T-FunS3D: Task-Driven Hierarchical Open-Vocabulary 3D Functionality Segmentation via Vision-Language Models

Jingkun Feng and Reza Sabzevari

***Abstract*— Open-vocabulary 3D functionality segmentation enables robots to localize functional object components in 3D scenes. It is a challenging task that requires spatial understanding and task interpretation. Current open-vocabulary 3D segmentation methods primarily focus on object-level recognition, while scene-wide part segmentation methods attempt to segment the entire scene exhaustively, making them highly resource-intensive. However, such significant computational and storage capacities are typically not accessible on the majority of mobile robots. To address this challenge, we introduce T-FunS3D, a task-driven hierarchical open-vocabulary 3D functionality segmentation method that provides actionable perception for robotic applications. Given a task description, T-FunS3D identifies the most relevant instances in an open-vocabulary scene graph and extracts their functional components. Experiments on SceneFun3D demonstrate that T-FunS3D outperforms baseline methods in open-vocabulary 3D functionality segmentation, while achieving faster runtime and reduced memory usage.**

I. INTRODUCTION

Open-vocabulary (OV) 3D functionality segmentation aims to extract the semantic meaning of functional object components in 3D scenes without relying on a fixed set of predefined annotations. This capability provides robots with both concrete objects and their functional parts, enabling them to execute the assigned tasks. Conventional 3D segmentation methods are primarily trained on labeled datasets and thus struggle to generalize to novel object classes. In contrast, open-vocabulary methods leverage large language models to infer scene semantics, supporting 3D exploration guided by free-form text descriptions.

While most existing methods focus on segmenting object-level instances, relatively few address finer-grained segmentation, such as functional object parts within scenes. However, such fine-grained entities are crucial for downstream tasks like scene interaction. For example, assistive robots must recognize not only objects in an open-set world but also their components, such as the chair arm and door of a washing machine, in order to execute general interactive tasks. Existing methods for this problem often perform fine-grained segmentation across nearly the entire scene without task-specific distinctions [1], [2], which imposes high computational overhead and memory costs. Therefore, there is a pressing need for more efficient designs that extract fine-grained segmentation in a task-dedicated manner.

To tackle this challenge, we propose a novel pipeline for hierarchical scene understanding and functionality segmentation based on task descriptions, using an open-vocabulary scene graph. Previous hierarchical methods typically extract semantics from image crops of every instance in the scene to obtain 3D part segmentation [1]. However, in practice, only a small subset of objects in the scene is relevant to the assigned task, making scene-wide over-segmentation an unnecessary burden on computation and storage. In contrast, our method identifies task-relevant areas and performs functional object part segmentation exclusively for selected objects. This design enables efficient online segmentation without demanding high computational power or large memory, making it well-suited for real-world robotic applications.

In summary, our main contributions are as follows:

- We introduce a training-free method for OV 3D functionality segmentation that leverages an actionable 3D scene graph containing OV semantics.
- We present a task-driven approach that hierarchically segments scenes from high-level instances into low-level functional parts, thereby facilitating efficient deployment in real-world robotic applications.
- Our method outperforms the baseline methods on the SceneFun3D [3] dataset in OV 3D segmentation of functional object components with faster runtime and lower memory consumption.

II. RELATED WORK

A. Open-Vocabulary 3D Instance and Part Segmentation

Recent works [9], [10], [11], [12] explore OV 3D scene understanding, but few provide a unified solution that supports both fine-grained segmentation and explicit hierarchical representations. Object-centric approaches [1], [10] achieve object-level segmentation but lack fine granularity, while point-level methods [11], [12] capture fine detail but lack hierarchical structure. [2] introduces a versatile mechanism, although it does not incorporate semantic information.

3D part segmentation has made some progress in OV-based frameworks thanks to recent efforts in [13], [14], [1]. A common strategy, as in [14]), is to lift 2D features from vision foundation models into 3D. However, these methods remain limited to single-object point clouds, restricting their applicability to full-scene understanding.

Conceptually, our work is most closely related to Search3D [1] and Fun3DU [15]. Whereas Search3D exhaustively embeds all objects and their parts, our method selectively segments objects and retrieves their corresponding functional component according to task requirements. Fun3DU, in turn, detects objects of interest across all images and lifts 2D segmentation masks of their functional parts into

*The authors are with the p4MARS Lab, Faculty of Aerospace Engineering, Delft University of Technology, The Netherlands
{jingkun.feng, r.sabzevari}@tudelft.nl

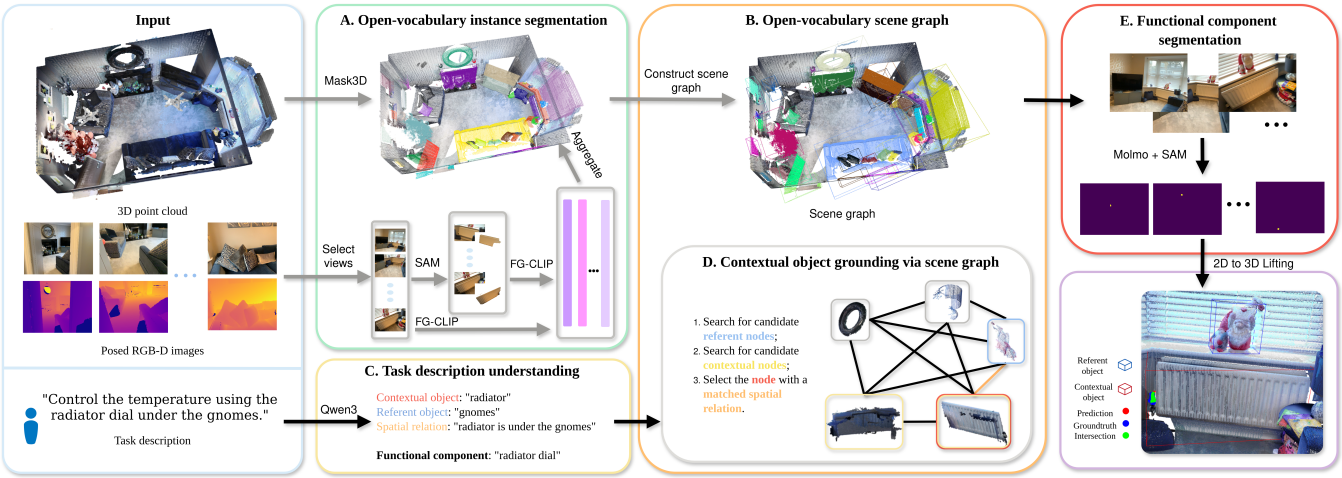

Fig. 1. T-FunS3D overview: The input of T-FunS3D are posed RBG-images and 3D point clouds of an indoor scene. (A) performs open-vocabulary instance segmentation by associating FG-CLIP [4] visual embeddings to class-agnostic instance segmentation from Mask3D [5]. We construct a scene graph of the featurized instances (B). Once a task is assigned, we decompose the description into ontologies using Qwen3 [6] (C) and identify the contextual object in the scene graph (D) by computing the text-visual embedding similarity. Lastly, (E) aggregates 2D masks extracted by combining Molmo [7] with SAM [8] to obtain 3D segmentation of functional parts.

3D. However, upon receiving a new query, Fun3DU needs to re-run the detection process for the object specified in the query, which is not efficient. In contrast, our approach constructs an actionable OV 3D scene graph for the entire scene, eliminating repeated instance detection and enabling efficient handling of complex 3D grounding tasks.

B. 3D Scene Graph

3D scene graphs encode objects and spatial concepts (e.g., rooms and floors) as nodes and their relations as edges, offering compact, object-centric representations well suited for robotics [16]. This decomposition facilitates higher-level reasoning and planning in object-centric tasks. Early works like [17] integrate SLAM with annotated data but remain limited to closed sets of objects. ConceptGraphs [18] advanced this line by combining 3D scene graphs with OV vision-language features, enabling queries in natural language. HOV-SG [19] extends this direction with hierarchical OV graphs, though they focus primarily on coarse semantics. Nevertheless, their graph encodes only parent-child relations(e.g., floor-rooms, room-objects), which is sufficient for navigation but inadequate for interactive tasks. Instead, T-FunS3D constructs an OV scene graph with inter-object relations to enable referential object grounding.

III. Method

This work proposes an efficient approach for segmenting OV functionality in robotic applications, utilizing RGB-D observations, ground-truth camera poses, and the 3D point cloud of an indoor environment. To this end, we introduce T-FunS3D, a novel method for task-driven hierarchical OV functionality segmentation in 3D scenes. The overall pipeline, illustrated in Fig. 1, consists of four modules organized into two stages. In the first stage, we perform 3D instance-level OV segmentation by extracting semantic embeddings for class-agnostic 3D instance segmentation (III-A). Then, an open-vocabulary scene graph is constructed based on the featurized 3D instance segments (III-B). In the second stage, the description of an assigned task is parsed using a large language model (LLM) to extract the task ontology (III-C). Afterwards, we identify the relevant contextual instances in the scene graph (III-D) and segment their functional parts required for task execution (III-E).

A. Open-vocabulary 3D instance segmentation

For OV 3D instance segmentation, we adapt the approach introduced in OpenMask3D [9]. To achieve this, we first apply Mask3D [5] to generate object-level mask proposals and decompose the scene. Unlike OpenMask3D, we compute the visual embeddings of the object with more pixel information to associate richer semantics with each proposal. For each proposed instance P_n, where $n \in [1, N]$, we crop the top $k \in [1, K]$ views with the highest visibility of the instance at multiple scales. The l-th ($l \in [1, L]$) scale crop of the k-th view for the n-th instance is denoted as $I^n_{k,l}$. K is set to 5 and L to 3 in our experiments. Next, we computed the OV semantics for the instances from both the full images and the crops of selected views using FG-CLIP [4], a multimodal vision-language model. However, the target object in the cropped images may be occluded or visible only through transparent materials (e.g., windows), introducing a foreground-background bias that reduces the accuracy of visual embeddings. To mitigate this, we additionally generate masked crops, denoted as $M^n_{k,l}$, where irrelevant pixels are masked out. These masked crops are used as complementary cues for the visual encoder. This combination retains contextual information while emphasizing the target instance. The final visual embedding for P_n is obtained by averaging across views and scales:

$$f(P_n) = \frac{1}{2K} \sum_{k}^{K} \left[f(I^n_k) + \frac{1}{2L} \sum_{l}^{L} f(I^n_{k,l}) + f(M^n_{k,l}) \right] \quad (1)$$

where $f(\cdot)$ denotes the visual embedding computation.

B. Open-vocabulary scene graph

A scene graph is defined as $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \{v_i\}$ for $i \in [1, N]$ denotes the set of vertices and $\mathcal{E} = \{e_{ij} | e_{ij} = (v_i, v_j),\ \text{for } v_i, v_j \in \mathcal{V} \text{ where } i \neq j\}$ the set of edges. In an OV scene graph, we store visual embeddings rather than explicit class labels in the nodes. To establish edges between two nodes, we average the visual embedding of the full-size images associated with the two vertices. In this way, we encode the area around the instances along with their spatial relations captured from multiple views. The resulting embedding features are stored on the edge, enabling downstream querying (see Sec. III-D).

C. Task understanding via LLM

The task description provided in the query $\mathcal{Q}$ may involve varying levels of complexity. For example, in the query "*Control the temperature using the radiator dial under the gnomes*", there is a spatial expression that specifies the position of the *radiator*, the contextual object $\mathcal{O}$, relative to the *gnomes*, the referent object $\mathcal{R}$. In this case, the functional component $\mathcal{F}$, *radiator dial*, is also explicitly mentioned in this example. However, in many cases $\mathcal{F}$ is not expressed directly in the query $\mathcal{Q}$. For instance, when the task is "*Open the door*", the agent must infer that *handle* or *knob* on the door is the functional component to manipulate.

In this work, we focus on task descriptions that explicitly or implicitly include space contexts, **spatial relations** $\mathcal{S}$, **referent object** $\mathcal{R}$, **contextual object** $\mathcal{C}$, and **functional component** $\mathcal{F}$. We employ an LLM to extract this information from the task descriptions. The standalone output when querying individual categories can be ambiguous. E.g., the functional part "handle" can be part of a "door" or a "cup". The contextual object is also unclear when multiple instances of the same class exist in the environment. Hence, inspired by [15], we prompt the LLM to jointly provide all categories in a single JSON-format output via multi-turn conversations.

D. Contextual object grounding

The contextual object is retrieved by querying the scene graph based on the task description. We denote the visual and textual embedding functions by $f(\cdot)$ and $g(\cdot)$, respectively. $t_{\mathcal{R}}$, $t_{\mathcal{C}}$, and $t_{\mathcal{S}}$ indicate the text description for referent, contextual object, and spatial relations respectively. First, we convert the text descriptions of the contextual object $\mathcal{O}$ and the referent object $\mathcal{R}$ to textual embeddings using FG-CLIP [4]. Then, we select candidate nodes for both referent and contextual objects based on embedding similarity. For the referent object $\mathcal{R}$, we retrieve candidate nodes $\{R_j\}$ from the vertices $\mathcal{V}$ whose visual embeddings are most similar to the textual embedding of $\mathcal{R}$ (see Eq. 2). where $sim(\cdot\ , \cdot)$ computes cosine similarity of two embedding vectors. Analogously, we select candidate nodes $\{C_i\}$ associated to the contextual object $\mathcal{C}$, formulated in Eq. 3.

Once we obtain the candidates, we examine the edges between all pairs $\{S_{ij}\} = \{(C_i, R_j)\}$ to identify those that have high embedding similarity scores with the spatial relations $\mathcal{S}$ described in the query (see Eq. 4). The nodes in the corresponding candidate pairs are then anchored as the contextual object and referent objects.

$$R_j = \arg\max_{v_j \in \mathcal{V}} sim(f(v_j), g(t_{\mathcal{R}})) \tag{2}$$

$$C_i = \arg\max_{v_i \in \mathcal{V}} sim(f(v_i), g(t_{\mathcal{C}})) \tag{3}$$

$$S = \arg\max_{s \in S_{ij}} sim(f(s), g(t_{\mathcal{S}})) \tag{4}$$

E. Functional component segmentation

We segment the functional components of the contextual object by combining the VLM Molmo [7] with SAM [8]. Once the contextual-referent object pair is identified, we retrieve its corresponding views with the top-K highest-ranked visibility scores and input them into Molmo. To guide detection, the VLM is prompted with the ontology extracted from the task description (see III-C). Molmo outputs pixel coordinates corresponding to the functional component of interest. After that, we generate the 2D segmentation masks by prompting SAM with these pixels. The resulting 2D masks are then back-projected into corresponding 3D spaces. Finally, we obtain the 3D functionality segmentation by aggregating the lifted masks across multiple views.

IV. Experiments

A. Experiment setup

We evaluated our method on SceneFun3D [3], the only available dataset that supports functionality segmentation in 3D scenes by providing annotations for functional components together with a diverse set of tasks in indoor scenes.

For comparison, we use Fun3DU [15] as the main baseline methods on functionality segmentation tasks. In addition, T-FunS3D is compared against selected open-vocabulary 3D segmentation methods, including OpenMask3D [9], OpenIns3D [20], and LeRF [11]. To ensure consistency, we do not re-run the baseline methods on SceneFun3D, but instead report their results as presented in Fun3DU [15].

B. Results

We evaluated our method's ability to segment functional parts according to task descriptions on the validation split of the SceneFun3D[3] dataset. Table I presents the Average Precision (AP) and Average Recall (AR) at different Intersection over Union (IoU) thresholds for successful predictions. As shown, T-FunS3D outperforms all baselines, including Fun3DU [15], which requires an additional VLM to detect contextual objects across all views.

TABLE I

Performance comparison on SceneFun3D dataset.

Methods	mAP	AP50	AP25	mAR	AR50	AR25	mIoU
OpenMask3D [9]	0.0	0.0	0.0	1.2	1.4	2.6	0.1
OpenIns3D [20]	0.0	0.0	0.0	**32.3**	**37.1**	39.9	0.0
LERF [11]	0.0	0.0	0.0	23.9	24.6	25.1	0.0
Fun3DU [15]	6.1	12.6	23.1	23.9	32.9	40.5	11.5
T-FunS3D (ours)	**8.1**	**17.8**	**34.5**	23.8	35.8	**46.9**	**15.7**

TABLE II compares the runtime performance of T-FunS3D against two baselines for instance ① and functionality segmentation ②. Considering the full pipeline, T-FunS3D achieves better accuracy than the state-of-the-art

Fun3DU [15] in a much shorter time. Unlike Fun3DU, which segments instances of interest across all images, T-FunS3D and OpenMask3D generate class-agnostic instance proposals from point clouds ⓐ and compute embeddings only from views where each instance is most visible ⓑ. Moreover, Fun3DU must repeat the complete segmentation pipeline for every new query, whereas T-FunS3D further reduces runtime at stage ② by caching object visual embeddings with top-ranked views obtained at ①. T-FunS3D is faster than OpenMask3D at ⓑ because T-FunS3D retains only high-confidence proposals. A comparison with OpenMask3D at ② is not applicable, as OpenMask3D does not perform functionality segmentation.

TABLE II

RUNTIME COMPARISON AT INSTANCE AND FUNCTIONALITY SEGMENTATION

Methods	① OV Inst. Segm. (per-scene average)	②Func. Segm. (per-query average)
OpenMask3D [9]	ⓐ 30s + ⓑ 720s	-
Fun3DU [15]	1920s	300s
T-FunS3D (ours)	ⓐ **30s** + ⓑ **580s**	**150s**

Fig. 2 illustrates qualitative results of functionality segmentation with T-FunS3D. Our method aggregates 2D segmentation masks from views selected based on the visibility of the contextual object and projects them onto the 3D point cloud. However, images captured from large oblique viewing angles can lead to less accurate segmentation, as indicated by either low precision or low recall values in the two examples on the left in Fig. 2. In addition, T-FunS3D struggles to identify functional elements that are not physically attached to contextual objects, obtaining zero IoU for the ceiling light switch in the most right example.

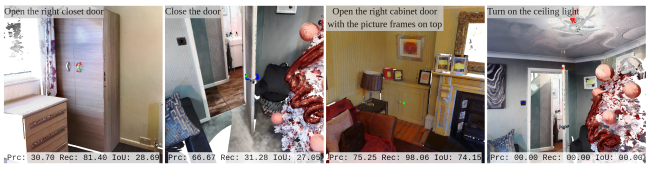

Fig. 2. Qualitative examples of T-FunS3D. We visualize point clouds around the functional components: red points indicate predictions, blue points denote ground truth, and green points represent overlaps.

V. CONCLUSION AND DISCUSSION

We propose T-FunS3D, a novel training-free method for referential functionality segmentation in 3D scenes using an open-vocabulary (OV) scene graph. As its main component, the OV scene graph represents nodes and edges with semantic embeddings, enabling T-FunS3D to localize task-relevant objects based on referential expressions and achieve accurate functionality segmentation. However, T-FunS3D can fail to detect functional parts from oblique views, leading to errors in functionality segmentation. Moreover, functional elements that are spatially separated from contextual objects, such as the switch of a ceiling light, remain particularly challenging. As future work, one may investigate more generalized approaches for contextual object proposals and address the recognition of spatially separated functional parts.

REFERENCES

[1] A. Takmaz, A. Delitzas, R. W. Sumner, F. Engelmann, J. Wald, and F. Tombari, "Search3D: Hierarchical Open-Vocabulary 3D Segmentation," *IEEE Robotics and Automation Letters*, vol. 10, 2024.

[2] Y. Yang, X. Wu, T. He, H. Zhao, and X. Liu, "Sam3d: Segment anything in 3d scenes," *arXiv preprint arXiv:2306.03908*, 2023.

[3] A. Delitzas, A. Takmaz, F. Tombari, R. Sumner, M. Pollefeys, and F. Engelmann, "Scenefun3d: Fine-grained functionality and affordance understanding in 3d scenes," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024.

[4] C. Xie, B. Wang, F. Kong, J. Li, D. Liang, G. Zhang, D. Leng, and Y. Yin, "Fg-clip: Fine-grained visual and textual alignment," *arXiv preprint arXiv:2505.05071*, 2025.

[5] J. Schult, F. Engelmann, A. Hermans, O. Litany, S. Tang, and B. Leibe, "Mask3D: Mask Transformer for 3D Semantic Instance Segmentation," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*, May 2023, pp. 8216–8223.

[6] A. Yang, A. Li, B. Yang, B. Zhang, B. Hui, B. Zheng, B. Yu, C. Gao, C. Huang, C. Lv *et al.*, "Qwen3 technical report," *arXiv preprint arXiv:2505.09388*, 2025.

[7] M. Deitke, C. Clark, S. Lee, R. Tripathi, Y. Yang, J. S. Park, M. Salehi, N. Muennighoff, K. Lo, L. Soldaini *et al.*, "Molmo and pixmo: Open weights and open data for state-of-the-art vision-language models," in *Proceedings of the Computer Vision and Pattern Recognition Conference*, 2025, pp. 91–104.

[8] A. Kirillov, E. Mintun, N. Ravi, H. Mao, C. Rolland, L. Gustafson, T. Xiao, S. Whitehead, A. C. Berg, W.-Y. Lo *et al.*, "Segment anything," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2023, pp. 4015–4026.

[9] A. Takmaz, E. Fedele, R. W. Sumner, M. Pollefeys, F. Tombari, and F. Engelmann, "OpenMask3D: Open-Vocabulary 3D Instance Segmentation," Oct. 2023, arXiv:2306.13631 [cs].

[10] P. D. A. Nguyen, T. D. Ngo, C. Gan, E. Kalogerakis, A. Tran, C. Pham, and K. Nguyen, "Open3DIS: Open-Vocabulary 3D Instance Segmentation with 2D Mask Guidance," *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. null, 2023.

[11] J. Kerr, C. M. Kim, K. Goldberg, A. Kanazawa, and M. Tancik, "Lerf: Language embedded radiance fields," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2023.

[12] F. Engelmann, F. Manhardt, M. Niemeyer, K. Tateno, M. Pollefeys, and F. Tombari, "Opennerf: open set 3d neural scene segmentation with pixel-wise features and rendered novel views," *arXiv preprint arXiv:2404.03650*, 2024.

[13] M. Liu, Y. Zhu, H. Cai, S. Han, Z. Ling, F. Porikli, and H. Su, "Partslip: Low-shot part segmentation for 3d point clouds via pretrained image-language models," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2023.

[14] Y. Zhou, J. Gu, X. Li, M. Liu, Y. Fang, and H. Su, "Partslip++: Enhancing low-shot 3d part segmentation via multi-view instance segmentation and maximum likelihood estimation," *arXiv preprint arXiv:2312.03015*, 2023.

[15] J. Corsetti, F. Giuliari, A. Fasoli, D. Boscaini, and F. Poiesi, "Functionality understanding and segmentation in 3D scenes," *ArXiv*, vol. abs/2411.16310, 2024.

[16] I. Armeni, Z.-Y. He, J. Gwak, A. R. Zamir, M. Fischer, J. Malik, and S. Savarese, "3d scene graph: A structure for unified semantics, 3d space, and camera," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2019.

[17] A. Rosinol, A. Gupta, M. Abate, J. Shi, and L. Carlone, "3d dynamic scene graphs: Actionable spatial perception with places, objects, and humans," *arXiv preprint arXiv:2002.06289*, 2020.

[18] 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.

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

[20] Z. Huang, X. Wu, X. Chen, H. Zhao, L. Zhu, and J. Lasenby, "OpenIns3D: Snap and Lookup for 3D Open-vocabulary Instance Segmentation," *ArXiv*, vol. abs/2309.00616, p. null, 2023.