

DYNEMO-SLAM: Dynamic Entity and Motion-Aware 3D Scene Graph SLAM

Marco Giberna, Muhammad Shaheer, Miguel Fernandez-Cortizas, Jose Andres Millan-Romera, Jose Luis Sanchez-Lopez, and Holger Voos

Abstract— Robots operating in dynamic environments face significant challenges due to the presence of moving agents and displaced objects. Traditional SLAM systems typically assume a static world or treat dynamic as outliers, discarding their information to preserve map consistency. As a result, they cannot exploit dynamic entities as persistent landmarks, do not model and exploit their motion over time, and therefore quickly degrade in highly cluttered environments with few reliable static features. This paper presents a novel 3D scene graph-based SLAM framework that addresses the challenge of modeling and estimating the pose of dynamic entities into the SLAM backend. Our framework incorporates semantic motion priors and dynamic entity-aware constraints to jointly optimize the robot trajectory, dynamic entity poses, and the surrounding environment structure within a unified graph formulation. In parallel, a dynamic keyframe selection policy and a semantic loop-closure prefiltering step enable the system to remain robust and effective in highly dynamic environments by continuously adapting to scene changes and filtering inconsistent observations. The simulation and real-world experimental results show a 49.97% reduction in ATE compared to the baseline method employed, demonstrating the effectiveness of incorporating dynamic entities and estimating their poses for improved robustness and richer scene representation in complex scenarios while maintaining real-time performance.

I. INTRODUCTION

Robots operate in dynamic environments with moving agents and occasionally displaced objects, making traditional SLAM approaches fail due to the lack of reliable static landmarks [1]. Most of these methods either assume the world is static or simplify the dynamic SLAM problem by tracking and processing dynamics separately. Consequently, they rely solely on known-to-be-static features in the environment, failing whenever actual static landmarks are scarce. To mitigate this, a body of work has attempted to incorporate moving [2], quasi-static [3], or occasionally displaced landmarks [4]. However, these approaches use such cues only locally and do not maintain movable entities as persistent landmarks that could support loop closures or drift reduction over revisits. They are also restricted to a specific motion class and fail to generalize when the environment exhibits mixed dynamics, such as both moving agents and displaced objects. A central limitation is the absence of semantic reasoning and prior knowledge to select an appropriate motion model for each detected entity. Additionally, they miss in exploiting any high-level semantic

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, miguel.fernandez, 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.

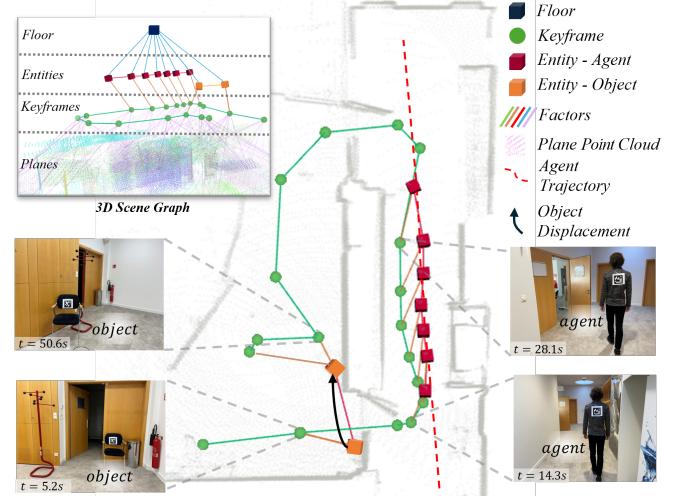


Fig. 1: Hierarchical SLAM factor graph integrating dynamic entities (a walking human and a displaced chair) with keyframes, entities, and floor. Image frames show selected keyframes.

relation between dynamic entities and the environment, such as a desk vertically constrained to the unmovable floor, which could further improve the map reconstruction accuracy. This suggests there is a need for models that capture both motion and high-level semantics to inform the SLAM back-end.

Recently, SLAM approaches [5], [6] have leveraged 3D scene graphs to model the environment in terms of different hierarchical levels. These solutions efficiently categorize, organize, and relate observable instances to higher-level abstractions, like rooms and floors, improving the SLAM accuracy, but assuming the world is static or simplifying the dynamic model by masking or filtering out moving entities. As a result, valuable information from the dynamic entities remains unused in scene graphs, limiting both robustness in dynamic environments and the expressiveness of the world model.

To the best of our knowledge, this is the first dynamic 3D graph SLAM (Fig. 1) framework that tightly couples 3D scene graphs with the SLAM backend to jointly model environmental structure, track robot trajectory, and monitor both dynamic agents and displaced objects. This is achieved through semantic motion priors and relational constraints that enable joint optimization of robot trajectory, entity poses, and environmental structure while exploiting semantics, high-level relational information, and their features. Through experimental validation, we demonstrate that our proposed method is able to generalize from static to highly dynamic environment in a real-time fashion, consistently outperforming the baseline in terms of localization accuracy, while simultaneously esti-

mating poses of the detected dynamic entities.

The primary contributions of our paper are as follows.

- A *real-time dynamic entity-aware SLAM system* that jointly estimates robot trajectory, static environment structure, and dynamic entity poses, detecting motion, and reusing entities as landmarks.
- A *dynamic keyframe selection policy* that registers new keyframes based on significant situational changes, such as robot motion, detection of new entities or pose changes of mapped entities, and incorporates a timer to periodically refresh static observations.
- A *set of factors* to constrain dynamic entities with robot keyframes; multiple observations of the same entity to estimate its pose over time via semantic-dependent motion models; and dynamic entities with the structural environment exploiting hierarchy and high-level abstractions.
- A *dynamic situationally-aware enhanced loop closure detection*, which filters loop closure candidates via semantic and temporal consistency, removing only features from moved entities to improve closure precision.

II. RELATED WORK

A. Dynamic SLAM

Research in dynamic SLAM has advanced significantly, with most works [7], [8], [9] focusing on detecting and filtering out dynamic entities to preserve the consistency of static-world SLAM systems, thereby losing valuable data and failing in highly-dynamic environments. Beyond filtering, some methods attempt to explicitly model and track dynamic elements. Object-level pose tracking approaches [10], [11], [12] have been proposed, attempting to unify SLAM and tracking process in a coupled framework. However, they miss in reasoning about the semantics of the tracked entities and using prior knowledge about their possible motion capabilities for improving tracking and consequentially the SLAM process. To this end, other approaches [13], [14] classify landmarks into static, dynamic, or potentially dynamic categories, allowing probabilistic reasoning about their stability but using this information exclusively to mask out unreliable moving features, missing in exploiting their moving features integrated with their inferred motion profiles in subsequent observations and revisits to improve self-localization and mapping.

A complementary line of research explores the use of moving or displaced entities as landmarks themselves within a graph optimization problem. Humans and other agents have been modeled as continuously moving landmarks whose trajectories can be jointly estimated with the robot's pose [2], [15], [16]. Similarly, displaced objects have been exploited as scene-consistent features when their motion is sporadic [4] or quasi-static [3]. These works highlight the potential of enriching SLAM beyond static landmarks by leveraging entities that follow different temporal motion profiles. However, such existing factor graph formulations typically separate static, dynamic, and potentially dynamic elements, generating independent pose graphs, rather than unifying them into a single representation, hence missing in jointly optimizing entity poses, self-trajectory, and the static background. Furthermore, they fail in mixed-dynamics scenarios because of the lack of

semantic reasoning to differentiate between possible motions. This leaves a gap in exploiting altogether the information provided by both persistently moving agents characterized by known motion models and intermittently displaced objects for long-term mapping and localization.

Recent advances have begun to address short-term dynamics and long-term environmental changes simultaneously. In this direction, *Khronos* [17] generalizes both the dynamics that occur within and outside the robot's field of view, presenting a formulation to generate metric-semantic maps that evolve over time. However, it does not fully integrate the dynamic entities in the SLAM pipeline to improve self-localization, as it does not model them as factors which relate geometrically and semantically the dynamic object trajectories with those of the robot. Moreover, it relies solely on raw observation, lacking the integration of any known semantic class-dependent motion models that characterize the detected dynamic entities.

B. 3D Scene Graphs for SLAM

3D scene graphs [18], [19] efficiently represent environments by encoding semantic abstractions and their relationships in a relational structure. This semantic-rich representation has proven valuable for robust SLAM, enabling geometric and object-level mapping. Recent works such as [20], [21], [22] use scene graphs for SLAM but only employ flat object-level relationships, ignoring hierarchical structure in higher semantic layers (*rooms*, *floors*, etc). *Hydra* [6] addresses this by generating real-time 3D scene graphs and linking objects with higher-level environmental abstractions such as *rooms*, *floors*, and *buildings*. They exploit this information to improve loop closure detection and optimize the scene graphs. However, *Hydra* [6] does not tightly couple the SLAM backend with the generated scene graphs for joint optimization of objects and robot poses. *Situational Graphs* [5] advance the field by implementing a four-layered optimizable factor graph tightly coupling pose estimation and 3D scene graph generation. These methods assume that the world is static and generate scene graphs that do not explicitly model the dynamic entities in the environment, furthermore not exploiting any dynamic or potentially dynamic entity to improve loop closure detection. *3D DSG* [19] combines semantic reasoning with scene graphs to represent and track dynamic elements, yet not integrating them into the SLAM backend. Moreover, it focuses exclusively on humans, overlooking other moving or repositioned objects.

III. OVERVIEW

A. System Architecture

The architecture of the proposed method is illustrated in Fig. 2. First, the front-end (Sec. IV) of our method processes synchronized LiDAR scans, image frames, and odometry measurements for keyframe selection, plane segmentation and entity detection. The *plane segmentation* module extracts planar surfaces from the LiDAR scans, while the *floor segmentation* module calculates floor center using all the currently extracted planes as in [5]. The *entity detection* module (Sec. IV-A) extracts dynamic entities and their attributes, generating a snapshot for each detected entity instance. Based on detected entities and odometry measurements, the *keyframe selection*

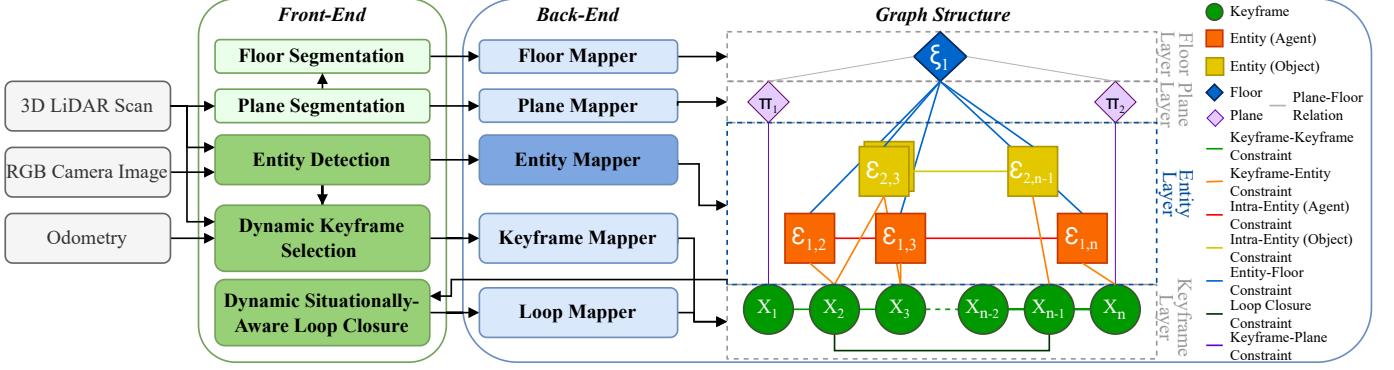


Fig. 2: **System Architecture.** The proposed system processes synchronized RGB images, LiDAR scans, and odometry data to build a SLAM graph. The **front-end** (green) performs entity detection, keyframe selection, plane segmentation, entity-aware loop closure, and floor segmentation. The **back-end** (blue) registers detected entities and keyframes in the SLAM graph and relative constraints. The **graph structure** (right) organizes the environment hierarchically, associating keyframes with a detected agent (always observed in new position in the depicted scenario) and object (initially observed twice at the same position, then in a different location), planes and floor. We highlight our contributions (green and blue) with respect to the modules reused from the employed baseline [5] (light green, light blue and gray).

module (Sec. IV-B) decides when a new keyframe should be added to the graph. Next, the back-end (Sec. V) registers keyframes, planes, floor and correlated entities in an optimizable hierarchical scene graph comprising keyframe, plane, entity, and floor layers, and jointly optimizes them. Furthermore, the *dynamic entity aware loop closure* module (Sec. IV-C) enhances scan matching-based loop detection by identifying entities with inconsistent poses across candidate keyframes and removing their corresponding point cloud fragments.

IV. FRONT-END

A. Entity Detection

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

B. Dynamic Keyframe Selection

The keyframe selection module determines when a new keyframe, which is a collection of the timestamp t , odometry measurement ${}^Ox_{R_t}$, LiDAR scan S_t and image frame at time t , should be added to the SLAM graph. This module is fundamental to our approach, as entity observations are integrated into the map exclusively upon new keyframe registration. Our keyframes selection policy is driven by significant *situational changes*, identified as: the robot moving beyond a predefined threshold δ_R (Eq. 1); new entities being detected for the first time (Eq. 2), and previously mapped entities changing their pose above a threshold δ_E , which depends on the entity pose detection accuracy (Eq. 3). Hence the situational changes check at current time T is:

$$\|{}^Ox_{R_{T-1}} \ominus {}^Ox_{R_T}\| > \delta_R \quad \vee \quad (1)$$

$$\exists i \in \mathcal{I}_T \setminus \mathcal{I}_{\text{map}} \quad \vee \quad (2)$$

$$\|{}^M\varepsilon_{i,t}^{-1} \oplus {}^M\tilde{\varepsilon}_{i,T}\| > \delta_E \quad \forall i \in \mathcal{I}_T \quad (3)$$

where $^{-1}$ is the inverse operator, \oplus and \ominus are the composition and inverse composition, $\|\cdot\|$ is the Euclidean norm over the Lie algebra, which accounts for both translational and rotational parts, ${}^Ox_{R_{T-1}} \in SE(3)$ is the odometry measurement in correspondence of the previous registered keyframe, \mathcal{I}_{map} is the set of entity IDs already mapped in the SLAM graph, ${}^M\varepsilon_{i,t} \in SE(3)$ is the last observation of the same entity available in the graph, and ${}^M\tilde{\varepsilon}_{i,T} = {}^Mx_O \oplus {}^Ox_{R_T} \oplus {}^{R_T}T_s \oplus {}^s\tilde{\varepsilon}_{i,T}$ is the detected entity pose in the map frame. ${}^Mx_O \in SE(3)$ models the drift between the odometry frame O and the map frame M , and ${}^{R_T}T_s \in SE(3)$ is the transformation from the robot's base link to the sensor frame s .

At this point, entities that remain static are registered again only when they are observed at the same time as any of the aforementioned conditions is triggered. However, static entities carry valuable information, since repeated observations at the same location can both refine their pose estimate and strengthen the graph via additional closed-loop constraints. To exploit this, we add a timer-based mechanism as an additional condition: when a static entity is re-detected after its timer expires, a new keyframe is registered. This captures a fresher, more reliable observation and reduces the accumulated uncertainty of older detections.

C. Dynamic Entity-Aware Loop Closure

We employ scan matching-based loop closure constraints at keyframe level, following the approach in [5]. To robustify loop closure detection and to improve scan alignment in the presence of moving agents and objects, we remove exclusively the point cloud fragments of entities that have moved between the candidate keyframes for loop closure. By querying the graph structure, we identify if the new entity has an inconsistent pose with respect to its previous observations. The pose at time t of entity i is considered inconsistent with the new

pose at time T and therefore added to the set $\mathbf{E}_{t,inc}$, if:

$$\|{}^M\varepsilon_{E_{i,t}} \ominus {}^M\varepsilon_{E_{i,T}}\| > \delta'_E \Rightarrow i \in \mathbf{E}_{t,inc} \quad (4)$$

where ${}^M\varepsilon_{E_{i,t}} \in SE(3)$ is the pose on the map frame of the entity i at time t extracted from the optimized graph, $\|\cdot\|$ is the Euclidean norm over the Lie algebra, δ'_E is a distance threshold that accounts for pose estimation accuracy.

At time T , for all pairs of loop closure original candidate scans ($\mathcal{S}_t, \mathcal{S}_T$), if the entity i , registered to both keyframes, has an inconsistent pose, its previously stored fragment $\psi_{i,t}$ and the new fragment $\psi_{i,T}$ are subtracted from the original scans of the loop closure candidates \mathcal{S}_t and \mathcal{S}_T . Hence, the preprocessed scan candidates \mathcal{S}'_t and \mathcal{S}'_T are:

$$\begin{aligned} \mathcal{S}'_t &= \mathcal{S}_t \setminus \bigcup_{i \in \mathbf{E}_{t,inc}} \psi_{i,t} \quad \forall t = 1, \dots, T-1 \\ \mathcal{S}'_T &= \mathcal{S}_T \setminus \bigcup_{i \in \bigcup_{j=t, \dots, T-1} \mathbf{E}_{j,inc}} \psi_{i,T} \end{aligned} \quad (5)$$

By continuously refining the loop closure candidates based on the entity motion, the system mitigates the incorrect loop closure detection and the imprecise scan matching.

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:

$$\mathbf{s} = [{}^Mx_{R_1}, \dots, {}^Mx_{R_T}, {}^M\varepsilon_{E_{1,1}}, \dots, {}^M\varepsilon_{E_{N,T}}, {}^M\pi_1, \dots, {}^M\pi_P, {}^M\xi_1, \dots, {}^M\xi_F, {}^Mx_O]^\top \quad (6)$$

where ${}^Mx_{R_t} \in SE(3)$, $t \in \{1, \dots, T\}$ 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, ${}^M\pi_p$, $p \in \{1, \dots, P\}$ are the plane parameters of the P planes in the scene as in [5], ${}^M\xi_f \in \mathbb{R}$, $f \in \{1, \dots, F\}$ are the F floors levels, and Mx_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 ${}^Mx_{R_t}$, the entity pose is then transformed into the map frame and added to the graph: ${}^M\tilde{\varepsilon}_{E_{i,t}} = {}^Mx_{R_t} \oplus {}^{R_t}\tilde{\varepsilon}_{E_{i,t}}$

Keyframe-Entity Constraint. Each entity observation is initially constrained to the keyframe from which it was detected. The associated cost function to minimize is:

$$\begin{aligned} c_{KF-E} ({}^Mx_{R_1}, \dots, {}^Mx_{R_T}, {}^M\varepsilon_{E_{1,1}}, \dots, {}^M\varepsilon_{E_{N,T}}) &= \\ \sum_{i=0}^N \sum_{t=0}^T \|{}^Mx_{R_t}^{-1} \oplus {}^M\varepsilon_{E_{i,t}} \ominus {}^{R_t}\tilde{\varepsilon}_{E_{i,t}}\|_{\Lambda_{\tilde{\varepsilon}_{i,t}}}^2 \end{aligned} \quad (7)$$

where $\|\cdot\|_{\Lambda_{\tilde{\varepsilon}_{i,t}}}$ 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 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 collected by

the robot at different keyframes. 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 clarity, $P_{E_{i,t}} = {}^M\varepsilon_{i,t-1} \ominus {}^M\tilde{\varepsilon}_{i,t}$. The associated cost function is:

$$\begin{aligned} c_{E-E} ({}^M\varepsilon_{E_{1,1}}, \dots, {}^M\varepsilon_{E_{N,T}}) &= \\ \sum_{i=0}^N \sum_{t=0}^T \|{}^M\varepsilon_{E_{i,t-1}}^{-1} \oplus {}^M\varepsilon_{E_{i,t}} \ominus \tilde{f}_{S_i, P_{E_{i,t}}}\|_{\Lambda_{\tilde{f}_{S_i, E_{i,t}}}}^2 \end{aligned} \quad (8)$$

where $\Lambda_{\tilde{f}_{S_i, E_{i,t}}}$ is the information matrix associated to the motion model $\tilde{f}_{S_i, E_{i,t}} \in SE(3)$.

The motion model is defined depending on the extracted entity's semantic class S_i and, when applicable, on the specific entity instance, as agents may exhibit distinct motion behaviors modeled accordingly. For objects, the motion model first checks whether the object has moved, accounting for estimation noise. If the relative pose remains below a fixed, experimentally determined noise threshold ν , 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 :

$$\tilde{f}_{obj, E_{i,t}} = \begin{cases} I_4 & \text{if } \|{}^M\varepsilon_{E_{i,t-1}} \ominus {}^M\varepsilon_{E_{i,t}}\|_{\Lambda_{\tilde{f}_{S_i, E_{i,t}}}} < \nu \\ P_{E_{i,t}} & \text{otherwise} \end{cases} \quad (9)$$

where $\|\cdot\|_{\Lambda_{\tilde{f}_{S_i, E_{i,t}}}}$ is the Mahalanobis distance. The motion model for entities in the agent class $\tilde{f}_{agents, E_{i,t}}$ can be any function which incorporates any prior knowledge of the trajectory of agent i at time t . If this prior is not available, we simply set the motion model:

$$\tilde{f}_{agents, E_{i,t}} = P_{E_{i,t}} \quad (10)$$

allowing for tracking the agent's motion evolution over time. In case of agents following a known straight trajectory since the last observation, the motion model incorporates this prior by projecting the relative pose $P_{E_{i,t}}$ onto the motion subspace defined by the unit direction vector \mathbf{u} :

$$\tilde{f}_{agents, E_{i,t}} = (\mathbf{u}\mathbf{u}^\top)P_{E_{i,t}} \quad (11)$$

Only the translational component along \mathbf{u} is retained, while the orthogonal translational and rotational components are set to zero and down-weighted, ensuring the residual reflects motion consistent with the expected linear path.

Floor-Entity Constraint. It can be assumed that entities within the same floor keep a constant distance with respect to the ground, therefore, their z coordinate remain constant over time. We hence constrain the entity observation with the current floor node by maintaining constant relative vertical position ${}^M\xi_j \tilde{z}_{E_{i,t}}$ across observations. The associated cost function is:

$$\begin{aligned} c_{F-E} ({}^M\xi_1, \dots, {}^M\xi_F, {}^M\varepsilon_{E_{1,1}}, \dots, {}^M\varepsilon_{E_{N,T}}) &= \\ \sum_{i=0}^N \sum_{t=0}^T \|{}^M\xi_j \ominus ({}^M\varepsilon_{E_{i,t}})_z \ominus {}^M\xi_j \tilde{z}_{E_{i,t}}\|_{\Lambda_{F-E}}^2 \end{aligned} \quad (12)$$

where ξ_j is the current floor state, $(\cdot)_z$ denotes the z coordinate, Λ_{F-E} is the entity-floor information matrix.

TABLE I: **Ablation Study.** Enabled modules, introduced in Sections IV and V, for each setup with respect to the baseline. KF, E, and F stand respectively for keyframe, entity and floor.

Setup	Contributed Modules					
	Constraints		Entity PC Removal		Keyframe Selection	
	V-A	IV-C	No Always	Conditional	IV-B	
KF-E	intra-E	F-E	No	Always	Conditional	Dynamic Policy
<i>Baseline</i> [5]			✓			
<i>only KF-E</i>	✓		✓			
<i>intra-E</i>	✓	✓	✓			
<i>F-E</i>	✓	✓	✓			
<i>Full Constraint Set (FCS)</i>	✓	✓	✓			
<i>always EPCR</i>	✓	✓	✓			
<i>Motion-Based (MB-EPCR)</i>	✓	✓	✓	✓		
Full System (Full)		✓	✓	✓	✓	✓

VI. EXPERIMENTAL EVALUATION

To the best of our knowledge, no existing SLAM method jointly models and optimizes the robot trajectory, environmental structure, and the time-varying poses of both objects and agents, rather focusing purely on, e.g., entity reconstruction [17] or entity tracking [12]. Furthermore, our framework combines LiDAR with image-based semantic cues, making direct comparison with purely vision-based [7] dynamic SLAM approaches inherently unfair. Since no prior work addresses dynamic environments with a comparable multi-sensor and joint-optimization formulation, we restrict our comparison to the baseline we build upon.

A. Assumptions

For ease of implementation, we exploit AprilTag [23] fiducial markers for detection of dynamic entities, their pose estimation and classification. This approach allows us to focus on our core contributions while considering the perception, segmentation and data association problem across multiple observations to be solved and outside the scope of this paper. The uncertainty of the entity detection $\sigma_{i,t}$ is quantified based on the subpixel corner localization error obtained by marker detection. In our experiments, agents are moving in an almost straight line fashion. Therefore, we incorporate this prior into the related motion model as presented in Eq.11.

B. 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. In all our experiments, the odometry is estimated through LiDAR odometry algorithm as in [5]. The entity's point cloud fragments are extracted using KD-tree-based Euclidean clustering.

Metrics. We quantitatively assess performance using Absolute Trajectory Error (ATE) comparing trajectories with ground truth to validate our approach's SLAM capabilities in environments with different degrees of dynamicity. Additionally, we demonstrate the effectiveness of our approach in jointly estimating and optimizing dynamic entity poses together with the robot trajectory and static background, measuring the error of the entity pose estimation with respect of the groundtruth over time. Finally, we report the average optimization time for each dataset and setup, and with respect to the number of nodes in the graph to show our approach's real-time capability.

TABLE II: **Datasets.** Summary of the employed datasets. Each dataset name encodes the type of dynamics it contains: stationary (*SA-*) or moving agents (*MA-*), and objects that remain static (*-SO*), are relocated (*-MO*) or only rotated (*-RO*).

Dataset	#agents	#objects	#moved obj.	Env.	Area [m ²]	Length [s]
<i>S-SASO</i>	-	11	-	<i>sim_1</i>	123	373
<i>S-SAMO</i>	-	11	7	<i>sim_1</i>	123	644
<i>S-MASO</i>	129	15	-	<i>sim_2</i>	675	351
<i>S-MAMO</i>	129	15	9	<i>sim_2</i>	675	324
<i>S-MASO2</i>	51	6	-	<i>sim_3</i>	225	111
<i>S-MASO3</i>	51	6	-	<i>sim_3</i>	225	292
<i>R-SARO</i>	-	8	5	<i>real_1</i>	90	111
<i>R-SAMO</i>	-	8	5	<i>real_1</i>	90	173
<i>R-MASO</i>	3	8	-	<i>real_1</i>	90	93
<i>R-MAMO</i>	3	8	5	<i>real_1</i>	90	324

Ablation. We assess the contributions of our developed modules by comparing them against each other. Table I summarizes the employed setups and indicates which modules are enabled in each case. The first four setups differ in terms of which factors are implemented: *only KF-E* incorporates only the keyframe-entity constraints; *intra-E* also integrates the intra-entity constraints; while *F-E* introduces constraints between entities and the floor semantic entity. A subset of possible combinations of factors has been selected to ensure graph connectivity. Finally, Full Constraint Set (*FCS*) setup, includes all proposed factors. Setups *FCS*, *always EPCR* (Entity Point Cloud Removal) and Motion-Based (*MB-*) *EPCR* analyze the effect of never, always, or conditionally removing dynamic entity point clouds before loop closure detection. The final configuration (**Full**) further introduces the entity timer within the dynamic keyframe registration policy, which is implemented in every setup but the baseline.

Simulated Dataset. We evaluate our approach in six simulated (*S-*) datasets, spanning three different kinds of indoor environment, each presenting different dynamics that are moving agents and/or relocated objects. Table II summarizes the characteristics of the datasets. *sim_1* consists in a 7 rooms small plan rich in furniture, used to validate our approach in a static environment and with displaced objects. *sim_2* is composed of three large rooms saturated with moving humans, presenting a total of 15 boxes scattered around, to assess our method in an environment poor of static features. *sim_3* is a large single room saturated with moving humans and 6 static boxes randomly placed, to test our algorithm in extremely cluttered scenarios poor of static features. In *S-MASO2* the robot explores the entire room just once, while in *S-MASO3* it repeats the same trajectory three times, to test our approach with and without revisits. In datasets with objects displacement (*S-SAMO* and *S-MAMO*), around 60% of the objects are randomly relocated after initial environment exploration. In moving agent scenarios, humans continuously traverse the environment following a straight trajectory. All datasets were generated using the Gazebo physics simulator.

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 recorded real (*R-*) datasets in a university hall (*real_1*) featuring seven chairs arranged throughout the space, with two humans and a legged robot moving among them. Eventual rotation or relocation of objects occur only after the initial environment

TABLE III: **Robot’s Average Trajectory Error (ATE) Comparison and Computation Time.** Results across six simulated datasets. For each dataset we report RMSE [cm], Std [cm], number of detected loop closures (#LC), and average computation time [ms]. Lower values are better, except for the number of detected loop closures (#LC). **Bold** values are best, and underlined values are second-best.

Setup	S-SASO				S-MASO				S-SAMO				S-MAMO				S-MASO2				S-MASO3			
	RMSE [cm]		Std [cm]	#LC [ms]	RMSE [cm]		Std [cm]	#LC [ms]	RMSE [cm]		Std [cm]	#LC [ms]	RMSE [cm]		Std [cm]	#LC [ms]	RMSE [cm]		Std [cm]	#LC [ms]	RMSE [cm]		Std [cm]	#LC [ms]
	12.567	7.116	7	37	31.066	9.721	6	12	24.156	17.025	5	24	32.192	18.842	5	10	25.990	10.188	0	10	21.550	9.982	10	9
<i>only KF-E</i>	9.970	5.614	7	77	24.733	<u>9.161</u>	6	26	13.693	11.487	7	78	18.920	10.087	6	31	24.904	16.074	0	15	18.963	9.923	11	36
<i>intra-E</i>	9.049	5.147	7	77	23.945	9.900	6	39	13.746	10.052	7	80	17.705	10.059	6	33	21.963	10.606	0	16	18.493	9.595	11	33
<i>F-E</i>	9.282	5.184	7	77	24.747	9.073	4	25	13.175	10.568	7	80	19.051	10.246	5	28	21.031	9.302	0	16	18.379	9.473	11	47
<i>FCS</i>	7.244	5.009	7	51	24.187	9.322	6	27	13.511	9.490	7	71	16.856	9.925	6	29	21.986	10.614	0	13	17.308	<u>7.982</u>	11	28
<i>always EPCR</i>	17.375	8.882	7	52	27.437	9.605	6	28	16.281	12.446	7	63	<u>15.313</u>	9.678	5	30	22.074	10.719	0	16	17.476	8.202	11	52
<i>MB-EPCR</i>	<u>7.176</u>	4.395	7	58	<u>24.097</u>	9.297	6	26	12.757	8.794	7	71	<u>16.157</u>	9.428	6	28	18.776	7.912	1	16	<u>17.142</u>	7.833	11	28
Full	6.963	<u>4.477</u>	8	51	24.157	9.309	6	31	9.045	7.197	6	78	15.211	9.226	5	30	13.810	<u>8.127</u>	1	25	17.098	8.627	11	52

exploration. Table II summarizes the employed datasets.

C. Results and Discussion

Impact of the Constraints. Tables III and IV present the Absolute Trajectory Error (ATE) results obtained in the simulated and real-world experiments. Across all datasets, adding geometric/semantic constraints consistently improves trajectory accuracy over the baseline. Among single-factor variants, *intra-E* and *F-E* generally outperform *only KF-E* by on average 4.62% and 4.02% across all datasets respectively, indicating that (i) exploiting multiple, temporally adjacent observations of the same entity stabilizes the estimate over time, and (ii) anchoring entities to structural semantics (e.g., the floor) provides a strong geometric prior—particularly in scenes with moving agents or sparse static features. Delving deeper, we observe how *intra-E* factors are particularly beneficial in presence of moving agents (*MA*-), demonstrating that in such challenging environment, where the baseline struggles the most, anchoring to static landmarks helps the system reducing possible drift induced by the moving entities, being this appreciable also in static objects datasets (-*SO*). Similarly, modeling moving entities expected trajectories directly into the factor graph mitigates potential accumulating error deriving from the detection process. In fact, this is evident in the real experiments (*R-MA*-), whereas the trajectory might deviate more easily with respect to the expected one, thus struggling more in these scenario, although suggesting as a better tuning of the covariances of both motion model and entities detection that better accounts the less-predictable nature of human trajectories might be beneficial to the system. When integrating the full set of constraints (*FCS*), the system attains ATE improvements in most simulated sets, and achieves even better performances on real data, confirming that jointly enforcing keyframe-entity, intra-entity, and floor-entity constraints yields the most robust solution.

The entity pose error trends are report in Fig. 3: **Full** starts comparably to other setups but converges to lower translational/rotational errors as more observations are incorporated, while *F-E* is typically second-best due to its semantic anchoring, creating a direct link in the optimization process of the static background with all the entities, and showing how incorporating semantic relations improves the pose estimates. The *only KF-E* setup performs poorly since the pose estimate relies exclusively on the last observation, while the *intra-E* setup achieves better results over time, especially after new observations come in, weighting all the previous observations.

This shows that keeping multiple recent observations of an entity in the optimization graph improves its pose estimate over time. Although adding new observations to the graph does not necessarily bring an improvement in the pose estimate, we see that an eventual worsening is greatly mitigated by the implementation of intra-entity constraints. Such eventual degradation can be due to less accurate or more noisy entity observations, such as being detected from greater distance or at a steep angle, leading to higher uncertainty. These results also highlight that keeping multiple recent observations in the graph (*intra-E*) mitigates degradation when new detections are noisier or at challenging viewpoints.

Impact of Entity Point-Cloud Removal. The effect of pre-processing loop-closure candidates by removing entity point clouds depends on scene dynamics (Tables III and IV). In static or mostly static environments, *always EPCR* consistently harms performance due to reduced feature availability for place recognition, whereas *FCS* can help by retaining all features. Conversely, in dynamic scenes (moved objects and/or moving agents), removing entity points before loop closure reduces outlier matches and can bring modest gains. Overall, conditionally applying EPCR provides the best trade-off. This behavior is reflected by the superior ATE of the **Full** system (*MB-EPCR*), where the dynamic, motion-aware EPCR improves loop-closure quality (accuracy of accepted closures) rather than quantity (#LC), which often remains similar across EPCR variants. This improvement is especially observable in the real datasets, resulting on 24.91% and 54.28% average improvement over *FCS* and *always EPCR*, respectively. In contrast, performance in the simulated datasets was less pronounced, as the entity point cloud fragment extraction occasionally failed due to the limitations of the implemented segmentation method. Nevertheless, it still achieves 4.93% and 17.12% improvement over *FCS* and *always EPCR*, respectively.

Impact of the Dynamic Keyframe Updater Policy. The dynamic keyframe updater policy increases the likelihood of inserting keyframes when environmental changes occur, improving situation awareness and keeping the graph populated with fresh, informative constraints. This is visible from consistent ATE gains of *only KF-E* over Baseline in every dataset, as the main different of this setup with the baseline is the keyframe selection policy, while the additional nodes constrained to the respective keyframe bring minimal new information. This is particularly observable in the moved

TABLE IV: Robot’s Average Trajectory Error (ATE) Comparison and Computation Time. Results across four real datasets. For each dataset we report RMSE [cm], Std [cm], number of loop closures (#LC), and average computation time [ms]. Lower values are better, except for the number of detected loop closures (#LC). **Bold** values are best and underlined are second best.

Setup	R-SARO				R-MASO				R-SAMO				R-MAMO			
	RMSE [cm]		Std [cm]		#LC		Time [ms]		RMSE [cm]		Std [cm]		#LC		Time [ms]	
	RMSE [cm]	Std [cm]	#LC	Time [ms]	RMSE [cm]	Std [cm]	#LC	Time [ms]	RMSE [cm]	Std [cm]	#LC	Time [ms]	RMSE [cm]	Std [cm]	#LC	Time [ms]
Baseline [5]	17.768	13.758	1	16	8.297	3.265	1	18	13.700	4.222	1	13	9.165	4.263	2	19
only KF-E	7.352	2.203	1	68	7.042	2.321	2	117	11.409	2.467	2	203	4.943	<u>1.884</u>	2	113
intra-E	5.441	2.811	1	139	8.019	2.365	2	81	10.356	<u>2.387</u>	2	121	5.989	2.012	2	118
F-E	8.442	2.797	1	107	7.384	2.338	2	86	9.435	2.849	2	125	4.684	2.102	2	108
FCS	5.583	2.947	1	93	6.607	1.749	2	59	5.404	2.911	2	89	5.002	1.933	2	78
always EPCR	9.164	3.575	1	96	9.209	2.598	2	76	12.820	2.699	2	81	5.919	2.162	2	82
MB-EPCR	5.557	<u>2.723</u>	1	76	<u>3.525</u>	<u>1.619</u>	2	67	4.470	<u>2.359</u>	2	49	3.413	<u>1.601</u>	2	88
Full	5.454	2.951	2	86	<u>3.795</u>	1.580	2	94	4.705	2.472	2	101	<u>4.317</u>	2.165	3	91

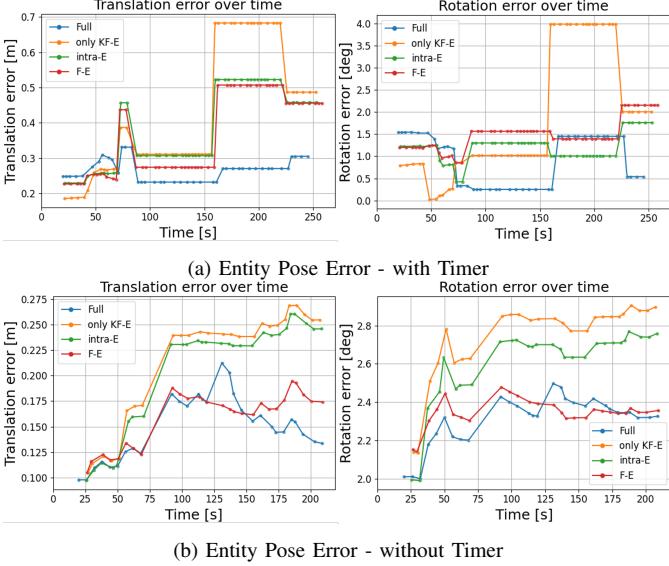


Fig. 3: Entity Pose Estimation Error and Joint Optimization Effect. Estimated entity pose error, translational and rotational, with respect to the ground truth over time of **Full** method and the setups as presented in Table I without timer in subfigure 3b and all implementing the timer in subfigure 3a in the dataset *S-MASO3*.

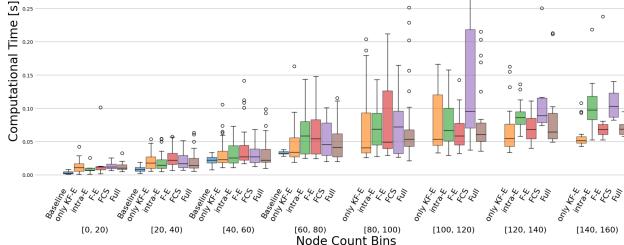


Fig. 4: Optimization time [s] over the number of nodes in the factor graph for each setup and the baseline, whereas data is available, across all employed datasets.

objects (-MO) scenarios, as registering new keyframes in correspondence of time instants in which any change in the environment is detected enrich the graph of valuable and up-to-date information through additional keyframes. A similar trend is observed in the *S-MASO* and *S-MAMO* datasets, where the robot remained stationary at several points. Thanks to the dynamic keyframe updater policy, new keyframes were still generated in these positions, not due to robot motion but as a result of changes occurring in the surrounding environment, thus maintaining an up-to-date scene understanding as soon as it restarted. Our keyframe selection policy tends to increase the quantity of loop-closure detections in dynamic scenarios,

as systematically generating a greater number of keyframes taking in correspondence of situational changes facilitates the loop detection process. In simulation, the implementation of the timer mechanism improves the ATE of **Full** over *MB-EPCR* in most datasets and reduced uncertainty through time, improving ATE of 10.22% on average. The timer is particularly beneficial in short trajectories with limited revisits (e.g., *S-MASO2*), where additional entity observations would otherwise be scarce. In real data, *MB-EPCR* can match or slightly outperform **Full** in compact environments densely populated with entities, since multiple entities often co-occur in single frames; this naturally yields frequent, fresh observations even without timer-triggered insertions, making the two policies nearly equivalent.

Focusing on the entity pose estimation error, Fig. 3 shows that when the timer is disabled the graph can be less resilient to transient degradations, whereas the timer helps maintain stability by regularly injecting up-to-date entity constraints. Such degradations, as well as possible later improvements, are due to the accumulated error in the robot trajectory estimation, which has repercussion on the static environment reconstruction and dynamic entity pose estimation, through the joint optimization procedure. Table V reports the entity pose errors at the time of first detection and after the final optimization in the real-world experiments. The results show on average a consistent reduction in error over time, confirming that incorporating additional observations and jointly optimizing robot and entity poses enhances overall estimation accuracy. Static objects benefit the most from multiple observations acquired from different points of view, allowing the optimization to leverage the most reliable ones for improved consistency. In contrast, the accuracy of moved objects depends on whether a sufficient number of high-quality observations are available before and after motion, highlighting the importance of observation diversity within the graph for achieving robust and accurate estimates.

Computation Time. Tables III and IV report the computation times for all setups. The average optimization time in simulated scenarios is 45ms for **Full** method, and 99ms in real ones, showing that our proposed framework runs real time.

Figure 4 shows the graph optimization time taken by our algorithm compared with the aforementioned setups and baseline with respect to the number of nodes in the graph. We can observe as in correspondence of a similar amount of nodes in the graph to be optimized, the type and amount of activated factors speed up the process. In fact, the optimization time

TABLE V: **Entity Pose Estimation Error.** Norm error with respect to ground truth for each entity running ***Full*** method, when first added to the map and after the last optimization step in the real-world datasets. Entities 4, 5, 6 and 7 are moved in *R-SAMO* and *R-MAMO*, while entities 1, 2 and 3 are rotated in *R-SARO*.

Entity ID	<i>R-SARO</i>		<i>R-SAMO</i>		<i>R-MASO</i>		<i>R-MAMO</i>	
	First	Last	[cm]	[cm]	First	Last	[cm]	First
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

initially increases with respect to the number of nodes consistently across all the approaches, up to showing evident discrepancies starting from over 60 nodes. The *only KF-E* setup generally achieves the fastest time, because of the absence of multiple constraints (closed loops) on the entity nodes. Adding floor-entity or, especially, intra-entity constraints increases the computation time as multiple constraints are now effecting the same state, and clearly this recurs in *FCS* setup, whereas both factors occur together. Notably, when adding the timer, in ***Full*** approach, we can generally observe a consistently faster optimization time, suggesting that adding redundant observations and hard constraints on unmoved objects simplifies and speed up the optimization process. Moreover, conversely to *FCS* and *intra-E* setups which tend to substantially increase in optimization time for larger graph size, ***Full*** flattens out, showing even less variability.

VII. CONCLUSION

We presented a 3D scene graph-based SLAM framework that jointly estimates the poses of both static and dynamic entities, including moved objects and moving agents. Comprehensive evaluations show that the best performance of our system is achieved through the combined use of all proposed modules. The set of factor constraints (*Keyframe–Entity*, *intra–Entity*, and *Floor–Entity*) provides the first substantial improvement in terms of robot trajectory and entity pose estimation by enforcing temporal consistency across observations, exploiting semantic motion prior, and leveraging structural relations with the environment. Building on this, the Dynamic Entity-Aware loop-closure filtering introduces an additional gain by removing only those entities that have moved between candidate keyframes, enabling more accurate loop closures in dynamic areas, proving especially effective in highly dynamic and cluttered environment. Finally, the dynamic keyframe selection policy with the entity-aware timer brings major improvement by registering keyframes when meaningful situational changes occur, keeping entity observations fresh and limiting temporal uncertainty growth. By exploiting sparse yet information-rich cues, the system anchors to the few static landmarks available to mitigate drift, even when the robot remains stationary and the surroundings evolve, achieving robust localization and mapping in challenging scenarios while maintaining real-time performance. Future works will focus on enabling marker-free detection and incorporating higher-level semantic layers, e.g., rooms, to enhance scene understanding and to address the

entity association problem. In addition, we plan to validate our approach using more complex motion models.

REFERENCES

- [1] H. Bavle, J. L. Sanchez-Lopez, C. Cimarelli, A. Tourani, and H. Voos, “From SLAM to Situational Awareness: Challenges and Survey,” *Sensors*, 2023.
- [2] B. Bescos, C. Campos, J. D. Tardós, and J. Neira, “Dynaslam ii: Tightly-coupled multi-object tracking and slam,” *IEEE Robotics and Automation Letters*, 2021.
- [3] A. Deeb, M. Seto, and Y.-J. Pan, “Piecewise-deterministic Quasi-static Pose Graph SLAM in Unstructured Dynamic Environments,” *Journal of Intelligent & Robotic Systems*, 2022.
- [4] A. Walcott-Bryant, M. Kaess, H. Johannsson, and J. J. Leonard, “Dynamic pose graph SLAM: Long-term mapping in low dynamic environments,” in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2012.
- [5] H. Bavle, J. L. Sanchez-Lopez, M. Shaheer, J. Civera, and H. Voos, “S-Graphs+: Real-Time Localization and Mapping Leveraging Hierarchical Representations,” *IEEE Robotics and Automation Letters*, 2023.
- [6] 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*, 2022.
- [7] B. Bescos, J. M. Fácil, J. Civera, and J. Neira, “Dynaslam: Tracking, mapping, and inpainting in dynamic scenes,” *IEEE Robotics and Automation Letters*, 2018.
- [8] S. Song, H. Lim, A. J. Lee, and H. Myung, “Dynavins: a visual-inertial slam for dynamic environments,” *IEEE Robotics and Automation Letters*, 2022.
- [9] P. Pfreundschuh, H. F. Hendrikx, V. Reijgwart, R. Dubé, R. Siegwart, and A. Cramariuc, “Dynamic object aware lidar slam based on automatic generation of training data,” in *IEEE International Conference on Robotics and Automation (ICRA)*, 2021.
- [10] 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*, 2021.
- [11] 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.
- [12] T. Behrens, R. Zurbrügg, M. Pollefeyns, Z. Bauer, and H. Blum, “Lost & found: Tracking changes from egocentric observations in 3d dynamic scene graphs,” *IEEE Robotics and Automation Letters*, 2025.
- [13] 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*, 2024.
- [14] 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*, 2024.
- [15] 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.
- [16] J. Morris, Y. Wang, and V. Ila, “The Importance of Coordinate Frames in Dynamic SLAM,” in *IEEE International Conference on Robotics and Automation (ICRA)*, 2024.
- [17] 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.
- [18] 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 *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019.
- [19] 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*, 2020.
- [20] 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*, 2020.
- [21] S.-C. Wu, J. Wald, K. Tateno, N. Navab, and F. Tombari, “Scenegraph-fusion: Incremental 3d scene graph prediction from rgb-d sequences,” in *Proceedings of IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021.
- [22] J. Wald, H. Dhamo, N. Navab, and F. Tombari, “Learning 3d semantic scene graphs from 3d indoor reconstructions,” in *Proceedings of IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020.
- [23] 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)*, 2016.