# Constraint-Based Modeling of Dynamic Entities in 3D Scene Graphs for Robust SLAM

Marco Giberna, Muhammad Shaheer, Hriday Bavle, Jose Andres Millan-Romera,
Jose Luis Sanchez-Lopez, and Holger Voos

***Abstract*— Autonomous robots depend crucially on their ability to perceive and process information from dynamic, ever-changing environments. Traditional simultaneous localization and mapping (SLAM) approaches struggle to maintain consistent scene representations because of numerous moving objects, often treating dynamic elements as outliers rather than explicitly modeling them in the scene representation. In this paper, we present a novel hierarchical 3D scene graph-based SLAM framework that addresses the challenge of modeling and estimating the pose of dynamic objects and agents. We use fiducial markers to detect dynamic entities and to extract their attributes while improving keyframe selection and implementing new capabilities for dynamic entity mapping. We maintain a hierarchical representation where dynamic objects are registered in the SLAM graph and are constrained with robot keyframes and the floor level of the building with our novel *entity-keyframe constraints* and *intra-entity constraints*. By combining semantic and geometric constraints between dynamic entities and the environment, our system jointly optimizes the SLAM graph to estimate the pose of the robot and various dynamic agents and objects while maintaining an accurate map. Experimental evaluation demonstrates that our approach achieves a $27.57\%$ reduction in pose estimation error compared to traditional methods and enables higher-level reasoning about scene dynamics.**

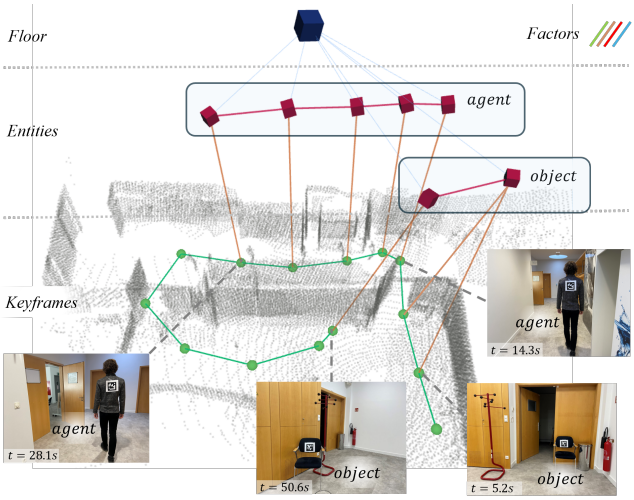



Fig. 1: Hierarchical SLAM factor graph integrating dynamic entities (a walking human and a displaced chair) with keyframes (green), entities (red), and floors (blue). Colored edges represent observation constraints. Image frames show selected keyframes.

## I. Introduction

Robots are being increasingly deployed for surveillance, reconnaissance, and object retrieval tasks that require operating in dynamic environments. Identifying environmental changes based on previous observations is essential for autonomous systems to maintain situational awareness. To navigate and interact with these environments, robots must build and maintain accurate representations of the world around them. This is addressed through Simultaneous Localization and Mapping (SLAM) techniques, which enable robots to construct maps of unknown environments while simultaneously tracking their position within them. However, most of the SLAM methods either assume the world is static or strongly simplify the dynamic world model. Indoor environments usually present a wide variety of dynamic instances, each possessing different properties. For instance, agents, i.e. humans and other robots, are expected to move frequently according to specific motion models, while objects, such as chairs, can also be sporadically relocated, thus changing the overall scene over time.

In this direction, Khronos [1] presents a unified approach to handle both the dynamics of agents and the changes in the scene resulting from the moving objects. However, they rely exclusively in raw observation, not employing any known semantic class-dependent motion models which characterize the detected dynamic entities. Furthermore, they miss in exploiting any high-level semantic relation between entities and the environment, such as a desk constrained to the floor, which could further improve the accuracy of the world model generation.

Recently, SLAM approaches [2], [3] have leveraged 3D scene graphs to model the environment in terms of different hierarchical levels. These solutions efficiently categorize, organize, and relate observable objects and higher level abstractions in the environment, like rooms, improving the SLAM accuracy. However, these methods heavily rely on the assumption that the world is semistatic or simplify the dynamic model by masking or filtering out moving entities. As a consequence, information coming from dynamic entities is neither exploited nor incorporated within the scene graphs, while an intelligent integration of them could improve their

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

* This work was funded by the Fonds National de la Recherche of Luxembourg (FNR) under the project DEFENCE22/IS/17800397/INVISIMARK.

handling as well as the accuracy of the generated world model.

To address this, we present a real-time dynamic scene graph framework capable of incorporating dynamic entities along with their category-wise semantic motion model into the scene graph. We use fiducial markers to detect dynamic entities and model them in the SLAM graph with our novel factors, as shown in Fig. 1. We constrain each instance of a dynamic entity with robot keyframes, its subsequent observations, and the floor level of the building. We then optimize the SLAM graph to accurately reconstruct the environment map, robot trajectory, and pose estimation of dynamic entities

To summarize, the primary contributions of our paper are as follows.

- A novel method to incorporate dynamic entities into 3D scene graph based SLAM.
- A novel factor to constrain dynamic entities with robot keyframes.
- A novel factor to constrain multiple instances of the same entity to estimate the change in its pose over time.
- A dynamic entity-aware SLAM system that jointly estimates robot trajectory, static environment structure, and dynamic object poses.

## II. Related work

### A. 3D Scene Graphs for SLAM

3D scene graphs [4]–[8] efficiently represent environments by encoding semantic abstractions and their relationships in a hierarchical structure. This semantic-rich representation has proven valuable for robust SLAM, enabling both geometric and object-level mapping. Recent works such as [5], [6], [8] use scene graphs for SLAM but only employ flat object-level relationships, ignoring hierarchical structure in higher semantic layers (*rooms, floors*, etc). Hydra [3] addresses this by generating real-time 3D scene graphs and linking objects with higher-level environmental abstractions such as *rooms*, *floors*, and *buildings*. They also exploit this information to improve loop closure detection and optimize the scene graphs. However, Hydra [3] does not tightly couple the SLAM backend with the generated scene graphs for joint optimization of objects and robot poses. Situational Graphs *S-Graphs* [2], [9] advance the field by implementing a four-layered optimizable factor graph tightly coupling pose estimation and the 3D scene graph generation.

In general, these methods assume that the world is static, and generate scene graphs that do not explicitly model the dynamic entities in the environment.

### B. Dynamic SLAM

Research in dynamic SLAM has advanced significantly [10]–[16]. While many approaches [14]–[16] focus on filtering out dynamic objects from SLAM systems, fewer methods attempt to explicitly model and track these dynamic elements. Work on dynamic object pose tracking exists [17]–[20] but has largely been limited to table-top environments, with limited success in larger indoor spaces with mobile robots. Other approaches [21]–[23] label the dynamic objects with respect to the probability they can move to either static, dynamic or potentially dynamic categories. In general, these works focus on detecting and elaborating on dynamics happening within the field of view of the robot, without considering possible changes in the scene that might have occurred while out of the field of view. To address this, other works such as *Dynablox* [24] and [25], [26] have been proposed for simultaneous map reconstruction and tracking dynamic entities in large environments. However, they primarily track short-term motion relative to static scenes and struggle with persistent environmental changes over time. *Khronos* [1] generalize to both dynamics happening within or outside the field of view, presenting a dynamic SLAM formulation to generate metric-semantic maps that change over time. However, they rely exclusively on raw entity observations, without exploiting semantic information distinguishing between agents and objects, their high-level semantical relations with the environment or prior motion models characterizing them. *3D DSG* [7] combines semantic reasoning with 3D scene graph modeling to track dynamic elements. However, it only tracks humans, overlooking other dynamic objects and static objects that change position over time.

To the best of our knowledge, no prior work has used 3D scene graphs to simultaneously model environmental structure, track inter-object relationships, monitor dynamic agents, and track the changing positions of static objects.

## III. Overview

### A. System Architecture

We exploit the work presented in [2] that generates a hierarchical, optimizable scene graph of the environment using a 3D LiDAR sensor. Our approach extends this framework by incorporating image data specifically for detecting and modeling dynamic entities in the scene graph, that was previously unaddressed. We exploit the floor segmentation module, and, keyframe and floor mapper modules from [2], while improving the keyframe selection, loop closure detection, and implementing new capabilities for dynamic entity detection and mapping. The architecture of the proposed method is illustrated in Fig. 2. First, the front-end (Section IV) of our method processes synchronized LiDAR scans, image frames, and odometry measurements, for entity detection and keyframe selection. *Entity detection* (Section IV-A) module extracts dynamic entities and their attributes from the raw sensor data, generating a snapshot for each detected entity instance. Based on this, the *keyframe selection* (Section IV-B) module decides when a new keyframe must be added in the graph. Then, the back-end (Section V) registers the keyframes and entities in the SLAM graph with appropriate constraints and jointly optimizes them. Finally, the graph optimizer refines the reconstructed map tightly with the poses of mapped dynamic entities. Furthermore, the *entity aware loop closure* (Section IV-C) module takes advantage of the information of the mapped entities and their eventual motion to make the loop detection more robust.

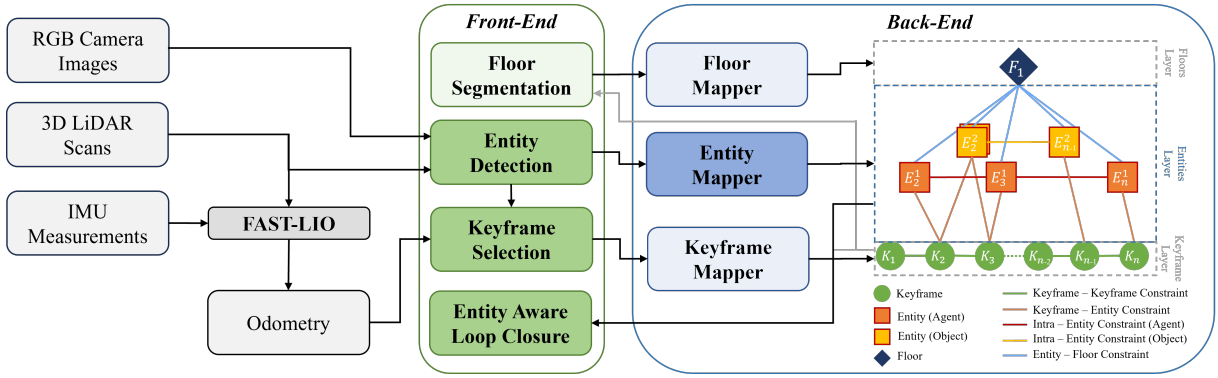



Fig. 2: **System Architecture.** The proposed system processes synchronized RGB images, LiDAR scans, IMU measurements, and odometry data to build a dynamic SLAM graph. The **front-end** (green) performs entity detection, keyframe selection, entity-aware loop closure, and floor segmentation. The **back-end** (blue) registers detected entities and keyframes in the SLAM graph and relative constraints. The **graph structure** (right) organizes the environment hierarchically, associating keyframes (green circles) with a detected agent (orange square, always observed in new position) and object (yellow squares, initially observed twice at the same position, then in a different location) and floor (blue diamond). We highlight our contributions (green and blue) with respect to the modules reused from the employed baseline (light green, light blue and gray).

## IV. Front-End

### A. Entity Detection

The entity detection module extracts entities from raw sensor data, associates them with prior observations, and classifies them semantically. This module estimates the pose of objects along with the related uncertainty, and segments their point cloud representation, obtaining an object fragment, that is, a partial view of the detected entity. Hence, the module outputs an entity snapshot at time $t$ containing the entity unique ID $i$, the semantic class $S_i \in \{agents, objects\}$, estimated entity pose ${}^{s}\tilde{\varepsilon}_{i,t}$ in the sensor frame $s$ and its related uncertainty $\sigma_{i,t}$, and the point cloud fragment $\psi_{i,t}$.

### B. Keyframe Selection

This module determines when new keyframes should be added to the SLAM graph. Our keyframes selection policy is driven by significant scene changes, identified as: the robot moving beyond a predefined threshold $\delta_R$; new entities entering the field of view, and previously mapped entities changing their pose above a threshold $\delta_E$. Moreover, to mitigate increasing uncertainty in older entity observations, we implement a timer-based mechanism. When an entity is re-detected after its timer expires, a new keyframe is registered to capture a fresh and reliable observation. The keyframe selection is fundamental to our approach, as entity observations are integrated into the map exclusively upon new keyframe registration.

### C. Entity Aware Loop Closure

To robustify loop closure detection in presence of moving agents and objects, we exploit the graph structure to remove exclusively the point cloud of the entities that have moved between the loop closure candidate keyframes.

By querying the graph structure, we can identify if the new entity has moved with respect to its previous observation. If an entity has moved, its previously stored fragment $\psi_{i,t}$ is removed from the loop closure candidate keyframe's complete scan $S_t$. If an entity remains static, its previous observations at the same position are maintained, increasing the amount of features for loop closure detection. Hence, the preprocessed scan candidates $S_t^{'}$ are:

$$\begin{aligned} \mathcal{S}_t^{'} &= \mathcal{S}_t \setminus \bigcup_{i \in \mathbf{E}_{t,inc}} \psi_{i,t} \quad \forall t = 1, \dots, T-1 \\ \mathcal{S}_T^{'} &= \mathcal{S}_T \setminus \bigcup_{i \in \bigcup_{t=1,\dots,T-1} \mathbf{E}_{t,inc}} \psi_{i,t} \end{aligned} \tag{1}$$

where $S_t$ is the original scan candidate, $\mathbf{E}_{t,inc}$ is the set of entities present in the candidate recorded at $t$ that have inconsistent poses with the current time $T$.

By continuously refining the loop closure candidates based on the entity motion, the system improves global consistency of the reconstructed map while maintaining accurate scene understanding.

## V. Back-End

The back-end is responsible for adding nodes and constraints in the graph, and optimizing the global state. Overall, our global optimization state is defined as:

$$\begin{aligned} s = \big[ {}^{M}x_{R_1}, \dots, {}^{M}x_{R_T}, {}^{M}\varepsilon_{E_{1,1}}, \dots, {}^{M}\varepsilon_{E_{N,T}}, \\ {}^{M}\xi_1, \dots, {}^{M}\xi_F, {}^{M}x_O \big]^\top \end{aligned} \tag{2}$$

where ${}^{M}x_{R_t} \in SE(3)$, $t \in \{1, \dots, T\}$ are the robot poses at T selected keyframes, ${}^{M}\varepsilon_{E_{i,t}} \in SE(3)$, $i \in \{1, \dots, N\}$, $t \in \{1, \dots, T\}$ are the poses of the $i^{th}$ entity at its observed keyframes, ${}^{M}\xi_f \in \mathbb{R}$, $f \in \{1, \dots, F\}$ are the $F$ floors levels, and ${}^{M}x_O$ models the drift between the odometry frame $O$ and the map frame $M$.

### A. Entity Mapper

When a keyframe is registered while one or more entities are in the field of view of the robot, the detected entities are

mapped. Given the detected entity pose ${}^{s}\tilde{\varepsilon}_{i,t}$ in the sensor frame $s$ and the known transformation from the robot's base link to the sensor frame ${}^{R_t}T_s$, the pose in the robot frame at time $t$ of the entity $i$ is computed as: ${}^{R_t}\tilde{\varepsilon}_{E_{i,t}} = {}^{R_t}T_s \oplus {}^{s}\tilde{\varepsilon}_{i,t}$. Using the newly registered keyframe ${}^{M}x_{E_t}$, the entity pose is then transformed into the map frame and added to the graph: ${}^{M}\tilde{\varepsilon}_{E_{i,t}} = {}^{M}x_{R_t} \oplus {}^{R_t}\tilde{\varepsilon}_{E_{i,t}}$

**Entity-Keyframe Constraint.** Each entity observation is initially constrained to the keyframe from which it was detected. The detected pose, transformed in the robot frame, is: ${}^{R_t}\tilde{\varepsilon}_{E_{i,t}} = {}^{R_t}T_s \oplus {}^{s}\tilde{\varepsilon}_{i,t}$.

The associated cost function to minimize is:

$$c_{KF-E}\left({}^{M}x_{R_1}, \dots, {}^{M}x_{R_T}, {}^{M}\varepsilon_{E_{1,1}}, \dots, {}^{M}\varepsilon_{E_{N,T}}\right) = \sum_{i=0}^{N}\sum_{t=0}^{T} || {}^{M}x_{R_t}^{-1} \oplus {}^{M}\varepsilon_{E_{i,t}} \ominus {}^{R_t}\tilde{\varepsilon}_{E_{i,t}} ||^2_{\Lambda_{\tilde{\varepsilon}_{i,t}}} \quad (3)$$

where $\oplus$ and $\ominus$ are the composition and inverse composition, $^{-1}$ is the inverse operator, $||\cdot||_{\Lambda_{\tilde{\varepsilon}_{i,t}}}$ is the Mahalanobis distance, and $\Lambda_{\tilde{\varepsilon}_{i,t}}$ is information matrix associated to the observation of the $i^{th}$ entity at time $t$. The information matrix is derived from the uncertainty previously extracted by the entity detection module: $\Lambda_{\tilde{\varepsilon}_{i,t}} = \sigma_{i,t}^{-1}$.

**Intra-Entity Constraint.** The Intra-Entity Constraint constrains two observations of the same entity observed by the robot at different times. Entity observation nodes are constrained pairwise using an expected motion model $\tilde{f}_{S_i,P_{E_{i,t}}}$, which is a function of the semantic class $S_i$, and the relative pose between the current and previous entity observations defined as, omitting some indices for sake of clarity, $P_{E_{i,t}} = {}^{M}\varepsilon_{i,t-1} \ominus {}^{M}\tilde{\varepsilon}_{i,t}$.

The associated cost function is:

$$c_{E-E}\left({}^{M}\varepsilon_{E_{1,1}}, \dots, {}^{M}\varepsilon_{E_{N,T}}\right) = \sum_{i=0}^{N}\sum_{t=0}^{T} || {}^{M}\varepsilon_{E_{i,t-1}}^{-1} \oplus {}^{M}\varepsilon_{E_{i,t}} \ominus \tilde{f}_{S_i,P_{E_{i,t}}} ||^2_{\Lambda_{\tilde{f}_{S,E_{i,t}}}} \quad (4)$$

where $\Lambda_{\tilde{f}_{S,E_{i,t}}}$ is the information matrix associated to the motion model $\tilde{f}_{S,E_{i,t}}$.

For objects, the motion model first checks whether the object has moved, accounting for unavoidable estimation noise. If the relative pose remains below a fixed, experimentally determined noise threshold $\nu$, it is assumed static. Otherwise, the computed relative pose is employed. We hence define the motion model $\tilde{f}_{obj,E_{i,t}}$ for $i^{th}$ entities in the objects class at time $t$:

$$\tilde{f}_{obj,E_{i,t}} = \begin{cases} I_4 & \text{if } ||{}^{M}\varepsilon_{E_{i,t-1}}^{-1} \oplus {}^{M}\varepsilon_{E_{i,t}}|| < \nu \\ P_{E_{i,t}} & \text{otherwise} \end{cases} \quad (5)$$

For agents, movement is generally assumed, simplifying the motion model definition for entities in the agents class to:

$$\tilde{f}_{agents,E_{i,t}} = P_{E_{i,t}} \quad (6)$$

**Entity-Floor Constraint.** Entities are assumed to lie on the ground, therefore their $z$ coordinate remain constant over time. We hence constrain the entity observation with the current floor node by maintaining constant the initially observed relative vertical position ${}^{M}\xi_j\tilde{z}_{E_{i,t}}$ .

The associated cost function is:

$$c_{F-E}\left({}^{M}\xi_1, \dots, {}^{M}\xi_F, {}^{M}\varepsilon_{E_{1,1}}, \dots, {}^{M}\varepsilon_{E_{N,T}}\right) = \sum_{i=0}^{N}\sum_{t=0}^{T} || \left({}^{M}\xi_j\right)_z \ominus \left({}^{M}\varepsilon_{E_{i,t}}\right)_z \ominus {}^{M}\xi_j\tilde{z}_{E_{i,t}} ||^2_{\Lambda_{F-E}} \quad (7)$$

where $\xi_j$ is the current floor, $(\cdot)_z$ denotes the $z$ coordinate, $\Lambda_{F-E}$ is the information matrix associated to the entity-floor constraints.

## VI. Experimental Evaluation

### A. Experimental Setup

We evaluated our algorithm in both simulated and real environments. All experiments were performed using a laptop computer with an Intel i9-12900H (8 cores, 2.5 GHz) with 32 GB of RAM. For ease of implementation, we exploit AprilTag [27] fiducial markers for detection of dynamic entities, their pose estimation and classification. This approach allows us to focus on our core contribution while considering the perception, segmentation and data association problem across multiple observations to be solved and outside the scope of this paper.

**Baseline.** To the best of our knowledge, there are no works that model and jointly optimize the environmental structure, objects and agents poses over time, and the robot trajectory simultaneously. For this reason, we assess the contributions of our developed modules with respect to the ground truth. In addition, we compare the improvement in robot trajectory estimation with respect to the baseline [2] we built on top of.

**Simulated Dataset.** We evaluate the our approach in four simulated datasets, each with different dynamics: Moving Agents (*MA*) or Static Agents (*SA*) combined with Static Objects (*SO*) or Moved Objects (*MO*). This results in four distinct datasets: SASOS, SAMOS, MASOS, MAMOS; whereas the final *S* stands for simulated dataset. All datasets were generated using the Gazebo physics simulator, which accurately models the robot platform, LiDAR sensor, RGB camera, and a 3D indoor environment. The environment contains 11 objects and 3 agents distributed throughout multiple rooms. In scenarios with objects displacement, 7 out of 11 objects are randomly relocated after initial environment exploration. In moving agent scenarios, a human agent continuously traverses the environment. We quantitatively assess performance using Absolute Trajectory Error (ATE) compared against ground truth trajectories. Additionally, we demonstrate the effectiveness of our approach in jointly estimating and optimizing robot trajectory, static background, and dynamic entity poses, measuring the progressive improvement in entity pose accuracy over time.

**Real Dataset.** We collected data with a handheld device equipped with Ouster OS1-64 3D LiDAR, a Realsense camera D435i and an Intel Nuc10 (i7 CPU) computer. We collected the data in our university hall environment featuring seven chairs arranged throughout the space, with a human subject and a legged robot moving among them. Each dataset

captures different dynamic scenarios, including stationary or moving agents, as well as objects that remain static, are relocated or are rotated. Our datasets capture diverse dynamic scenarios, systematically varying the motion states of both agents and objects. We categorize datasets based on agent status — Static Agents (*SA*) or Moving Agents (*MA*) — combined with object conditions — Static Objects (*SO*), Rotated Objects (*RO*), or Moved Objects (*MO*) This results in four distinct datasets: *SAMOR*, *SAROR*, *MASOR* and *MAMOR*, whereas the final *R* identifies the real dataset. The eventual rotation and relocation of objects happen after they have been detected for the first time. We precisely measured all object positions at the beginning of each experiment and after any subsequent relocations. All measurements were recorded relative to the dataset starting point, which serves as the origin in our reconstructed maps. This ground truth enables us to evaluate the accuracy of our joint optimization approach.

### *B. Results and Discussion*

**Simulated Data.** Table I presents the estimated robot trajectory error across various simulation datasets. The experiments evaluate the impact of different modules in the SLAM pipeline, considering both individual and combined effects, comparing it to the employed baseline, that is [2]. The first three setups differ in terms of which factors are implemented: *Setup 1* incorporates only the keyframe-entity constraints, resulting in a 28.97% reduction in average error compared to the baseline; *Setup 2* also integrates the intra-entity constraints, achieving an 18.97% improvement; while *Setup 3* introduces constraints between entities and the floor semantic entity, leading to a 25.53% improvement. These configurations aim to assess the relative contribution of each factor to overall trajectory estimation performance. A subset of possible combinations of factors have been selected to ensure graph connectivity.

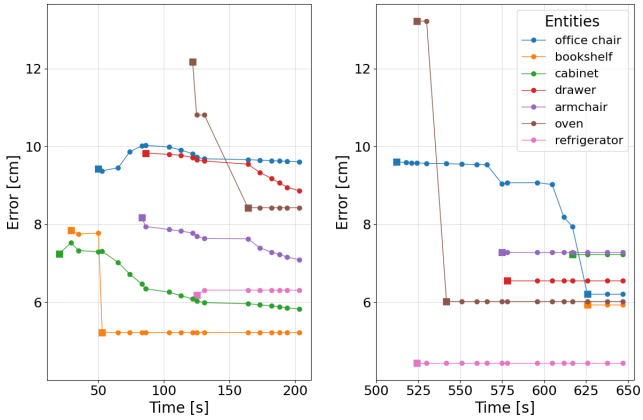
*Setups 4* and *5* analyze the effect of removing entity point clouds for loop closure detection. *Setup 4* retains all entity point clouds regardless of motion, showing a 9.43% improvement, while *Setup 5* removes them unconditionally, leading to an 8.06% degradation. These results confirm that maintaining static entity observations benefits loop closure performance.

Our proposed approach (*Ours w/o Timer*) integrates all constraints on observed entities while selectively removing entity point clouds for loop closure detection only when movement is detected. This method achieves an average error reduction of 22.19%. The final configuration (*Ours*) further introduces the entity timer within the keyframe registration policy, resulting in a 27.57% improvement in trajectory estimation accuracy with respect to the baseline.

The results indicate that incorporating entity constraints within the keyframes enhances trajectory estimation accuracy compared to the baseline. When entity-entity constraints are enforced, performance also improves, particularly in environments without moving agents. This suggests the effectiveness of the entity-entity constraints how they are defined for objects, while the agent motion model may require additional refinements, for instance leveraging velocity information.

Examining *Setups 4* and *5*, we observe that indiscriminately removing entity point clouds degrades performance in datasets where objects remain static. However, in environments with dynamic objects, the impact is less pronounced, and performance even slightly improves when both agents and objects move. These findings highlight the importance of conditionally removing entity point clouds based on detected motion rather than applying a fixed policy.

Considering the complete system, integrating all modules yields the best trajectory estimation performance in terms of mean error and, generally, standard deviation, especially in scenarios with changing-pose objects, but also when both agents and objects are static. This reinforces the fact that the object motion model is well defined. On the other side, in moving agents scenarios, the performances tend to improve less significantly. This reaffirms the agent motion model requires further work in its definition in order to make the overall system consistently benefiting also from the agents' integration. Additionally, we see that incorporating the entity timer improves performance because of the introduction of fresh observations which mitigate the higher uncertainty characterizing older data. This effectively reduces the accumulated uncertainty, ensuring more reliable and effective optimization.




(a) Before Moving Objects (b) After Moving Objects

Fig. 3: **Joint Optimization Effect.** Estimated entity pose error with respect to the ground truth over time of *Ours* method, before (3a) and after (3b) objects are relocated and/or rotated in the *MAMOS* dataset. Squares indicate time instants when a new observation of an entity has been added to the map. In the second plot, errors are shown only after the entity has been detected again in its new position.

Figure 3 presents the Euclidean norm of the estimated entity pose errors relative to the ground truth over time of *Ours* method in the *MAMOS* dataset. Only data corresponding to moved objects is reported in the figures, specifically before moving them in Figure 3a and after they have been moved in Figure 3b. We hence report the trend of the estimated entity pose error over time in relation to time and the addition of

| Setup | Enabled Modules | | | | | | | Datasets | | | | | | | | Average | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Constraints | | | Entity PC Removal | | | Updater Policy | *SASOS* | | *MASOS* | | *SAMOS* | | *MAMOS* | | | | |
| | KF-E | intra-E | E-F | No | Always | Moved | Timer | Mean [cm] | Std [cm] | Mean [cm] | Std [cm] | Mean [cm] | Std [cm] | Mean [cm] | Std [cm] | Mean [cm] | Std [cm] | Relative Variation |
| Baseline | | | | ✓ | | | | 10.358 | 7.116 | 12.126 | 10.149 | 20.124 | 17.023 | 7.193 | 6.612 | 12.451 | 10.226 | - |
| Setup 1 | ✓ | | | | | ✓ | | 8.035 | 5.606 | 14.973 | 10.973 | 6.492 | **6.895** | **5.877** | **7.871** | **8.844** | 7.836 | **-28.97%** |
| Setup 2 | ✓ | ✓ | | | | ✓ | | 7.355 | 6.535 | 18.659 | 12.378 | 7.325 | 7.422 | 7.016 | 8.733 | 10.089 | 8.767 | -18.97% |
| Setup 3 | ✓ | | ✓ | | | ✓ | | 8.232 | **5.444** | **14.819** | **8.776** | 7.902 | 7.772 | 6.136 | 8.096 | 9.272 | **7.522** | -25.53% |
| Setup 4 | ✓ | ✓ | ✓ | ✓ | | | | 7.264 | 7.091 | 17.283 | 11.162 | 12.956 | 14.052 | 7.144 | 8.735 | 11.277 | 10.338 | -9.43% |
| Setup 5 | ✓ | ✓ | ✓ | | ✓ | | | 14.934 | 8.882 | 17.745 | 11.475 | 14.658 | 12.349 | 6.941 | 9.067 | 13.454 | 10.365 | +8.06% |
| **Ours w/o Timer** | ✓ | ✓ | ✓ | | | ✓ | | 7.256 | 6.572 | 14.892 | 9.717 | 8.575 | 9.832 | 6.523 | 9.053 | 9.687 | 8.966 | -22.19% |
| **Ours** | ✓ | ✓ | ✓ | | | ✓ | ✓ | **7.158** | 6.759 | 16.395 | 10.408 | **5.479** | 7.197 | 7.040 | 8.525 | 9.018 | 8.222 | -27.57% |

TABLE I: **Robot's Average Trajectory Error (ATE) Comparison and Ablation Study.** Experimental results retrieved on the simulated datasets using different setups, each specifying which modules have been enabled, with respect to the baseline. The table reports the mean error (Mean [cm]), standard deviation (Std [cm]), and, for the average results across all the datasets, the relative variation of the mean error with respect to the baseline. The datasets differ in terms of which entity class (Agent or Object) show dynamic properties (Static or Moving) while the robot explores the environment. **Bold** values are the best and the second best are underlined.

new observations. Figure 3a illustrates how pose estimates are significantly affected by the graph optimizer over time, with errors generally decreasing over time for most detected entities. In contrast, Figure 3b depicts less pronounced changes, indicating that when revisiting a previously mapped environment, the joint optimization has a smaller impact on entity pose estimates. Nonetheless, in both cases, adding new observations to the map (represented by squares in the plots) consistently and immediately reduces pose estimation error, supporting the importance of incorporating new observations even when objects remain stationary, for example due to the timer policy. Notably, the office chair, which underwent only rotation, maintains a stable error with respect to its last pose estimates before the environmental modifications, and subsequent optimization further improves its accuracy. In general, when detecting an entity in a new position, the estimated pose error tends to be lower with respect to its first observation, showing that exploiting past observations of an entity even in a different location improves the accuracy, since they can still bring information such as the expected vertical position. The only encountered exception in this dataset is the oven entity, where the first observation after relocation exhibits a higher error due to a challenging point of view - being detected from a greater distance and at a steep angle, leading to high uncertainty. In fact, a later observation from a more advantageous point of view quickly and significantly improves its pose estimate. Therefore, we can affirm that the system benefits from maintaining both current and past observations of objects and agents within the SLAM optimization process, eventually enhancing overall situational awareness.

**Real Data.** Table II presents the entity pose estimation results, reporting the Euclidean norm of the error with respect to the ground truth. The error is measured at two key moments: when each entity is first added to the map (*first*), and after the final optimization at the end of the exploration (*last*). Across all datasets, the optimization process, along with additional observations acquired during exploration, generally reduces the error between the estimated and actual poses of the entities. This trend aligns with the results observed in the simulated datasets. The *MASO* dataset exhibits the most significant improvement over time, as the objects remain in fixed positions, allowing new observations to directly refine the estimated poses of the mapped entities. Comparing the datasets in which objects have changed their positions (*SAMO* and *MAMO*), although they both show considerable improvements in the entity pose estimation accuracy, it is notable that in absence of moving agents the system is able to better refine with more precision the estimated poses over time. This affirms that, even though with minor effects, constantly moving agents makes the overall optimization more challenging. Finally, we note that when the objects are only rotated in the *SARO* dataset, the system struggles more in correctly estimating the new pose of the entities, up to worsening the pose estimation for two (entities 2 and 3)

| Entity ID | Estimated Entity Pose Error [cm] | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | *SARO* | | *SAMO* | | *MASO* | | *MAMO* | |
| | First | Last | First | Last | First | Last | First | Last |
| 1 | 16.7 | 9.8 | 20.2 | 6.0 | 7.6 | 6.0 | 10.6 | 8.4 |
| 2 | 2.1 | 5.8 | 5.1 | 5.9 | 2.5 | 2.1 | 10.3 | 4.9 |
| 3 | 1.7 | 8.0 | 6.3 | 2.6 | 7.7 | 5.3 | 6.4 | 5.2 |
| 4 | 13.4 | 11.9 | 6.4 | 2.5 | 2.2 | 3.0 | 9.9 | 8.2 |
| 5 | 8.8 | 4.9 | 8.3 | 5.0 | 5.4 | 0.9 | 5.3 | 2.0 |
| 6 | 9.3 | 6.9 | 2.5 | 6.3 | 8.4 | 1.0 | 0.9 | 4.3 |
| 7 | 4.2 | 3.8 | 2.1 | 1.4 | 3.3 | 0.9 | 7.8 | 4.4 |
| **Average** | 8.0 | 7.3 | 7.3 | 4.3 | 4.2 | 1.9 | 7.3 | 5.4 |

TABLE II: **Real World Experimental Results.** Norm error with respect to ground truth for each entity, when first added to the map and after the last optimization step. Entities 4, 5, 6 and 7 are moved in the *SAMO* and *MAMO* datasets, while entities 1, 2 and 3 are rotated in the *SARO* dataset.

out of the three rotated objects. This is due to the fact that the model employed for computing uncertainty in the entity detector is mainly modeled for translational inaccuracy, thus under performing in this case. A better uncertainty estimation in the detection module can mitigate this.

| **Setup** | **Computation Time** [ms] | | |
|---|---|---|---|
| | *Sim* | *Real* | *Average* |
| Baseline | 27 | 19 | 23 |
| Setup 1 | 71 | 131 | 101 |
| Setup 2 | 63 | 118 | 91 |
| Setup 3 | 65 | 108 | 87 |
| **Ours w/o Timer** | 61 | 88 | 74 |
| **Ours** | 71 | 91 | 81 |

TABLE III: **Computation Time.** Average time [ms] taken by each optimization cycle across all employed simulated (*Sim*) and *Real* datasets, and the overall *Average*.

**Computation Time.** Table III shows the computation time of our algorithm compared with the aforementioned setups and baseline. The computation time is defined as the time required for each global graph optimization cycle. The baseline achieves the fastest performance due to the lower number of states being optimized, as it does not incorporate entities in the graph. When entities are included, *Ours* method results on average the fastest, showing that integrating all the presented constraints simplifies and speeds up the optimization step. The results show that our algorithm is able to run in real time in both simulated and real scenarios.

## VII. Limitations and Future Works

As mentioned in section IV-A, our approach relies on a threshold for dynamic object detection across consecutive observations. As a result, translations or rotations below this threshold cannot be detected by our system. Future work will focus on improving the detection of subtle motion changes. Secondly, we also plan to implement fiducial marker-free perception using image segmentation and object tracking in future iterations. We intentionally omitted the room layer from [2] to better evaluate the impact of dynamic entities in the SLAM graph. Future directions include reintroducing this semantic layer with high-level constraints (e.g., associating desks with offices, kitchen objects with kitchens), which would provide crucial information for solving association challenges.

## VIII. Conclusion

In this paper, we presented a novel 3D scene graph-based SLAM framework that models and estimates the pose of both static objects that change position over time and dynamic agents in indoor environments. By extending *S-Graphs+* [2] through the incorporation of dynamic entity detection and modeling, our approach offers significant improvements in mapping and localization accuracy and robustness. Our key contributions include a 3D scene graph representation that explicitly models static objects with changing positions while preserving their identity, modeling dynamic entities as factors in the pose graph, and jointly estimating robot trajectory, static environment structure, and dynamic entities poses. Experimental results across multiple simulated and real-world datasets demonstrate that our approach achieves superior performance in environments with varying dynamics, reducing pose estimation error by $27\%$ compared to *S-Graphs+*, which treats dynamic elements as outliers. Our system effectively handles scenarios with both moved objects and moving agents, maintaining consistent environmental representations even under significant dynamic changes, enabling and enhancing the robot's temporal situational awareness.

## References


[1] L. Schmid, M. Abate, Y. Chang, and L. Carlone, "Khronos: A unified approach for spatio-temporal metric-semantic slam in dynamic environments," in *Proc. of Robotics: Science and Systems*, 2024.

[2] 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, Aug. 2023, conference Name: IEEE Robotics and Automation Letters. [Online]. Available: https://ieeexplore.ieee.org/document/10168233/?arnumber=10168233

[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 XVIII*. Robotics: Science and Systems Foundation, June 2022. [Online]. Available: http://www.roboticsproceedings.org/rss18/p050.pdf

[4] I. Armeni, Z.-Y. He, A. Zamir, J. Gwak, J. Malik, M. Fischer, and S. Savarese, "3D Scene Graph: A Structure for Unified Semantics, 3D Space, and Camera," in *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, Oct. 2019, pp. 5663–5672, iSSN: 2380-7504. [Online]. Available: https://ieeexplore.ieee.org/document/9008302/?arnumber=9008302

[5] U.-H. Kim, J.-M. Park, T.-j. Song, and J.-H. Kim, "3-D Scene Graph: A Sparse and Semantic Representation of Physical Environments for Intelligent Agents," *IEEE Transactions on Cybernetics*, vol. 50, no. 12, pp. 4921–4933, Dec. 2020, conference Name: IEEE Transactions on Cybernetics. [Online]. Available: https://ieeexplore.ieee.org/document/8796400

[6] 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.

[7] 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 XVI*. Robotics: Science and Systems Foundation, July 2020. [Online]. Available: http://www.roboticsproceedings.org/rss16/p079.pdf

[8] J. Wald, H. Dhamo, N. Navab, and F. Tombari, "Learning 3d semantic scene graphs from 3d indoor reconstructions," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 3961–3970.

[9] 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, Oct. 2022, conference Name: IEEE Robotics and Automation Letters. [Online]. Available: https://ieeexplore.ieee.org/document/9826367

[10] N. Brasch, A. Bozic, J. Lallemand, and F. Tombari, "Semantic monocular slam for highly dynamic environments," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 393–400.

[11] L. Cui and C. Ma, "Sof-slam: A semantic visual slam for dynamic environments," *IEEE access*, vol. 7, pp. 166 528–166 539, 2019.

[12] B. Bescos, J. M. Fácil, J. Civera, and J. Neira, "Dynaslam: Tracking, mapping, and inpainting in dynamic scenes," *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 4076–4083, 2018.

[13] Y. Qiu, C. Wang, W. Wang, M. Henein, and S. Scherer, "Airdos: Dynamic slam benefits from articulated objects," in *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, 2022, pp. 8047–8053.

[14] S. Song, H. Lim, A. J. Lee, and H. Myung, "Dynavins: a visual-inertial slam for dynamic environments," *IEEE Robotics and Automation Letters*, vol. 7, no. 4, pp. 11 523–11 530, 2022.

[15] C. Yu, Z. Liu, X.-J. Liu, F. Xie, Y. Yang, Q. Wei, and Q. Fei, "Ds-slam: A semantic visual slam towards dynamic environments," in *2018 IEEE/RSJ international conference on intelligent robots and systems (IROS)*. IEEE, 2018, pp. 1168–1174.

[16] P. Yu, C. Guo, y. Liu, and H. Zhang, "Fusing semantic segmentation and object detection for visual slam in dynamic scenes," in *Proceedings of the 27th ACM Symposium on Virtual Reality Software and Technology*, 2021, pp. 1–7.

[17] R. Long, C. Rauch, T. Zhang, V. Ivan, and S. Vijayakumar, "Rigidfusion: Robot localisation and mapping in environments with large dynamic rigid objects," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 3703–3710, 2021.

[18] M. Rünz and L. Agapito, "Co-fusion: Real-time segmentation, tracking and fusion of multiple objects," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2017, pp. 4471–4478.

[19] M. Strecke and J. Stuckler, "Em-fusion: Dynamic object-level slam with probabilistic data association," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 5865–5874.

[20] B. Xu, W. Li, D. Tzoumanikas, M. Bloesch, A. Davison, and S. Leutenegger, "Mid-fusion: Octree-based object-level multi-instance dynamic slam," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 5231–5237.

[21] Y. Gao, M. Hu, B. Chen, W. Yang, J. Wang, and J. Wang, "Multi-Mask Fusion-Based RGB-D SLAM in Dynamic Environments," *IEEE Sensors Journal*, pp. 1–1, 2024. [Online]. Available: https://ieeexplore.ieee.org/document/10599144/

[22] X. Du, C. Zhang, K. Gao, J. Liu, X. Yu, and S. Wang, "YPL-SLAM: A Simultaneous Localization and Mapping Algorithm for Point–line Fusion in Dynamic Environments," *Sensors*, vol. 24, no. 14, p. 4517, July 2024. [Online]. Available: https://www.mdpi.com/1424-8220/24/14/4517

[23] S. Peng, T. Ran, J. Zhang, W. Xiao, and L. Yuan, "STS-SLAM: Joint Visual SLAM and Multi-Object Tracking Based on Spatio-Temporal Similarity," *IEEE Transactions on Intelligent Vehicles*, pp. 1–15, 2024. [Online]. Available: https://ieeexplore.ieee.org/document/10559403/

[24] L. Schmid, O. Andersson, A. Sulser, P. Pfreundschuh, and R. Siegwart, "Dynablox: Real-time Detection of Diverse Dynamic Objects in Complex Environments," *IEEE Robotics and Automation Letters*, vol. 8, no. 10, pp. 6259–6266, Oct. 2023, arXiv:2304.10049 [cs]. [Online]. Available: http://arxiv.org/abs/2304.10049

[25] B. Mersch, T. Guadagnino, X. Chen, I. Vizzo, J. Behley, and C. Stachniss, "Building volumetric beliefs for dynamic environments exploiting map-based moving object segmentation," *IEEE Robotics and Automation Letters*, 2023.

[26] Y. Ren, B. Xu, C. L. Choi, and S. Leutenegger, "Visual-inertial multi-instance dynamic slam with object-level relocalisation," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2022, pp. 11 055–11 062.

[27] J. Wang and E. Olson, "AprilTag 2: Efficient and robust fiducial detection," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, October 2016.