arXiv:2508.06283v1 [cs.RO] 8 Aug 2025

Situationally-aware Path Planning Exploiting 3D Scene Graphs

Saad Ejaz, Marco Giberna, Muhammad Shaheer, Jose Andres Millan-Romera, Ali Tourani, Paul Kremer, Holger Voos, and Jose Luis Sanchez-Lopez

Abstract*—3D Scene Graphs integrate both metric and semantic information, yet their structure remains underexploited for improving path planning efficiency and interpretability. In this work, we present S-Path, a Situationally-aware Path planner that leverages the metric-semantic structure of indoor 3D Scene Graphs to significantly enhance planning efficiency. S-Path follows a two-stage process: it first performs a search over a semantic graph derived from the scene graph to yield a human-understandable high-level path. This also identifies relevant regions for planning, which later allows the decomposition of the problem into smaller, independent subproblems that can be solved in parallel. We also introduce a replanning mechanism that, in the event of an infeasible path, reuses information from previously solved subproblems to update semantic heuristics and prioritize re-use to further improve the efficiency of future planning attempts. Extensive experiments on both real-world and simulated environments show that S-Path achieves average reductions of 5.7x in planning time while maintaining comparable path optimality to classical sampling-based planners, and surpassing them in complex scenarios, making it an efficient and interpretable path planner for environments represented by indoor 3D scene graphs.

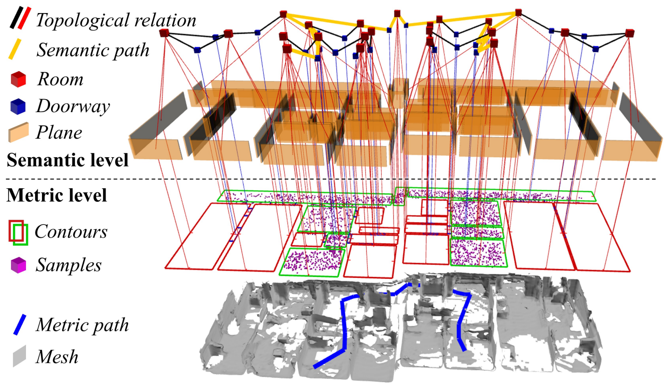


Fig. 1: S-Path augments sampling-based planners with high-level metric-semantic scene understanding to restrict sampling to only the relevant regions (green contours), thereby reducing planning time in indoor environments.

I. Introduction

Traditional path planners for indoor robot navigation rely on dense sampling of the configuration space (C-space) to find feasible paths, but their computational cost grows rapidly with its size and complexity, limiting efficiency and usability. To mitigate this, some planners [1, 2] incorporate heuristics to sample *intelligently*, but these heuristics often perform poorly in indoor environments characterized by narrow passageways and limited lines of sight. Moreover, such planners typically lack semantic understanding, making integration with human-intuitive reasoning and commands challenging. To address these limitations, several approaches have incorporated heuristics based on semantic information, such as objects [3, 4] and terrain types [5, 6, 7] into the planning process to improve both efficiency and interpretability. However, these methods rely on non-heirarchical, low-level representations, which suffer from scalability and ambiguity issues when applied to large and structurally complex environments.

3D scene graphs [8] are emerging as efficient data structures that encode spatial relationships between semantic entities in a hierarchical manner, thereby enabling effective organization of complex spaces [9, 10]. Recent research has explored the use of 3D scene graphs for task and motion planning [11, 12, 13]; however, they focus solely on plan feasibility, operate predominantly within the task planning domain, and fail to leverage the potential of high-level semantics (e.g., *rooms*, *doorways*, *floors*) to improve path planning efficiency.

To address this gap, we introduce S-Path, a *Situationally-aware Path planner* that takes advantage of semantic information from indoor scene graphs to significantly improve planning efficiency while providing human-interpretable path planning. By exploiting the structure of the scene graph, S-Path decomposes the global planning problem into smaller, independent subproblems by performing a search on the high-level semantic layers of the scene graph. These subproblems can then be solved in parallel on a geometric level using classical sampling-based planners, which operate significantly faster on the restricted C-space. Moreover, in the event of environmental changes, S-Path can swiftly replan by focusing only on the affected portions of the problem, while reusing

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

* This work was funded, in whole or in part, by the Fonds National de la Recherche of Luxembourg (FNR), DEUS Project (Ref. C22/IS/17387634/DEUS), INVISIMARK project (Ref. DEFENCE22/IS/17800397/INVISIMARK), and RoboSAUR project (Ref. 17097684/RoboSAUR). In addition, it was partially funded by the Institute of Advanced Studies (IAS) of the University of Luxembourg - "Audacity" grant (project TRANSCEND - 2021), and by SATORI project (SATORI/8_euROBIN).

For the purpose of Open Access, the author has applied a CC-BY-4.0 public copyright license to any Author Accepted Manuscript version arising from this submission.

previously solved subproblems to efficiently construct a new path. The main contribution of this paper is a path planning algorithm that:

- exploits the hierarchical structure of indoor 3D scene graphs to constrain planning to only the relevant regions of the environment.
- decomposes the global planning problem into smaller, independent subproblems, enabling parallel solving, thereby reducing planning time.
- includes a resource-efficient replanning strategy that isolates and updates only the affected parts of the plan, thereby reducing replanning latency.

Moreover, S-Path is built as a ROS-based framework utilizing the Open Motion Planning Library (OMPL) framework [14] and provides a lightweight interface to exploit the semantics of 3D indoor scene graphs to enhance the efficiency of sampling-based planners.

II. Related Works

A. Semantics-aware Path Planning

Terrain semantics such as the classification of reliable terrain has been employed in path planning to guide UGVs [5] and UAVs [7, 6]. In other works, semantic maps generated using cameras, semantic segmentation, and object detection, have been utilized. Sun *et al.* [15] uses multi-level indoor semantic maps encoding obstacles, dynamic entities, and room properties to guide an RRT planner with a path smoother to avoid collision zones. Achat *et al.* [16] uses a semantic pointcloud to generate multi-layer semantized maps that guide an A* path planner. Other works, such as [17, 18], utilize semantic information about various objects in the environment to inform path planning for efficient object search. Dharmadhikari *et al.* [3] presented a semantics-aware path planner that explores unknown environments, reconstructs meshes, and inspects semantics of interest. Ryll *et al.* [7] demonstrated how semantic information improves the perception–estimation–planning loop for high-speed drone flight in urban settings.

While the use of semantics in path planning is increasingly common, most approaches rely on flat, low-level representations of the environment semantics, which do not scale effectively in large or complex environments.

B. 3D Scene Graphs for Path Planning

The structured representation of 3D scene graphs has been predominantly employed for task planning by using their hierarchical structure [11, 19, 13], and even incorporating natural language processing [12, 20, 21] by using Large Language Model (LLM)s to match keywords in human-intuitive commands to nodes in the scene graph. Other works have utilized scene graphs to reason [22], infer [21, 23], or learn [24] navigation actions and policies. However, their application to low-level path planning in geometric spaces remains highly underexplored. Recently, HOV-SG [25] processes natural language queries across three levels of abstraction (floor, room, and object) to exploit the hierarchical structure of a scene

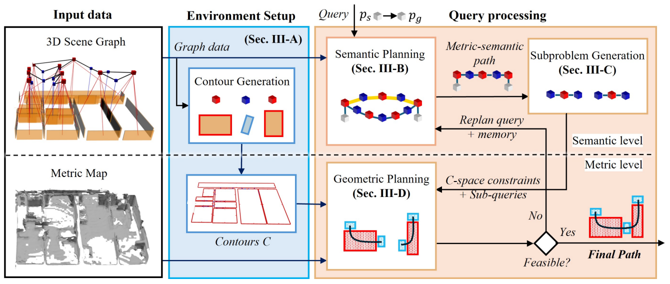


Fig. 2: S-Path architecture: A 3D scene graph and metric map are used as input to set up the environment, which consists of contours. A start-to-goal query is decomposed into subproblems that constrain the C-space for faster geometric planning. Solutions to these are joined to form the final path. If the path is infeasible, reusing results from subproblems solved during the initial attempt speeds up replanning.

graph and progressively narrow down the solution space. However, this is only used to identify the goal node for navigation; the path planning still relies on a Voronoi graph over the entirety of the free space, which does not scale well.

The approach closest to ours is that of Ray *et al.* [13], who proposed a task and motion planning framework over 3D scene graphs. Their method partitions the traversable space into *places* and employs a tri-level planner for hierarchical planning. However, the *places* are structured in a grid-like fashion and lack high-level semantic meaning. Moreover, their focus is primarily on plan feasibility, without exploiting the potential gains in efficiency or parallelism that S-Path offers.

III. Methodology

The architecture of S-Path is outlined in Figure 2. As *Input Data* (Section III-A), S-Path requires both a metric-semantic indoor 3D scene graph that includes information about rooms and doorways, and a metric mesh of the environment. Provided this input, a one-time *Environment Setup* (Section III-B) generates contours that associate nodes in the scene graph with corresponding regions in the metric mesh. Subsequently, for each path planning query, *Semantic Planning* (Section III-C) is performed on the scene graph to generate a semantic path, effectively restricting the problem to only the relevant areas of the environment. This problem is further decomposed through *Subproblem Generation* (Section III-D), which breaks the task into smaller, independent doorway-to-doorway subproblems. These subproblems are solved in parallel during *Geometric Planning* (Section III-E) using sampling-based planners. Finally, the solutions to the local subproblems are stitched together to get the resulting set of waypoints. In the event that the path is infeasible, *Replanning* (Section III-F) ensures an efficient reroute.

A. Input Data

S-Path requires both an indoor 3D metric-semantic scene graph to support semantic planning, and a metric mesh of the environment for geometric planning.

The 3D scene graph is an undirected topological graph that encodes both metric-semantic information and relational data from an indoor environment. S-Path expects the graph to contain at least the following types of nodes:

- *Walls*, where each wall $\mathbf{W}_i = \{\boldsymbol{\pi}_i \mid \boldsymbol{\pi}_i \in \mathbb{R}^4\}$ is defined by the plane equation $\boldsymbol{\pi}_i = [\mathbf{n}_i^\top, d_i]^\top$, with the normal $\mathbf{n}_i \in \mathbb{R}^3$ pointing inside the room and scalar offset d_i.
- *Rooms*, where each room $\mathbf{R}_i = \{\boldsymbol{\Psi}_i, \boldsymbol{\rho}_i \mid \boldsymbol{\rho}_i \in \mathbb{R}^3\}$ is defined by its centroid $\boldsymbol{\rho}_i$ and a set of walls bounding it, i.e., $\boldsymbol{\Psi}_i = \{\mathbf{W}_{i,1}, \mathbf{W}_{i,2}, \ldots, \mathbf{W}_{i,m}\}$ with $m \geq 3$.
- *Doorways*, where each doorway $\mathbf{D}_i = \{\boldsymbol{\delta}_i, w_i, s_i \mid \boldsymbol{\delta}_i \in \mathbb{R}^3, w_i \in \mathbb{R}, s_i \in \{0, 1\}\}$ is defined by its centroid $\boldsymbol{\delta}_i$, width w_i, and a binary state s_i indicating whether the doorway is traversable.

This graph can be generated from real-time LiDAR [10] or visual [9, 26] measurements or extracted from real or synthetic environment maps using Building Information Modeling (BIM) [27], or from public datasets [28].

The metric mesh is obtained from a voxel-based map, which is converted to a Euclidean Signed Distance Field (ESDF), using Voxblox [29]. This forms the foundation for path planning heuristics and collision checking.

The aforementioned input data is used once to set up the environment and then can be updated in response to changes. For path planning, a query with start and goal points, $\mathbf{p}_s$ and $\mathbf{p}_g$, is required. This query can also be provided in a human-interpretable form, i.e., from a start room $\mathbf{R}_s$ to an end room $\mathbf{R}_g$, in which case the centroids of the rooms are used: $\mathbf{p}_s = \boldsymbol{\rho}_s$ and $\mathbf{p}_g = \boldsymbol{\rho}_g$. Moreover, a planning time budget, denoted by *time-to-plan* (*ttp*), also needs to be specified.

B. Environment Setup

Semantic graph generation The scene graph is transformed into an undirected, weighted *semantic graph* that connects the semantic elements (rooms and doors) within the scene, which is used to compute a semantic path. Each connection between nodes is assigned a cost equal to the Euclidean distance between the centroids of the elements representing those nodes. Lastly, edges connected to untraversable doorways are assigned an infinite weight.

Contour generation The wall planes information in the scene graph is used to generate room and doorway *contours* $\mathbf{C} = \{\mathbf{C}_R, \mathbf{C}_D\}$. Contours serve two purposes: first, they restrict sampling-based geometric planners to areas that directly contribute to the final path; second, they enable the association of geometric locations with semantic elements, such as rooms and doorways, via a *point-in-polygon* test. As shown in Figure 3, room polygons are derived by projecting wall planes into 2D lines, computing all possible intersections, and generating line segments between them. These segments are then connected at their endpoints to form enclosed 2D polygons. Doorway polygons are constructed by casting a ray through the centroid of the doorway, perpendicular to the closest wall segment of the closest room. Two additional rays are cast parallel to this ray at a half-width offset on either side (refer to Figure 3). The four resulting intersection points with the walls of the connecting rooms are used to construct the doorway polygon. Both the room and doorway polygons are extruded along the *z*-axis to match the respective room height.

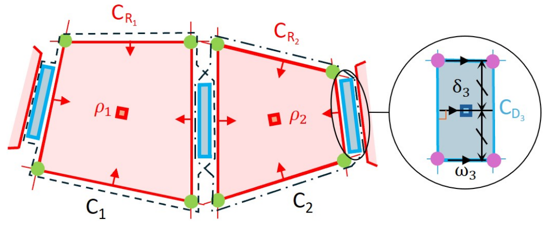


Fig. 3: Room contours $\mathbf{C}_{\mathbf{R}_i}$ are derived from the intersection points of wall lines. Doorway contours $\mathbf{C}_{\mathbf{D}_i}$ are fitted between two rooms using the doorway's center, its width, and the closest room segment as a reference. $\mathbf{C}_i$ contours are the union of those of rooms and doorways.

It is important to note that the quality of contour generation is dependent on the accuracy of the wall information associated with each room in the input scene graph. Moreover, the proposed algorithm does not work for concave-shaped rooms; however, such rooms are uncommon in indoor environments and, if present, can either be split into convex rooms connected by *artificial* doorways or be represented by a convex polygon (rectangle in the simplest case) that spans the area of the room.

C. Semantic Planning

Given the input query $\mathbf{p}_s \rightarrow \mathbf{p}_g$, *point-in-polygon* tests are performed over room contours to determine the rooms associated with the start and goal points, denoted as $\mathbf{R}_s$ and $\mathbf{R}_g$, respectively. The start and goal points may also lie within doorway contours; however, since the area of doorway contours is typically much smaller than that of room contours, and the overall process remains similar, we assume, without loss of generality, that these points lie within room contours. Once the relevant contours are identified, an A^* search is then executed on the semantic graph to compute the shortest semantic path from $\mathbf{R}_s$ to $\mathbf{R}_g$, as follows:

$$\boldsymbol{\Pi}_{\text{sem}} := \mathbf{R}_s \rightarrow \mathbf{D}_k \rightarrow \mathbf{R}_r \rightarrow \cdots \rightarrow \mathbf{D}_{k+n} \rightarrow \mathbf{R}_g \qquad (1)$$

where $\mathbf{D}_k$ and $\mathbf{D}_{k+n}$ is the first and last doorway in the semantic path with $n{+}1$ doorways and $n{+}2$ rooms. Furthermore, since the semantic graph is generally sparse, performing an A^* search on it incurs negligible computational cost compared to a full search on the metric layer. $\boldsymbol{\Pi}_{\text{sem}}$ is used to generate coarse waypoints for the robot to follow, in the form of the center points of the doorways along the path. These waypoints define the coarse geometric path, $\boldsymbol{\Pi}_{\text{geo}}$, given by:

$$\begin{aligned} &\boldsymbol{\Pi}_{\text{geo}} := \mathbf{p}_s \rightarrow \boldsymbol{\delta}_k \rightarrow \boldsymbol{\delta}_{k+1} \rightarrow \cdots \rightarrow \boldsymbol{\delta}_{k+n} \rightarrow \mathbf{p}_g, \\ &\textit{via: } \mathbf{C}' = \mathbf{C}_{\mathbf{R}_s} \cup \bigcup_{i=0}^{n-1} \mathbf{C}_{\mathbf{R}_{r+i}} \cup \bigcup_{i=0}^{n} \mathbf{C}_{\mathbf{D}_{k+i}} \cup \mathbf{C}_{\mathbf{R}_g} \end{aligned} \qquad (2)$$

where the reduced sample space $\mathbf{C}'$ is composed of the room and doorways contours within the semantic path and δ_k is the center point of $\mathbf{D}_k$. It is important to note that introducing intermediate waypoints forces the robot to pass exactly through the center of each doorway, which may adversely affect path optimality compared to solving the global planning problem directly from $\mathbf{p}_s \rightarrow \mathbf{p}_g$. However, in practical scenarios, doorways are often narrow, meaning their center points are typically close to the globally optimal crossing points, thus minimizing the impact of subproblem decomposition on the overall path length. Another consideration is that the semantic graph is constructed with edge weights corresponding to the Euclidean distance between room centers and the centers of the connecting doorways. If room sizes are highly uneven, for instance, a very large room that is only partially traversed, the resulting semantic path may be suboptimal. Nonetheless, these added costs are arguably outweighed by the planning-time benefits that S-Path achieves through decomposing the problem, as validated in Section IV.

D. Subproblem Generation

The coarse geometric path in Equation (2) can be decomposed into independent subproblems as follows:

$$\begin{aligned} 1 : \mathbf{p}_s &\rightarrow \boldsymbol{\delta}_k, && \textbf{via: } \mathbf{C}_1 \\ 2 : \boldsymbol{\delta}_k &\rightarrow \boldsymbol{\delta}_{k+1}, && \textbf{via: } \mathbf{C}_2 \\ &\cdots \\ i : \boldsymbol{\delta}_{k+i-2} &\rightarrow \boldsymbol{\delta}_{k+i-1}, && \textbf{via: } \mathbf{C}_i \\ &\cdots \\ n+2 : \boldsymbol{\delta}_{k+n} &\rightarrow \mathbf{p}_g, && \textbf{via: } \mathbf{C}_{n+2} \end{aligned} \tag{3}$$

where for the i^{th} subproblem, $\mathbf{C}_i$ refers to the composed contours, *i.e.,* the boolean union of the room and doorway contours required to solve the sub-problem. The subproblems vary in size; hence, time allocation during geometric planning must account for this to ensure a fair distribution of time. Therefore, for each subproblem, an effort heuristic is computed to estimate its relative complexity, by correlating with the size of the subproblem as follows:

$$e_i = \begin{cases} \|\boldsymbol{\delta}_k - \mathbf{p}_s\| + \sqrt{a_1} & i = 1 \\ \|\boldsymbol{\delta}_{k+i-2} - \boldsymbol{\delta}_{k+i-1}\| + \sqrt{a_i} & 2 \leq i \leq n+1 \\ \|\mathbf{p}_g - \boldsymbol{\delta}_{k+n}\| + \sqrt{a_{n+2}} & i = n+2 \end{cases} \tag{4}$$

where a_i is the area of the composed contour $\mathbf{C}_i$. Subproblems with higher heuristic values receive more computational resources and time. Moreover, to prevent excessively small time allocations, subproblems with heuristic values less than a predefined value, defined as a proportion of an equal split, are merged with one of their neighboring subproblems, specifically, the neighbor with the lower effort heuristic, among the at most two adjacent candidates. Following this, the effort heuristics in Equation (4) are recomputed. This ensures that planning does not fail due to avoidable time constraints and also relaxes the strict requirement of passing exactly through the center of doorways for the merged subproblems. This merging process is performed iteratively until no subproblem remains excessively small.

E. Geometric Planning

The geometric planning in S-Path employs a thread pool approach, distributing subproblems across m parallel workers executing sampling-based planners from the OMPL library [14]. For such planners, the quality of the path is directly related to the time spent sampling and checking points, as a

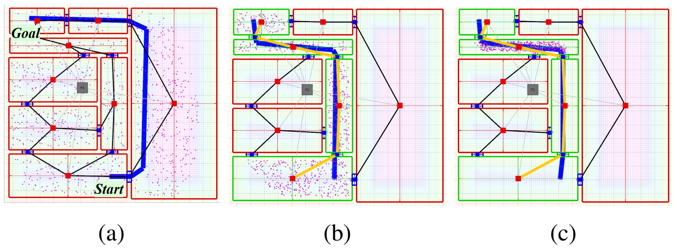


Fig. 4: Illustration of the features of S-Path : (a) The baseline creates many samples that do not contribute to the final path, thus finding a suboptimal solution given a limited time. (b) Restricting sample space to $\mathbf{C}'$ leads to denser coverage of only relevant regions (c) Subproblem decomposition divides results in smaller and sometimes trivial (notice the straight line path in the last two rooms) problems.

denser set of samples is more likely to yield shorter paths. For this reason, the ttp for each subproblem is distributed proportionally to the effort heuristics defined in Equation (4).

Planning for smaller subproblems avoids unnecessary collision checks and rewiring of vertices that do not contribute to the local solution, leading to significantly faster planning. This is evident in Figure 4, where, compared to planning over the global problem (see Figure 4a), restricting the C-space to $\mathbf{C}'$ results in denser coverage and a shorter path (see Figure 4b) Further subproblem decomposition can even yield point-to-point, obstacle-free subproblems, which are trivial to solve using an informed planner like BIT*. This is demonstrated in the empty regions of the first two and last rooms along the path in Figure 4c, where no sampling was required at all.

Once all subproblems have been solved, local solutions are stitched at the respective doorway centers to get the final path.

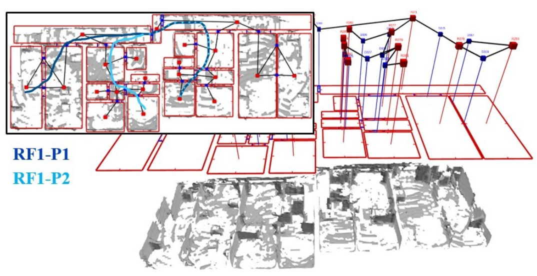
F. Replanning

Since the environment setup is performed only once per map, changes in the environment or mismatches between the semantic graph and the actual mesh traversability can render a previously generated path infeasible. Such cases may arise due to newly closed doorways or unforeseen obstructions. To address this, S-Path first updates the environment representation and reconstructs the semantic graph, assigning infinite edge weights to connections involving blocked nodes. Additionally, edges corresponding to already planned portions of the path are updated with reduced weights (using a multiplicative factor less than 1), thereby encouraging their reuse during subsequent replanning. Semantic planning is then re-executed, starting from the node on the original path closest to the detected blockage. Previously solved subproblems are retrieved from memory to prevent redundant computation, and only the updated portions of the path are replanned during the geometric planning, thereby reducing replanning time. Finally, the newly computed and cached path segments are concatenated to form the final path.

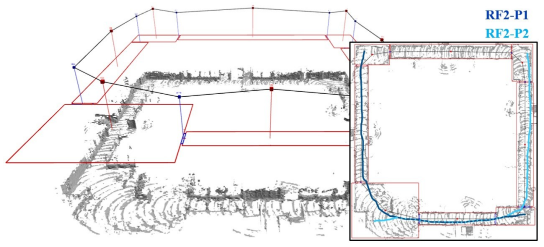
IV. Evaluation

A. Validation Methodology

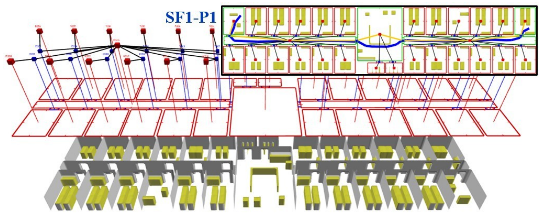
To the best of our knowledge, there is no prior work directly comparable to ours. The most closely related approach [13]



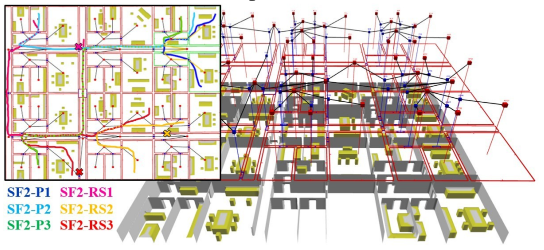
(a) RF1: Real construction site, augmented by the scene graph built from its BIM model.



(b) RF2: Campus building augmented by the scene graph built from its BIM model, featuring long, wide corridors with some obstacles.



(c) SF1: Synthetic office floor environment with several obstacles. The offices on either side of the floor are connected via a central corridor that leads to a shared space in the center.



(d) SF2: Synthetic environment representing a large apartment floor with several units and long, narrow corridors.

Fig. 5: Real and synthetic environments used for evaluations. The paths shown in the top-down orthognal view are the specific queries used from each environment in the sequential execution evaluation. -P* represents a query for planning scenarios and -RS* for replanning scenarios.

focuses on plan feasibility and does not address efficiency. Moreover, it is not open-source. Therefore, we evaluate performance improvements of S-Path relative to the underlying sampling-based planners. Three planners were selected: 1) BIT* [1] as a tree-style planner with built-in heuristics. 2) PRM* [30] as an unbiased coverage-of-space planner; and 3) RRT* [31] as a tree-style planner with no built-in heuristics. The validation is divided into two parts: sequential and parallel execution. The sequential evaluation ablates the effects of the individual components of S-Path, while the parallel evaluation showcases its scalability.

Environments Two real and two synthetic environments are

TABLE I: The characteristics of the real (RF*) and synthetic (SF*) environments and used for evaluations.

Environment	#obstacles	#walls	#doorways	#rooms	Area (m^2)
RF1	-	88	23	22	371
RF2	-	32	8	8	2901
SF1	116	116	28	29	493
SF2	139	236	60	59	1740

used for evaluations, as described in Table I and shown in Figure 5. For real environments (Figures 5a and 5b), LiDAR scans are used to generate the scene graphs using [27] and manually contructed BIM models, and the metric mesh following [29]. Moreover, concave rooms are represented by a spanning rectangular contour as described in Section III-B (see bottom-left of environment RF2). Furthermore, since the real environments are limited in size and complexity, we employ synthetically generated environments (Figures 5c and 5d) that cover relatively larger areas with multiple bounded obstacles. In this case, the scene graph is derived from the positions of rooms and doorways in the synthetic environment, while the voxels representing obstacles and walls are used to generate an occupancy grid that is converted to an ESDF, using [29].

Sequential execution methodology A version of S-Path that solves all subproblems in sequence, referred to as S-Path(s), is compared against the aforementioned planners operating on the global problem, which is denoted by **I** (refer to Figure 4a). In addition, other ablations are also evaluated, specifically, **II** (refer to Figure 4b), which extends **I** by adding C-space restrictions following Equation (2), and **III** (refer to Figure 4c), which extends **II** by adding subproblem decomposition following Equation (3), but without replanning memory.

Performance is evaluated on 11 queries divided into two scenarios: planning and replanning. For the planning scenarios, four queries are used from the synthetic environments and four from the real environments. These queries differ in path length and complexity as can be seen in Figure 5. ttp_{min} is set to $1\,\mathrm{ms}$ while $ttp_{\max}$ is set to $6\,\mathrm{s}$. Moreover, in these scenarios, the ablation **III** and S-Path(s) are equivalent, as there is no replanning involved, so **III** is omitted from the results. For the replanning scenarios, only the large and complex SF2 environment is used, as it offers multiple routes between most node pairs. This enables the intentional blocking of key doorways to create challenging evaluation cases. In these scenarios, when the planner receives an update to the scene graph that renders the current path infeasible due to an untraversable doorway, it updates the start position to the closest doorway along the path preceding the obstruction. All ablations are then re-executed for the modified query, incorporating the necessary updates to the semantic graph and mesh owing to the blockage. Additionally, S-Path(s) retains memory of the previously computed plan to inform subsequent replanning. The replanning scenarios are generally more difficult than planning scenarios, as heuristics for sampling-based solvers are based on Euclidean distance, which might fail in the presence of a closed doorway because the shortest path could require moving in the opposite direction of the goal initially. To accommodate this, $ttp_{\min}$ is set to $25\,\mathrm{ms}$ and $ttp_{\max}$

is increased to 10 s. Overall, three replanning scenarios are evaluated as visualized in Figure 5d.

All ablations $\mathcal{A} = \{\mathbf{I}, \mathbf{II}, \mathbf{III}, \text{S-Path(s)}\}$ are evaluated with respect to their computational efficiency (CPU time) and path length optimality. To this end, an overall efficiency metric $\bar{\eta}$ is introduced, which considers both factors and also accounts for robustness by repeating each computation 100 times to obtain results at a 95 % success rate. In this regard, computational efficiency $\eta_{ttp,i}$ measures the improvement in time-to-plan (ttp) relative to ablation **I**:

$$\eta_{ttp,i} = \frac{ttp_{95\%,\mathbf{I}}}{ttp_{95\%,i}}, \quad \forall\, i \in \mathcal{A} \tag{5}$$

where a higher $\eta_{ttp,i}$ indicates better computational performance relative to the underlying sampling-based planner. The $ttp_{95\%}$ is computed by first defining bounds, i.e., $ttp \in [ttp_{\min}, ttp_{\max}]$, and then sampling values logarithmically within this range to obtain a series of candidate ttp values. For each sampled value, the success rate is calculated as the percentage of runs that result in a feasible path. Finally, $ttp_{95\%}$ is estimated using piecewise linear interpolation over the success rates to determine the minimum ttp at which at least 95% of the runs are successful.

Similarly, the path efficiency $\eta_{l,i}$ is defined as the ratio of the shortest path length $l_{conv,i}$ to the path length at the 95 % success mark $l_{95\%,i}$, determined similarly to $ttp_{95\%}$ by running 100 trials for each ablation:

$$\eta_{l,i} = \frac{l_{conv,i}}{l_{95\%,i}}, \quad \forall\, i \in \mathcal{A} \tag{6}$$

which favors consistent path lengths and is used as a measure of robustness. Finally, the combined efficiency η_i of a planning method is given by:

$$\eta_i = \eta_{l,i}\, \eta_{ttp,i}, \quad \forall\, i \in \mathcal{A} \tag{7}$$

leading to the normalized efficiency gain $\bar{\eta}_i$ as:

$$\bar{\eta}_i = \frac{\eta_i}{\eta_{\mathbf{I}}}, \quad \forall\, i \in \mathcal{A} \tag{8}$$

which is considered as the final unified metric for comparisons.

Parallel execution methodology This aspect is evaluated by enabling multithreading in the sequential S-Path(s), resulting in the proposed S-Path. A total of 77 random queries are evaluated, distributed across all environments, with each query consisting of between 2 and 10 subproblems. The evaluation metric is the *speedup* factor, defined as the ratio of the time required to solve the subproblems sequentially (using S-Path(s)) versus in parallel (using S-Path). For these experiments, ttp is set to a constant value of 1000 ms, and PRM* is used as the underlying sampling-based planner.

Hardware All experiments were conducted on a laptop equipped with a 12th Gen Intel(R) Core(TM) i9-12900H processor that has 14 cores ($m = 20$ threads).

B. Results and Discussions

Sequential execution results Table II summarizes the efficiency gains of the evaluated methods across all queries. Specifically, in the planning scenarios, S-Path(s) demonstrates

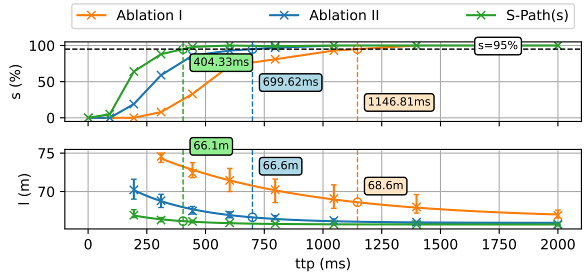


Fig. 6: Success rate s and path length l with respect to ttp values across different methods on the SF1-P1 scenario with BIT*. Annotated values indicate $ttp_{95\%}$ and $l_{95\%}$, respectively.

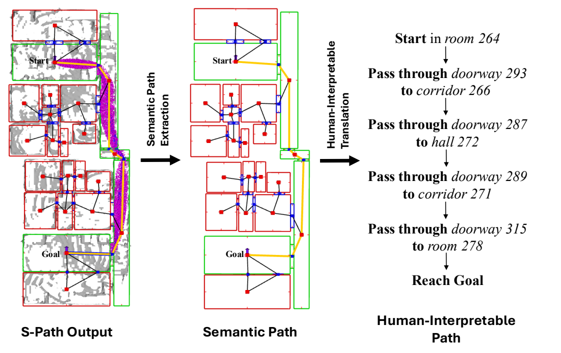
multifold gains in efficiency, with an average increase of 5.7×. This is even without considering its pairing with RRT*, where in many cases S-Path(s) converges while other ablations do not. The efficiency gains from pairing with PRM* are greater (8.1× on average) than those from pairing with BIT* (4.3× on average), because the uninformed PRM* benefits more from the C-space restrictions provided by S-Path(s). Moreover, while S-Path(s) on average produces slightly longer paths (difference in $l_{95\%}$ between −5.4% and +15.9%, with an average of +1.9%), due to its constraint of passing through the center of doorways; this is compensated for by substantial improvements in the *time-to-plan* (reduction in $ttp_{95\%}$ between 1.1× and 27.2×, with an average of 5.7×).

In replanning scenarios, S-Path consistently produces shorter paths, averaging 14.9% shorter than those generated by **I**. Its heuristics prove especially beneficial in environments with blockages, where increased complexity frequently causes traditional heuristic-guided planners to fail. Furthermore, the ability to cache and reuse previously solved subproblems eliminates the need for redundant computations, further reducing ttp and improving overall efficiency, by up to 52× for BIT* and 22× for PRM*. A notable example is the SF2-RS2 with BIT*, where the replanning task becomes nearly trivial, as most subproblems are either cached from the initial planning attempt or are reduced to simple point-to-point queries without obstacles, that are trivial for BIT*.

Furthermore, each individual component of S-Path contributes incrementally to the overall efficiency gains (see Figure 6 for an example run). The C-space restriction introduced in **II** provides the most significant improvement, as it limits sampling to only relevant regions, allowing planners to make more progress within the same computational time. Further efficiency gains are achieved through subproblem decomposition in S-Path (or **III** in the *Replanning Scenarios*), although these gains are still limited by the sequential execution setup (the results with parallel execution that follow demonstrate additional speedups). An exception was observed in the RRT* experiments on the RF2 environment, where the C-space restriction introduced in **II** leads to performance regressions, likely due to interference with RRT*'s random sampling strategy. Despite this, S-Path(s) manages to converge by decomposing the global planning problem into smaller subproblems, thereby mitigating the negative impact of the C-space restriction. Finally, the reuse of previously solved subproblems in the *Replanning Scenarios* leads to further improvements over **III**, particularly

TABLE II: Efficiency gains of the tested ablations across all queries using *PRM**, *BIT**, and *RRT** planners. The best and second-best efficiency results for each configuration are boldfaced and underlined respectively. While S-Path(s) shows improvement in the *time-to-plan*, *ttp*, for all configurations, path optimality varies. Therefore, path length results by S-Path(s) are highlighted in green or in red if they are more than 2% shorter or longer compared to **I**, respectively; and in yellow otherwise.

			BIT*				PRM*				RRT*			
	Scenario	**Ablation**	$ttp_{95\%}[ms]$	$l_{95\%}$ $[m]$	l_{conv} $[m]$	$\bar{\eta}$	$ttp_{95\%}[ms]$	$l_{95\%}$ $[m]$	l_{conv} $[m]$	$\bar{\eta}$	$ttp_{95\%}[ms]$	$l_{95\%}$ $[m]$	l_{conv} $[m]$	$\bar{\eta}$
Planning Scenarios	**RF1-P1**	**I**	272.97	32.65	32.02	1.00	1537.47	32.51	32.10	1.00	> 6000.00	-	-	-
		II	140.19	32.98	31.83	1.92	661.93	32.41	31.83	2.39	> 6000.00	-	-	-
		S-Path(s)	104.77	34.74	34.28	**2.62**	445.35	34.52	34.28	**3.47**	1238.62	33.96	33.80	-
	RF1-P2	**I**	157.24	28.17	25.60	1.00	675.94	28.21	26.70	1.00	> 6000.00	-	-	-
		II	86.40	29.10	26.96	1.86	279.80	28.98	27.31	2.41	3347.76	26.51	26.03	-
		S-Path(s)	34.23	31.55	30.91	**4.95**	68.85	31.96	31.01	**10.06**	742.91	30.80	30.66	-
	RF2-P1	**I**	87.70	88.96	87.97	1.00	162.86	89.65	88.53	1.00	188.01	88.31	88.71	1.00
		II	68.37	89.98	89.11	1.28	159.15	89.93	89.25	1.03	309.92	88.77	88.51	0.61
		S-Path(s)	21.93	89.96	89.41	**4.02**	27.16	90.81	89.60	**5.99**	169.48	90.04	89.35	**1.11**
	RF2-P2	**I**	130.57	88.40	87.52	1.00	226.39	90.79	88.53	1.00	188.01	88.31	87.71	1.00
		II	71.11	88.41	87.73	1.84	176.34	88.61	87.44	1.31	482.60	86.74	86.60	0.64
		S-Path(s)	13.31	88.08	87.86	**9.88**	26.71	88.72	87.77	**8.65**	150.65	88.57	87.94	**2.05**
	SF1-P1	**I**	377.03	52.39	51.79	1.00	974.88	52.54	52.05	1.00	> 6000.00	-	-	-
		II	197.55	52.46	51.56	1.90	497.24	52.54	51.70	1.95	> 6000.00	-	-	-
		S-Path(s)	120.99	52.30	51.72	**3.12**	355.26	52.27	51.70	**2.74**	480.08	51.86	51.38	-
	SF2-P1	**I**	380.36	30.06	29.01	1.00	3036.94	31.71	30.97	1.00	> 6000.00	-	-	-
		II	122.57	31.07	30.24	3.13	220.35	31.58	30.46	13.61	> 6000.00	-	-	-
		S-Path(s)	98.27	29.95	29.65	**3.97**	111.70	32.95	30.06	**25.40**	284.09	29.88	28.87	-
	SF2-P2	**I**	830.98	68.27	65.31	1.00	3076.78	68.86	67.15	1.00	> 6000.00	-	-	-
		II	494.74	66.38	64.67	1.71	680.71	66.85	64.96	4.09	> 6000.00	-	-	-
		S-Path(s)	284.09	65.79	65.26	**3.03**	309.19	66.34	65.31	**4.56**	1310.23	65.38	64.95	-
	SF2-P3	**I**	1146.81	68.61	66.89	1.00	4196.83	70.47	68.57	1.00	> 6000.00	-	-	-
		II	699.62	66.64	65.77	1.66	1525.88	67.09	66.09	2.79	> 6000.00	-	-	-
		S-Path(s)	404.33	66.15	65.66	**2.89**	1163.57	66.67	66.18	**3.68**	2045.37	65.56	65.34	-
Re-planning Scenarios	**SF2-RS1**	**I**	3631.95	69.53	66.85	1.00	9624.65	78.88	74.41	1.00	> 10000.00	-	-	-
		II	800.33	66.84	66.14	4.67	2830.32	66.69	66.26	3.58	> 10000.00	-	-	-
		III	467.46	66.53	66.21	8.04	1088.09	67.04	66.23	9.26	2830.32	66.57	66.57	-
		S-Path(s)	458.61	66.49	66.27	**8.21**	672.77	67.18	66.21	**14.95**	2731.59	66.06	65.86	-
	SF2-RS2	**I**	1329.12	33.54	33.31	1.00	3897.78	34.41	33.85	1.00	> 10000.00	-	-	-
		II	457.66	33.64	33.25	2.89	736.55	34.39	33.27	5.20	9699.72	33.48	33.36	-
		III	367.32	33.61	33.39	3.64	410.11	34.06	33.32	9.45	864.12	33.26	33.11	-
		S-Path(s)	25.00	34.51	34.03	**52.79**	318.42	34.28	34.01	**12.35**	462.41	34.06	33.92	-
	SF2-RS3	**I**	1820.57	67.38	46.22	1.00	9061.63	79.60	55.38	1.00	> 10000.00	-	-	-
		II	800.33	46.78	46.00	3.26	2896.14	46.75	46.23	4.47	> 10000.00	-	-	-
		III	458.61	46.46	46.08	5.74	800.33	47.15	46.10	15.91	4929.15	45.76	45.66	-
		S-Path(s)	455.50	46.40	46.20	**5.80**	566.47	46.71	46.22	**22.75**	2160.89	46.35	46.22	-



Start in *room 264*
Pass through *doorway 293* **to** *corridor 266*
Pass through *doorway 287* **to** *hall 272*
Pass through *doorway 289* **to** *corridor 271*
Pass through *doorway 315* **to** *room 278*
Reach Goal

Fig. 7: Example of human-interpretable path generation from S-Path output. First, the semantic path is extracted, which is then translated into human-interpretable commands.

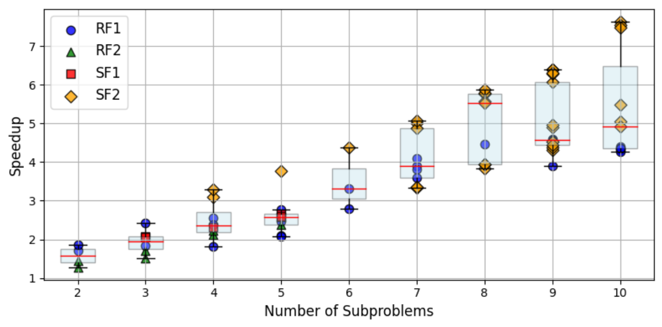


Fig. 8: Parallel execution speedups relative to sequential execution as a function of the number of identified subproblems, evaluated across queries in different environments. Each marker represents a query in the respective environment. On average, speedup is proportional to the number of subproblems, with queries in synthetic environments exhibiting higher speedups due to the absence of irregular obstacles.

for RRT*, which otherwise incurs high computational costs due to its random sampling.

Overall, S-Path substantially benefits underlying planners that lack built-in heuristics, such as RRT*, while also enhancing the performance of informed planners like BIT*. Additionally, S-Path produces interpretable plans by utilizing semantic concepts such as rooms and doorways (see Figure 7).

Parallel execution results S-Path yields parallel execution speedups that correlate with the number of subproblems, as shown in Figure 8, though the actual gains depend heavily on the subproblem setup, since the subproblems vary in complexity. When sufficient cores are available to process all subproblems concurrently, the total parallel execution time approaches the *ttp* of the largest subproblem. This results in greater variation in speedups for queries with more subproblems, as emphasized by the taller boxplots in Figure 8. Conversely, when there are fewer cores than subproblems, the speedup gains diminish accordingly. However, in all queries, including those in large environments such as SF2 (see area in Table I), the number of subproblems was at most 10, so the number of available cores was not a limiting factor. This is largely due to subproblem merging, which avoids the creation of very small, practically inefficient subproblems (as discussed in Section III-D), making S-Path highly scalable. Furthermore, it is important to note that the semantic planning for all queries took less than $1\,\mathrm{ms}$, making its overhead negligible.

V. Conclusion

In this work, we proposed S-Path, a novel metric-semantic path planning algorithm that augments sampling-based planners with semantic information derived from indoor 3D scene graphs, which improves planning efficiency while providing human-interpretable path planning. Experimental results demonstrate that S-Path achieves up to $25\times$ efficiency improvement over the underlying sampling-based planner. This originates from the decomposition of the global planning problem into multiple independent subproblems by exploiting room and doorway semantics, effectively reducing the configuration space. This decomposition also enables parallel execution, allowing for further speedup which correlate with the number of subproblems. In scenarios where the original path becomes infeasible due to blockages, S-Path can efficiently replan by reusing previously solved subproblems, achieving up to $52\times$ overall efficiency improvement. However, S-Path enforces traversal through the center of doorways, which may lead to suboptimal paths when considering wide doorways. Additionally, S-Path currently assumes rooms delimited by planar walls, which might arise challenges in contour generation and sample space restriction in environments with complex curved architectural geometries. Future work will focus on refining path optimality through a local-to-global path optimization step and on extending contour generation to handle more complex architectural structures for improved generalizability.

References

[1] J. D. Gammell, S. S. Srinivasa, and T. D. Barfoot, “Batch Informed Trees (BIT*): Sampling-based optimal planning via the heuristically guided search of implicit random geometric graphs,” in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, 2015.

[2] D. Kirilenko, A. Andreychuk, A. Panov, and K. Yakovlev, “Transpath: Learning heuristics for grid-based pathfinding via transformers,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2023.

[3] M. Dharmadhikari and K. Alexis, “Semantics-aware exploration and inspection path planning,” in *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2023.

[4] J. A. Cobano, S. Martinez, L. Merino, and F. Caballero, “3d semantic heuristic planning for safer aerial robot navigation indoors,” in *2024 IEEE 29th International Conference on Emerging Technologies and Factory Automation (ETFA)*. IEEE, 2024.

[5] Y. Zhao, P. Liu, W. Xue, R. Miao, Z. Gong, and R. Ying, “Semantic probabilistic traversable map generation for robot path planning,” in *IEEE international conference on robotics and biomimetics*, 2019.

[6] L. Bartolomei, L. Teixeira, and M. Chli, “Perception-aware path planning for uavs using semantic segmentation,” in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020.

[7] M. Ryll, J. Ware, J. Carter, and N. Roy, “Semantic trajectory planning for long-distant unmanned aerial vehicle navigation in urban environments,” in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2020.

[8] 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)*. IEEE, oct 2019.

[9] N. Hughes, Y. Chang, and L. Carlone, “Hydra: a real-time spatial perception engine for 3D scene graph construction and optimization,” *Robotics: Science and Systems (RSS)*, 2022.

[10] H. Bavle, J. L. Sanchez-Lopez, M. Shaheer, J. Civera, and H. Voos, “S-graphs+: Real-time localization and mapping leveraging hierarchical representations,” *IEEE Robotics and Automation Letters*, 2023.

[11] C. Agia, K. M. Jatavallabhula, M. Khodeir, O. Miksik, V. Vineet, M. Mukadam, L. Paull, and F. Shkurti, “Taskography: Evaluating robot task planning over large 3d scene graphs,” in *Conference on Robot Learning*. PMLR, 2022.

[12] K. Rana, J. Haviland, S. Garg, J. Abou-Chakra, I. Reid, and N. Suenderhauf, “Sayplan: Grounding large language models using 3d scene graphs for scalable task planning,” in *Conference on Robot Learning*, 2023.

[13] A. Ray, C. Bradley, L. Carlone, and N. Roy, “Task and motion planning in hierarchical 3D scene graphs,” *Intl. Symp. of Robotics Research (ISRR)*, 2024.

[14] I. A. Sucan, M. Moll, and L. E. Kavraki, “The Open Motion Planning Library,” *IEEE Robotics & Automation Magazine*, 2012.

[15] N. Sun, E. Yang, J. Corney, and Y. Chen, “Semantic path planning for indoor navigation and household tasks,” in *Annual Conference towards Autonomous Robotic Systems*. Springer, 2019.

[16] S. Achat, J. Marzat, and J. Moras, “Path Planning Incorporating Semantic Information for Autonomous Robot Navigation,” in *International Conference on Informatics in Control, Automation and Robotics*, 2022.

[17] D. S. Chaplot, D. Gandhi, A. Gupta, and R. Salakhutdinov, “Object goal navigation using goal-oriented semantic exploration,” *Advances in Neural Information Processing Systems*, 2020.

[18] Y. Qiu, A. Pal, and H. I. Christensen, “Learning hierarchical relationships for object-goal navigation,” *Conference on Robot Learning*, 2020.

[19] Z. Ni, X. Deng, C. Tai, X. Zhu, Q. Xie, W. Huang, X. Wu, and L. Zeng, “Grid: Scene-graph-based instruction-driven robotic task planning,” in *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2024.

[20] Z. Dai, A. Asgharivaskasi, T. Duong, S. Lin, M.-E. Tzes, G. Pappas, and N. Atanasov, “Optimal scene graph planning with large language model guidance,” in *2024 IEEE International Conference on Robotics and Automation (ICRA)*, 2024.

[21] Q. Gu, A. Kuwajerwala, S. Morin, K. M. Jatavallabhula, B. Sen, A. Agarwal, C. Rivera, W. Paul, K. Ellis, R. Chellappa *et al.*, “Conceptgraphs: Open-vocabulary 3d scene graphs for perception and planning,” in *2024 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2024, pp. 5021–5028.

[22] S. Amiri, K. Chandan, and S. Zhang, “Reasoning with scene graphs for robot planning under partial observability,” *IEEE Robotics and Automation Letters*, 2022.

[23] A. Rajvanshi, K. Sikka, X. Lin, B. Lee, H.-P. Chiu, and A. Velasquez, “Saynav: Grounding large language models for dynamic planning to navigation in new environments,” in *Proceedings of the International Conference on Automated Planning and Scheduling*, vol. 34, 2024.

[24] Z. Ravichandran, L. Peng, N. Hughes, J. D. Griffith, and L. Carlone, “Hierarchical representations and explicit memory: Learning effective navigation policies on 3d scene graphs using graph neural networks,” in *International Conference on Robotics and Automation*. IEEE, 2022.

[25] A. Werby, C. Huang, M. Büchner, A. Valada, and W. Burgard, “Hierarchical Open-Vocabulary 3D Scene Graphs for Language-Grounded Robot Navigation,” in *Robotics: Science and Systems*, 2024.

[26] A. Tourani, S. Ejaz, H. Bavle, D. Morilla-Cabello, J. L. Sanchez-Lopez, and H. Voos, “vs-graphs: Integrating visual slam and situational graphs through multi-level scene understanding,” *arXiv preprint arXiv:2503.01783*, 2025.

[27] M. Shaheer, J. A. Millan-Romera, H. Bavle, J. L. Sanchez-Lopez, J. Civera, and H. Voos, “Graph-based global robot localization informing situational graphs with architectural graphs,” in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2023.

[28] C. Van Engelenburg, F. Mostafavi, E. Kuhn, Y. Jeon, M. Franzen, M. Standfest, J. van Gemert, and S. Khademi, “Msd: A benchmark dataset for floor plan generation of building complexes,” in *European Conference on Computer Vision*. Springer, 2024.

[29] H. Oleynikova, Z. Taylor, M. Fehr, R. Siegwart, and J. Nieto, “Voxblox: Incremental 3D Euclidean Signed Distance Fields for on-board MAV planning,” in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017.

[30] L. E. Kavraki, P. Svestka, J. C. Latombe, and M. H. Overmars, “Probabilistic roadmaps for path planning in high-dimensional configuration spaces,” *IEEE transactions on Robotics and Automation*, 1996.

[31] S. Karaman and E. Frazzoli, “Incremental Sampling-based Algorithms for Optimal Motion Planning,” in *Robotics: Science and Systems VI*. Robotics: Science and Systems Foundation, 2010.