **12.23** | 12.75 | -1.13 | 1.85 | 19.87 | 16.47 |
| | MR02 | 210 | 23.97 | – | 53.48 | 21.68 | 23.78 | **20.13** | 21.19 | 9.58 | 0.83 | 16.03 | 11.61 |
| | MR03 | 783 | 22.19 | 81.21 | – | 22.00 | 19.23 | **17.70** | 17.84 | 0.87 | 13.35 | 20.24 | 19.61 |
| | **Total** | 1609 | 14.84 | 40.94 | 30.05 | 14.35 | 14.19 | **12.69** | 13.12 | 2.63 | 3.46 | 12.27 | 9.39 |
| **Overall Total** | | 43m 37s | 9.29 | 37.93 | 26.22 | 8.96 | 9.12 | **7.63** | 7.87 | 3.47 | 2.07 | 15.22 | 13.82 |

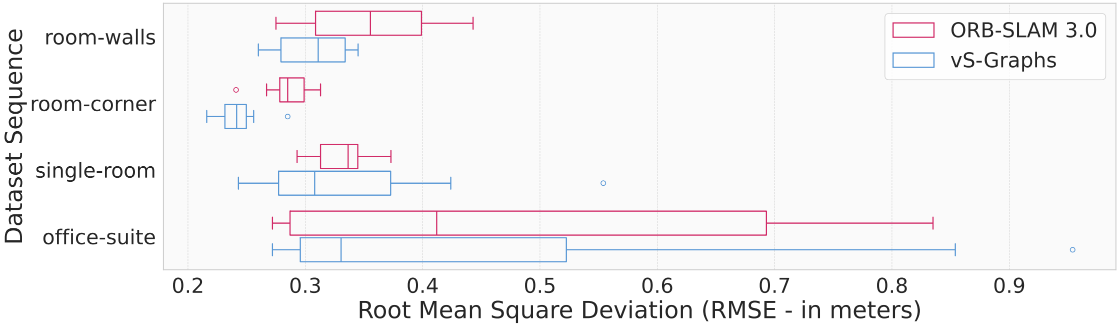


Figure 5: Mapping performance across eight iterations, showing *AutoSense* sequences with less than one meter RMSE.

#### 4.2.3 Scene Understanding Performance

This experiment evaluates vS-Graphs in terms of semantic scene understanding, with a particular focus on detecting entities essential for interpreting the environment's structural layout. For benchmarking, we selected the multi-room sequences from the *AutoSense* dataset, which include ground truth annotations derived from LiDAR scans. Table 3 reports a quantitative comparison of vS-Graphs against two state-of-the-art approaches: Hydra [8] and *S-Graphs* [10]. For presentation clarity, *precision* and *recall* values for structural elements were omitted from the table to avoid visual saturation. The ground truth counts of rooms and walls were obtained by manual inspection of the dataset, where each entity was visually verified and counted. While *S-Graphs* benefits from the geometric accuracy of LiDAR point clouds, Hydra was configured to use visual point clouds, ensuring a fair comparison against our purely vision-based approach.

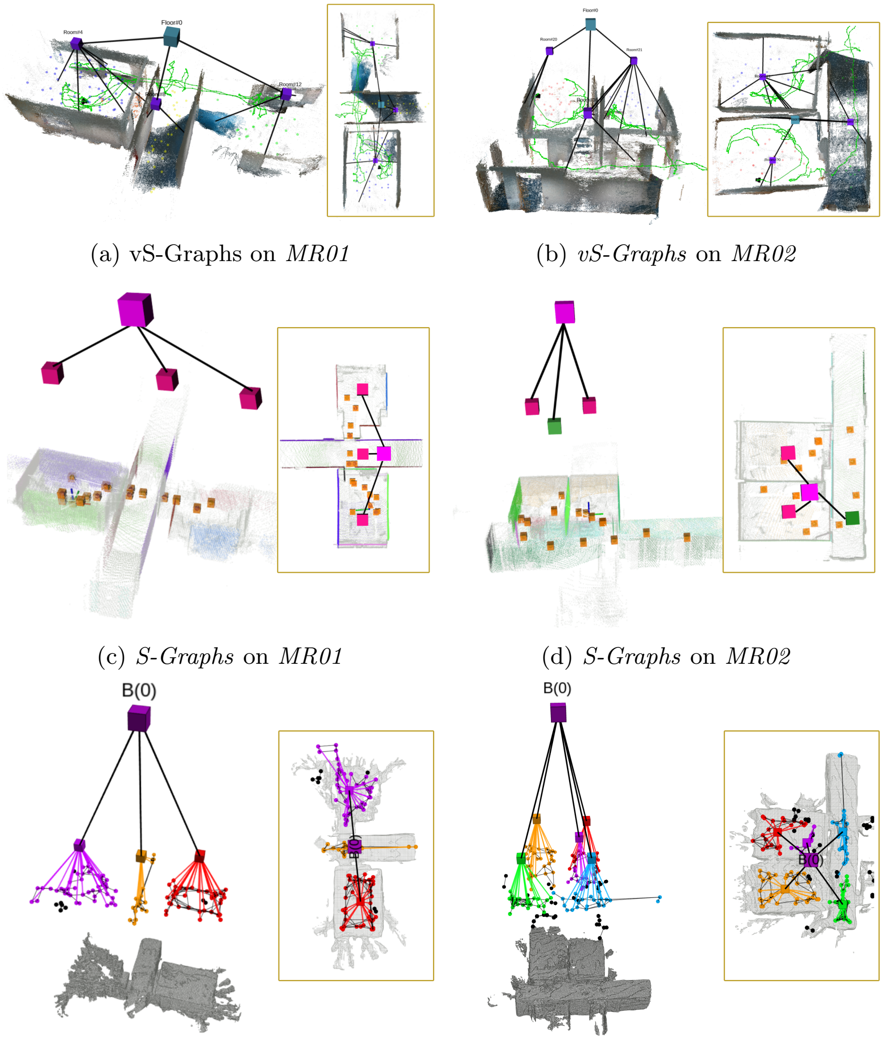
Experimental results show that vS-Graphs, despite operating solely on RGB-D input, achieves an accuracy comparable to the LiDAR-based method (*S-Graphs*) in detecting building components and structural elements. This highlights the effectiveness of our visual feature processing and scene graph generation in capturing structural layouts with high precision. Notably, vS-Graphs demonstrates higher *recall*, even in larger and more complex environments such as MR03, owing to its visual verification of building components that ensures consistent entity detection across extended scenes. It should be noted that "wall" entities are not directly provided in Hydra; therefore, Hydra's performance is assessed based on correct "room" element counting and recognition. Additionally, Fig. 6 provides a qualitative comparison of the reconstructed scene graphs generated by vS-Graphs, *S-Graphs*, and Hydra across two representative dataset sequences.

#### 4.2.4 Runtime Analysis

vS-Graphs achieves real-time performance with an average processing rate of $22 \pm 3$ FPS, exceeding the 20 FPS threshold considered for real-time operation. This is accomplished through a multi-threaded architecture, as shown in Fig. 7. The "*Tracking*" thread processes visual features at the frame level, while "*Local Mapping*" concurrently maps objects and optimizes their positions. "*Building Component Recognition*", running in parallel at the KeyFrame level, identifies potential wall and ground surfaces from the online panoptic segmentation. The "*Structural Element Recognition*" runs less frequently and in constant periods (every two seconds) to infer rooms and floors in the map. Compared to ORB-SLAM 3.0's $29 \pm 3$ FPS on the same hardware and dataset, the slightly reduced frame rate is a reasonable trade-off for vS-Graphs's rich semantic scene understanding.

Table 3: Scene understanding accuracy of vS-Graphs on multi-room sequences of *AutoSense*. *BC* and *SE* denote "building components" and "structural elements," respectively, while *GT* indicates the ground truth count of manually verified items.

| | | **Detected / GT** | | **Precision** | **Recall** |
|---|---|---|---|---|---|
| *Method* | *Sequence* | *BC* | *SE* | *BC* | *BC* |
| ***S-Graphs*** [10] | MR01 | 11 / 14 | 4 / 4 | 0.92 | 0.92 |
| | MR02 | 12 / 13 | 4 / 4 | 1.00 | 0.92 |
| | MR03 | 20 / 22 | 6 / 6 | 0.90 | 0.95 |
| **Hydra** [8] | MR01 | N/A | 4 / 4 | N/A | N/A |
| | MR02 | N/A | 5 / 4 | N/A | N/A |
| | MR03 | N/A | 6 / 6 | N/A | N/A |
| ***vS-Graphs*** (ours) | MR01 | 13 / 14 | 4 / 4 | 0.86 | 1.00 |
| | MR02 | 13 / 13 | 4 / 4 | 0.92 | 0.92 |
| | MR03 | 23 / 22 | 6 / 6 | 0.96 | 1.00 |



(a) vS-Graphs on *MR01*

(b) *vS-Graphs* on *MR02*

(c) *S-Graphs* on *MR01*

(d) *S-Graphs* on *MR02*

(e) Hydra on *MR01*

(f) Hydra on *MR02*

Figure 6: Qualitative comparison of reconstructed scene graphs on multi-room sequences of the *AutoSense* dataset.

# 5 Conclusions

This paper introduced vS-Graphs, a real-time VSLAM framework that reconstructs the robot's operating environment using optimizable hierarchical 3D scene graphs. It detects building components (wall and ground surfaces), from which structural elements (rooms and floors) are inferred, and incorporates them into hierarchical representations. Consequently, beyond enhancing map reconstruction by integrating these entities, vS-Graphs offers structured representations of spatial relationships among high-level environment-driven semantic entities. Experimental results on standard and in-house indoor datasets demonstrated that the framework achieves superior trajectory estimation and mapping performance compared to state-of-the-art VSLAMs, reducing trajectory error by 15.22% across a wide range of dataset instances. Other evaluations have shown that the visual features processed by vS-Graphs can identify semantic entities describing the environment's layout with accuracy comparable to that of precise LiDAR-based methods.

Future work includes integrating additional building components (*e.g.,* ceilings, doorways), extending support for complex and concave room layouts via GNN-based approaches, and developing fully semantic-driven loop closure detection, where revisited areas can be identified primarily based on previously detected structural elements and building components.

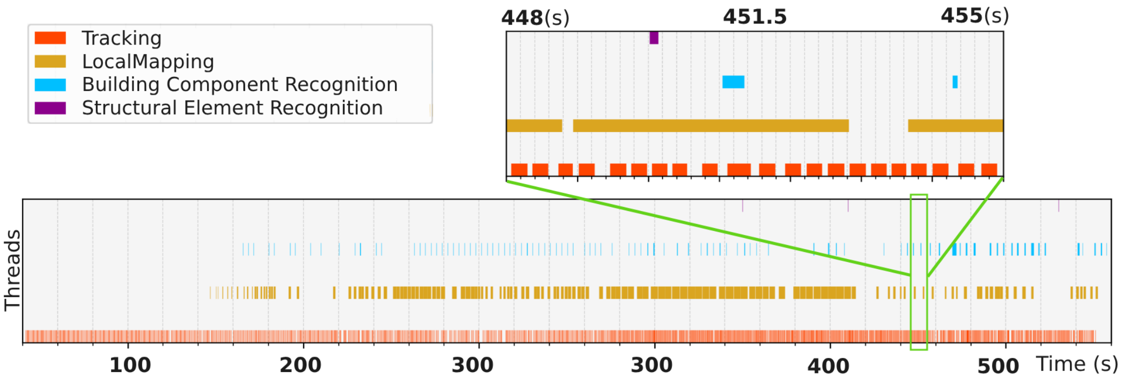


Figure 7: Timeline of thread execution within *vS-Graphs* while processing a sample dataset instance (sequence *room-walls*).

## Appendix I. Semantic Augmentation using Fiducial Markers

In vS-Graphs, fiducial markers serve as an **optional semantic augmentation mechanism**, providing high-level contextual information to the reconstructed map. Unlike marker-based Visual SLAM methods [4, 24,25,35], where markers play a central role in localization and map construction, their role in our vS-Graphs system is supportive rather than structural. The pipeline does not depend on the presence of markers for map accuracy or robustness; instead, they enrich the representation by linking structural elements to semantic labels. To enable this, we assume that each structural element (*e.g.,* a room) may contain one unique **ArUco marker**. This criterion is performed during the data collection procedures, where unique markers are placed in each free space zone, such as rooms or corridors, and their association is stored in a dedicated environment-driven database (`JSON` file). For instance, a marker with marker_id = 5 is assigned to "*corridor-A*", while marker marker_id = 8 was assigned to "*office-B*". At runtime, if (i) **the ArUco detector is active** and (ii) **the corresponding association file (database of marker information) is provided**, the system uses this information to enrich detected structural elements semantically. Thus, if the pose of a detected marker falls within the area of a recently detected n-wall room, it will be augmented with the semantic label stored in the database.

Fig. 8 shows the overall architecture of vS-Graphs with the fiducial marker detection and integration module, with recognized markers being stored in the *ATLAS Map Manager*. In brief, RGB-D data is processed in real-time, supplying integrated visual and depth information for the subsequent modules. Meanwhile, "*Fiducial Marker Detection*" (utilizing the ArUco library in this work) operates independently on input frames,

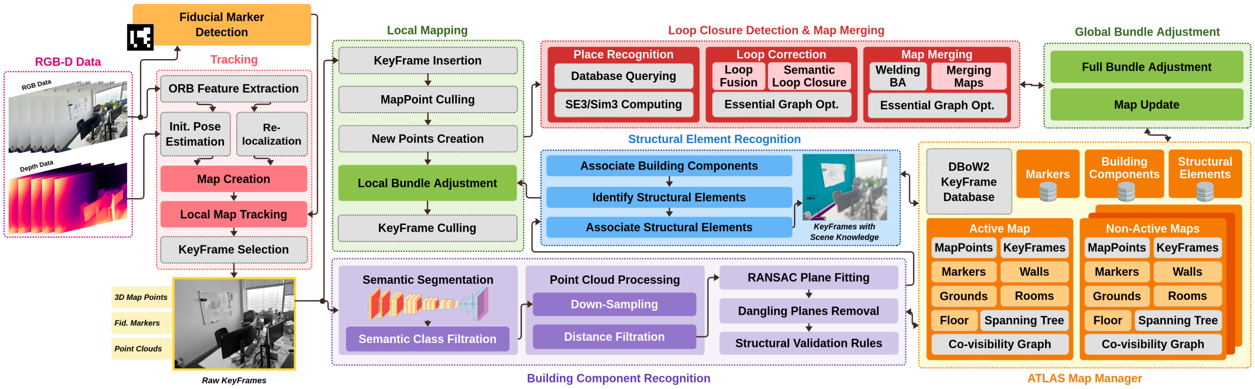


Figure 8: The multi-thread architecture of vS-Graphs with the optional fiducial marker detector module. Modules with dashed borders and a light gray background are inherited directly from the baseline (*i.e.,* ORB-SLAM 3.0), while the remaining components are the contributions of vS-Graphs.

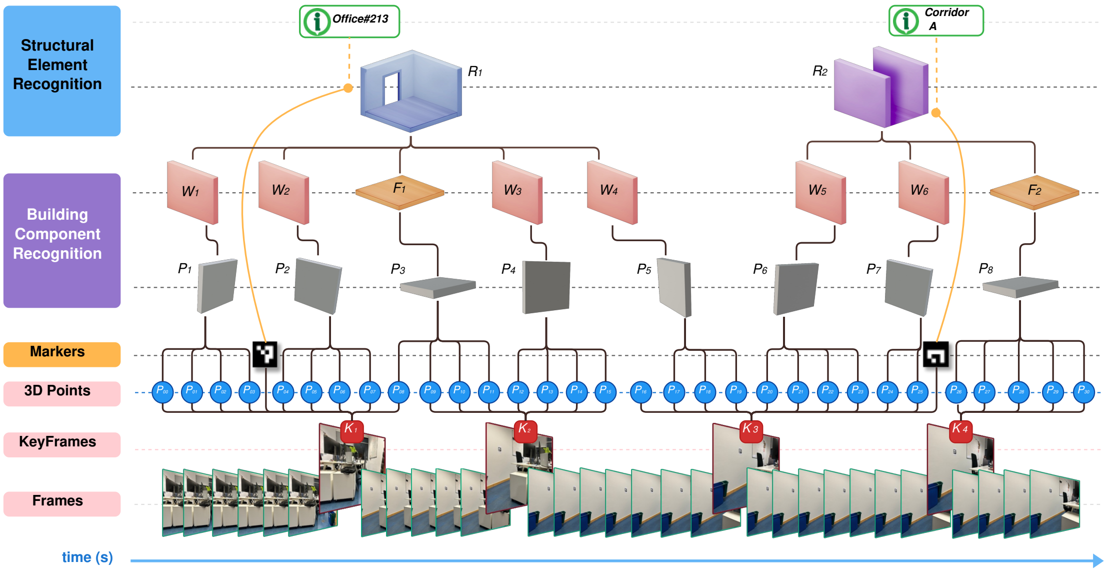

Structural Element Recognition
Building Component Recognition
Markers
3D Points
KeyFrames
Frames
time (s)
Office#213
Corridor A
$R_1$
$R_2$
$W_1$
$W_2$
$F_1$
$W_3$
$W_4$
$W_5$
$W_6$
$F_2$
$P_1$
$P_2$
$P_3$
$P_4$
$P_5$
$P_6$
$P_7$
$P_8$
$P_{00}$ $P_{01}$ $P_{02}$ $P_{03}$ $P_{04}$ $P_{05}$ $P_{06}$ $P_{07}$ $P_{08}$ $P_{09}$ $P_{10}$ $P_{11}$ $P_{12}$ $P_{13}$ $P_{14}$ $P_{15}$ $P_{16}$ $P_{17}$ $P_{18}$ $P_{19}$ $P_{20}$ $P_{21}$ $P_{22}$ $P_{23}$ $P_{24}$ $P_{25}$ $P_{26}$ $P_{27}$ $P_{28}$ $P_{29}$ $P_{30}$
$K_1$
$K_2$
$K_3$
$K_4$

Figure 9: Scene graph structure generated using vS-Graphs, creating a hierarchical representation of the environment.

identifying markers *only if they are present* in the environment and associating them with the corresponding frames. Visual features are extracted and tracked across sequential frames in the "*Tracking*" thread, where pose information is either initialized or refined, depending on the map reconstruction stage, creating a 3D map with tracked features across frames. KeyFrame selection, a critical step following feature extraction, is performed within the "*Tracking*" thread by analyzing the visual data. These KeyFrames contain 3D map points, point clouds, and potentially detected fiducial markers, forming the foundation for subsequent processes.

Accordingly, each fiducial marker $\mathbf{m}_i^K = \{\mathbf{p}_m, \nu_m\}$ in $G_t$ is constrained by the KeyFrame $K$ observing it, where $\mathbf{p}_m \in SE(3)$ is the marker's pose and $\nu_m$ is its center point. The mentioned constraint is defined below:

$$\sigma_m({}^{G}\mathbf{K}, \mathbf{m}_i^K) = \| {}^{L}\mathbf{m}_i \boxplus {}^{G}\mathbf{K} \boxminus {}^{G}\mathbf{m}_{i_1} \|^2_{\mathbf{\Lambda}_{\tilde{\mathbf{m}}_i}} \tag{17}$$

where ${}^{L}\mathbf{m}_i$ refers to the locally observed marker's pose, $\boxplus$ and $\boxminus$ represent the composition and inverse composition, $\| \dots \|$ is the *Mahalanobis* distance, and $\mathbf{\Lambda}_{\tilde{\mathbf{m}}_i}$ is marker's information matrix.

The marker $\mathbf{m}_i^{K_j} \in \mathbf{M}$ is associated with a structural element if it lies within the spatial bounds of the detected room/corridor $\delta^{K_g} \in \mathbf{\Delta}_\phi$. Thus, the Euclidean distance between the center point $\nu_m$ of $\mathbf{m}_i^{K_j}$ and the room/corridor's centroid $\nu_s$, expressed as $d(\nu_m, \nu_s) \leq \epsilon_s$, where $\epsilon_s$ is the proximity threshold. The supporting condition is that $\nu_m$ must be enclosed among all bounding walls $\mathbf{\Pi}_{wall}$ with the normal vector $\mathbf{n}$:

$$\forall \pi_{wall}^{K_i} \in \mathbf{\Pi}_{wall}, \quad (\mathbf{n}_{wall}^{K_i} \cdot (\nu_m - \nu_r)) \leq 0 \tag{18}$$

With this, Fig. 9 illustrates the extended scene graph structure of the vS-Graphs framework, emphasizing the role of fiducial markers in labeling the detected structural elements.

## Appendix II. Mapping Performance Relative to Baseline

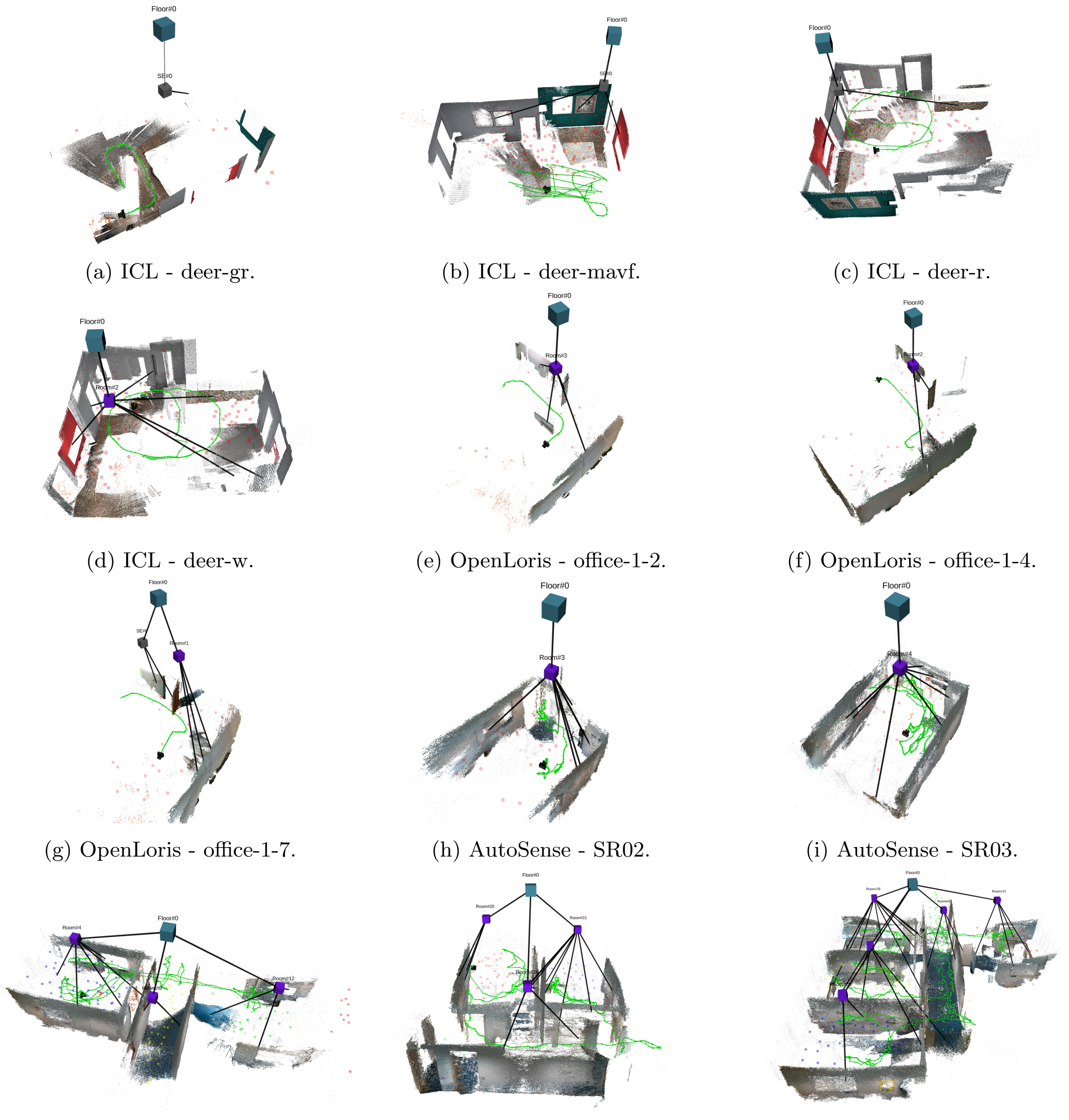
Table 4: Root Mean Square Error (RMSE) values for ORB-SLAM 3.0 and vS-Graphs across different sequences of the AutoSense dataset (over eight iterations). The results indicate that vS-Graphs generally achieves lower RMSE values.

| **Method** | **Sequence** | **Iteration Index** | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| vS-Graphs | SR01 | 0.2598 | 0.3102 | 0.2608 | 0.3454 | 0.3409 | 0.3122 | 0.2891 | 0.3246 |
| | SR02 | 0.2333 | 0.2165 | 0.2301 | 0.2494 | 0.2499 | 0.2414 | 0.2564 | 0.2853 |
| | SR03 | 0.2749 | 0.5544 | 0.2426 | 0.3509 | 0.2837 | 0.3801 | 0.2725 | 0.2920 |
| | MR01 | 4.3774 | 5.5194 | 6.2763 | 4.7143 | 6.8155 | 4.4640 | 4.8322 | 4.5823 |
| | MR02 | 1.1153 | 1.1836 | 0.9430 | 0.9017 | 0.9210 | 1.0799 | 0.9575 | 0.7648 |
| | MR03 | 1.2933 | 0.2978 | 0.3472 | 0.3137 | 1.1864 | 0.2894 | 0.4122 | 0.2721 |
| ORB-SLAM 3.0 | SR01 | 0.2772 | 0.4435 | 0.3509 | 0.3964 | 0.2753 | 0.3006 | 0.3995 | 0.3602 |
| | SR02 | 0.3111 | 0.2781 | 0.2862 | 0.2668 | 0.3000 | 0.2955 | 0.2408 | 0.2787 |
| | SR03 | 0.3451 | 0.3380 | 0.3279 | 0.2931 | 0.3054 | 0.3728 | 0.3076 | 0.3435 |
| | MR01 | 5.1266 | 5.0866 | 5.6484 | 5.6531 | 4.7557 | 6.2066 | 5.2847 | 5.2242 |
| | MR02 | 1.1828 | 1.1521 | 0.9291 | 0.9648 | 0.9379 | 1.0269 | 1.2719 | 1.0016 |
| | MR03 | 0.2869 | 2.5426 | 1.5765 | 2.3271 | 2.0381 | 0.2725 | 0.6926 | 0.4120 |

Table 5: Number of map points generated by ORB-SLAM 3.0 and vS-Graphs on the AutoSense dataset (over eight iterations). The measurements show that vS-Graphs produces fewer points than ORB-SLAM 3.0, while positively impacting the mapping accuracy.

| **Method** | **Sequence** | **Iteration Index** | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| vS-Graphs | SR01 | 6759 | 6704 | 6744 | 6741 | 6875 | 6804 | 6780 | 6842 |
| | SR02 | 7304 | 7120 | 7420 | 7457 | 7421 | 7212 | 7382 | 7160 |
| | SR03 | 11764 | 11591 | 11937 | 12054 | 12156 | 11844 | 11471 | 11475 |
| | MR01 | 20607 | 20682 | 19692 | 20714 | 21042 | 20274 | 19433 | 20546 |
| | MR02 | 16838 | 16622 | 16479 | 16563 | 17065 | 17200 | 16905 | 17274 |
| | MR03 | 48364 | 46237 | 48250 | 47850 | 47149 | 49008 | 48442 | 45224 |
| ORB-SLAM 3.0 | SR01 | 6931 | 7085 | 7057 | 6903 | 7012 | 6894 | 7063 | 7130 |
| | SR02 | 7639 | 7378 | 7548 | 7712 | 7540 | 7590 | 7363 | 7377 |
| | SR03 | 13140 | 12631 | 12818 | 12903 | 13021 | 13187 | 12613 | 12990 |
| | MR01 | 22564 | 22685 | 21916 | 22688 | 22552 | 22759 | 22701 | 22506 |
| | MR02 | 18816 | 18196 | 17899 | 18547 | 17722 | 18492 | 18117 | 17643 |
| | MR03 | 54797 | 55342 | 56531 | 55499 | 56056 | 55853 | 56187 | 54545 |

## Appendix III. Generated Scene Graphs (Qualitative Analysis)



(a) ICL - deer-gr.

(b) ICL - deer-mavf.

(c) ICL - deer-r.

(d) ICL - deer-w.

(e) OpenLoris - office-1-2.

(f) OpenLoris - office-1-4.

(g) OpenLoris - office-1-7.

(h) AutoSense - SR02.

(i) AutoSense - SR03.

(j) AutoSense - MR01.

(k) AutoSense - MR02.

(l) AutoSense - MR03.

Figure 10: Scene graph examples generated by vS-Graphs across different environments, demonstrating consistent hierarchical semantic representation of structural components.

# 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, p. 4849, 2023.

[2] H. Pu, J. Luo, G. Wang, T. Huang, and H. Liu, "Visual slam integration with semantic segmentation and deep learning: A review," IEEE Sensors Journal, 2023.

[3] D. Cai, R. Li, Z. Hu, J. Lu, S. Li, and Y. Zhao, "A comprehensive overview of core modules in visual slam framework," Neurocomputing, p. 127760, 2024.

[4] R. Muñoz-Salinas and R. Medina-Carnicer, "Ucoslam: Simultaneous localization and mapping by fusion of keypoints and squared planar markers," Pattern Recognition, vol. 101, p. 107193, 2020.

[5] A. Rosinol, A. Gupta, M. Abate, J. Shi, and L. Carlone, "3d dynamic scene graphs: Actionable spatial perception with places, objects, and humans," 2020.

[6] S. Koch, P. Hermosilla, N. Vaskevicius, M. Colosi, and T. Ropinski, "Sgrec3d: Self-supervised 3d scene graph learning via object-level scene reconstruction," in Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision, 2024, pp. 3404–3414.

[7] S.-C. Wu, J. Wald, K. Tateno, N. Navab, and F. Tombari, "Scenegraphfusion: Incremental 3d scene graph prediction from rgb-d sequences," in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2021, pp. 7515–7525.

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

[9] A. Werby, C. Huang, M. Büchner, A. Valada, and W. Burgard, "Hierarchical open-vocabulary 3d scene graphs for language-grounded robot navigation," in First Workshop on Vision-Language Models for Navigation and Manipulation at ICRA 2024, 2024.

[10] H. Bavle, J. L. Sanchez-Lopez, M. Shaheer, J. Civera, and H. Voos, "S-graphs+: Real-time localization and mapping leveraging hierarchical representations," IEEE Robotics and Automation Letters, vol. 8, no. 8, pp. 4927–4934, 2023.

[11] C. Campos, R. Elvira, J. J. G. Rodríguez, J. M. Montiel, and J. D. Tardós, "Orb-slam3: An accurate open-source library for visual, visual-inertial, and multimap slam," IEEE Transactions on Robotics, vol. 37, no. 6, pp. 1874–1890, 2021.

[12] A. Tourani, H. Bavle, J. L. Sanchez-Lopez, and H. Voos, "Visual slam: What are the current trends and what to expect?" Sensors, vol. 22, no. 23, p. 9297, 2022.

[13] T. Whelan, S. Leutenegger, R. F. Salas-Moreno, B. Glocker, and A. J. Davison, "Elasticfusion: Dense slam without a pose graph." in Robotics: science and systems, vol. 11. Rome, Italy, 2015, p. 3.

[14] T. Schops, T. Sattler, and M. Pollefeys, "Bad slam: Bundle adjusted direct rgb-d slam," in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2019, pp. 134–144.

[15] C. Yan, D. Qu, D. Xu, B. Zhao, Z. Wang, D. Wang, and X. Li, "Gs-slam: Dense visual slam with 3d gaussian splatting," in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2024, pp. 19 595–19 604.

[16] Z. Peng, T. Shao, Y. Liu, J. Zhou, Y. Yang, J. Wang, and K. Zhou, "Rtg-slam: Real-time 3d reconstruction at scale using gaussian splatting," in ACM SIGGRAPH 2024 Conference Papers, 2024, pp. 1–11.

[17] N. Keetha, J. Karhade, K. M. Jatavallabhula, G. Yang, S. Scherer, D. Ramanan, and J. Luiten, "Splatam: Splat track & map 3d gaussians for dense rgb-d slam," in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2024, pp. 21 357–21 366.

[18] M. Grinvald, F. Furrer, T. Novkovic, J. J. Chung, C. Cadena, R. Siegwart, and J. Nieto, "Volumetric instance-aware semantic mapping and 3d object discovery," IEEE Robotics and Automation Letters, vol. 4, no. 3, pp. 3037–3044, 2019.

[19] Z. Zhu, S. Peng, V. Larsson, W. Xu, H. Bao, Z. Cui, M. R. Oswald, and M. Pollefeys, "Nice-slam: Neural implicit scalable encoding for slam," in Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, 2022, pp. 12 786–12 796.

[20] X. Yuan and S. Chen, "Sad-slam: A visual slam based on semantic and depth information," in 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE, 2020, pp. 4930–4935.

[21] J. He, M. Li, Y. Wang, and H. Wang, "Ovd-slam: An online visual slam for dynamic environments," IEEE Sensors Journal, vol. 23, no. 12, pp. 13 210–13 219, 2023.

[22] P. Cong, J. Liu, J. Li, Y. Xiao, X. Chen, X. Feng, and X. Zhang, "Ydd-slam: Indoor dynamic visual slam fusing yolov5 with depth information," Sensors, vol. 23, no. 23, p. 9592, 2023.

[23] J. Han, R. Dong, and J. Kan, "Basl-ad slam: A robust deep-learning feature-based visual slam system with adaptive motion model," IEEE Transactions on Intelligent Transportation Systems, 2024.

[24] A. Tourani, H. Bavle, J. L. Sanchez-Lopez, R. M. Salinas, and H. Voos, "Marker-based visual slam leveraging hierarchical representations," in 2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2023, pp. 3461–3467.

[25] A. Tourani, H. Bavle, D. I. Avşar, J. L. Sanchez-Lopez, R. Munoz-Salinas, and H. Voos, "Vision-based situational graphs exploiting fiducial markers for the integration of semantic entities," Robotics, vol. 13, no. 7, p. 106, 2024.

[26] Y. Li, H. Zhao, X. Qi, L. Wang, Z. Li, J. Sun, and J. Jia, "Fully convolutional networks for panoptic segmentation," in Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, 2021, pp. 214–223.

[27] J. Hu, L. Huang, T. Ren, S. Zhang, R. Ji, and L. Cao, "You only segment once: Towards real-time panoptic segmentation," in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2023, pp. 17 819–17 829.

[28] K. G. Derpanis, "Overview of the ransac algorithm," Image Rochester NY, vol. 4, no. 1, pp. 2–3, 2010.

[29] H. Oleynikova, Z. Taylor, M. Fehr, R. Siegwart, and J. Nieto, "Voxblox: Incremental 3d euclidean signed distance fields for on-board mav planning," in IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2017.

[30] 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.

[31] S. Saeedi, E. D. Carvalho, W. Li, D. Tzoumanikas, S. Leutenegger, P. H. Kelly, and A. J. Davison, "Characterizing visual localization and mapping datasets," in 2019 International Conference on Robotics and Automation (ICRA). IEEE, 2019, pp. 6699–6705.

[32] X. Shi, D. Li, P. Zhao, Q. Tian, Y. Tian, Q. Long, C. Zhu, J. Song, F. Qiao, L. Song et al., "Are we ready for service robots? the openloris-scene datasets for lifelong slam," in 2020 IEEE International Conference on Robotics and Automation (ICRA). IEEE, 2020, pp. 3139–3145.

[33] A. Dai, A. X. Chang, M. Savva, M. Halber, T. Funkhouser, and M. Nießner, "Scannet: Richly-annotated 3d reconstructions of indoor scenes," in Proceedings of the IEEE conference on computer vision and pattern recognition, 2017, pp. 5828–5839.

[34] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers, "A benchmark for the evaluation of rgb-d slam systems," in 2012 IEEE/RSJ international conference on intelligent robots and systems. IEEE, 2012, pp. 573–580.

[35] F. J. Romero-Ramirez, R. Muñoz-Salinas, M. J. Marín-Jiménez, M. Cazorla, and R. Medina-Carnicer, "sslam: Speeded-up visual slam mixing artificial markers and temporary keypoints," Sensors, vol. 23, no. 4, p. 2210, 2023.