Context-Aware Entity Grounding with Open-Vocabulary 3D Scene Graphs

Anonymous Author(s)
Affiliation
Address
email

Abstract: We present an Open-Vocabulary 3D Scene Graph (OVSG), a formal framework for grounding a variety of entities, such as object instances, agents, and regions, with free-form text-based queries. Unlike conventional semantic-based object localization approaches, our system facilitates context-aware entity localization, allowing for queries such as "pick up a cup on a kitchen table" or "navigate to a sofa on which someone is sitting". In contrast to existing research on 3D scene graphs, OVSG supports free-form text input and open-vocabulary querying. Through a series of comparative experiments using the ScanNet [1] dataset and a self-collected dataset, we demonstrate that our proposed approach significantly surpasses the performance of previous semantic-based localization techniques. Moreover, we highlight the practical application of OVSG in real-world robot navigation and manipulation experiments. The code and dataset used for evaluation will be made available upon publication.

Keywords: Open-Vocabulary Semantic, Scene Graph, Object Grounding

1 Introduction

In this paper, we aim to address a fundamental problem in robotics – grounding semantic entities within the real world. Specifically, we explore how to unambiguously and accurately associate entities present in commands, such as object manipulation, navigation to a specific location, or communication with a particular user. Currently, there are two prevailing methods for grounding entities in the robotics domain, namely fiducial markers [2] and semantic detection [3]. Fiducial markers require attaching physical tags to entities, which is not always feasible in real-world scenarios. Semantic detection only demands users to provide a semantic category. However, in scenes with multiple entities of the same category, semantic labels alone cannot provide a unique specification. In contrast, humans naturally possess the ability to overcome these grounding issues by providing context-aware specifications, such as detailed descriptions and relative relations. For example, rather than simply designating "a cup", humans often specify "a blue cup on the shelf", "a coffee cup in the kitchen", or "Mary's favorite tea cup". By taking into account not only the entity's category but also spatial information, user-object relations, and attributes, we can achieve a better solution for accurate specification in the field of robotics.

Recently, there has been a growing interest in leveraging 3D scene graphs [4, 5, 6, 7, 8] to bridge the gap between humans and robots. These approaches employ 3D scene graphs as a scene representation that concurrently accounts for instance categories and inter-instance spatial contexts. In a 3D scene graph, concepts such as people, objects, and rooms are depicted as nodes, with attributes like color, material, and affordance assigned as node attributes. Moreover, spatial relationships are represented as graph edges. Such structure enables 3D scene graphs to seamlessly support context-aware object queries, such as "the red cup on the table in the dining room", provided that the attribute, the semantic category, and the relationship have been predefined in the graph.

Submitted to the 7th Conference on Robot Learning (CoRL 2023). Do not distribute.

However, this inevitably brings us to a more crucial question that this paper aims to answer: how do we cope with scenarios when the class category, relationship, and attribute are not available in the constructed 3D scene graph? Tackling this question is vital if we wish to effectively integrate robots into real-world scenarios. To resolve the challenge, we present a novel framework in this paper – the Open-Vocabulary 3D Scene Graph (OVSG). To the best of our knowledge, OVSG is the first 3D scene graph representation that facilitates context-aware entity grounding, even with unseen semantic categories and relationships.

To evaluate the performance of our proposed system, we conduct a series of query experiments on ScanNet [1] and a self-collected dataset DOVE-G (Dataset for Open-Vocabulary Entity Grounding). We demonstrate that by combining open-vocabulary detection with 3D scene graphs, we can ground entities more accurately in real-world scenarios than using state-of-the-art open-vocabulary semantic localization methods alone. Additionally, we designed two experiments to investigate the open-vocabulary capability of our framework. Finally, we showcase potential applications of OVSG through a demonstration of real-world robot navigation and manipulation.

Our contributions are threefold: 1) A new dataset contains eight unique scenarios and 4,000 language queries for context-aware entity grounding. 2) A novel 3D scene graph-based method to address the context-aware entities grounding from open-vocabulary queries. 3) Demonstrate the real-world applications of OVSG, such as context-aware object navigation and manipulation.

2 Related Work

Open-Vocabulary Semantic The development of foundation vision-language pre-trained models, such as CLIP [9], ALIGN [10], and LiT [11], has facilitated the progress of 2D open-vocabulary object detection and segmentation techniques [12, 13, 14, 15, 16, 17, 18]. Among these approaches, Detic [16] stands out by providing open-vocabulary instance-level detection and segmentation simultaneously. However, even state-of-the-art single-frame methods like Detic suffer from perception inconsistency due to factors such as view angle, image quality, and motion blur. To address these limitations, Lu et al. proposed OVIR-3D [19], a method that fuses the detection result from Detic into an existing 3D model using 3D global data association. After fusion, the 3D scene is segmented into multiple instances, each with a unique Detic feature attached. Owing to its stable performance, we choose OVIR-3D as our semantic backbone.

3D Scene Graph 3D scene graphs provide an elegant representation of objects and their relationships, encapsulating them as nodes and edges, respectively. The term "3D" denotes that each node within the scene possesses a three-dimensional position. In [4], Fisher et al. first introduced the concept of 3D scene graphs, where graph nodes are categorized by geometric shapes. Armeni et al. [5] and Kim et al. [6] then revisited this idea by incorporating semantic labels to graph nodes. These works establish a good foundation for semantic-aware 3D scene graphs, demonstrating that objects, rooms, and buildings can be effectively represented as graph nodes. Recently, Wald et al. [8] showed that 3D feature extraction and graph neural networks (GNN) can directly infer semantic categories and object relationships from raw 3D point clouds. Rosinol et al. [7] further included dynamic entities, such as users, within the scope of 3D scene graph representation. While 3D scene graphs exhibit great potential in object retrieval and long-term motion planning, none of the existing methods support open-vocabulary queries and direct natural language interaction. Addressing these limitations is crucial for real-world deployment, especially for enabling seamless interaction with users.

3 Open-Vocabulary 3D Scene Graph

3.1 Open-Vocabulary 3D Scene Graph Representation

An Open-Vocabulary 3D Scene Graph (OVSG) is denoted as $G = |V, E|$, where V signifies graph nodes and E stands for graph edges. Each node v^i in $V = \{v^i\} = \{t^i, f^i, l^i, p^i\}$ consists of a node type t^i, a open-vocabulary feature f^i, a language description l^i (optional), and

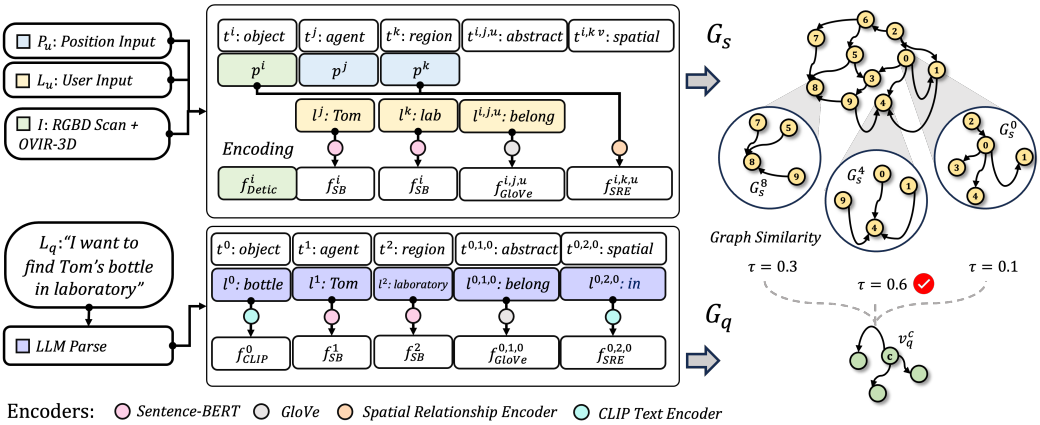

Figure 1: This is an illustration of the proposed pipeline. The system inputs are the positional input P_u, user input L_u, RGBD Scan I, and a query language L_q. The top section depicts the construction of G_s. Both P_u and L_u are directly fed into G_s. The RGBD Scan I inputs into the open-vocabulary fusion system referred to as *OVIR-3D*. This system outputs a position and a Detic feature for each object. Subsequently, the language descriptions for the agent and region are converted into features via different encoders. A unique Spatial Relationship Encoder is employed to encode spatial relationship features from pose pairs. The bottom section shows the building of G_q. The query L_q used in this example is, "I want to find Tom's bottle in the laboratory." An LLM is used to parse it into various elements, each with a description and type. These descriptions are then encoded into features by different encoders, forming G_q. Finally, grounding the query language L_q within scene S becomes a problem of locating G_q within G_s. A candidate proposal and ranking algorithm are introduced for this purpose. The entity we wish to locate is represented by the central node of the selected candidate.

a 3D position p^i (optional); i is the node index. In this study, we identify three primary node types, t^i: object, agent, and region. The open-vocabulary feature f^i associated with each node v_i is contingent on the node type t_i. The encoder utilized for f^i is accordingly dependent on t_i. The 3D position $p^i = \{x_c, y_c, z_c, x_{min}, y_{min}, z_{min}, x_{max}, y_{max}, z_{max}\}$ of each entity is defined by a 3D bounding box and its center position. Edges in the graph are represented by $E = \{e^{i,j} | v^i, v^j \in V\}, e^{i,j} = \{r^{i,j,k} = \{t^{i,j,k}, f^{i,j,k}, l^{i,j,k}\} | k = 0, \ldots\}$. Each edge $e^{i,j}$ encapsulates all relationships $r^{i,j,k}$ between the nodes v^i and v^j. The triplet notation (i, j, k) refers the kth relationship between node v^i and v^j, $t^{i,j,k}$ indicates the type of this relationship. We primarily categorize two relationships in this study: spatial relations and abstract relationships. A short sentence l^i is optionally provided to describe this relationship. The feature $f^{i,j,k}$ encodes the semantic meaning of the relationship, whose encoder depends on $t^{i,j,k}$. For a more detailed definition of these types, please refer to Sec. 3.3.

The primary distinction of OVSG from conventional 3D scene graph work is its utilization of semantic features, instead of discrete labels, to characterize nodes and relationships. These features are either directly trained within the language domain like Sentence-BERT [20] and GloVe [21], or aligned to it, as seen with CLIP [9] and Detic [16]. The versatility of language features enables OVSG to handle diverse queries. The degree of similarity among nodes and edges is depicted using a distance metric applied to their features:

$$\text{dist}(v^i, v^j) = \begin{cases} \infty & \text{if } t^i \neq t^j \\ 1 - \text{dot}(f^i, f^j) & \text{else} \end{cases} ; \text{dist}(e^{i,j}, e^{u,v}) = \min_{\forall k \in |e^{i,j}|, \forall w \in |e^{u,v}|} \text{dist}(r^{i,j,k}, r^{u,v,w})$$

$$\text{dist}(r^{i,j,k}, r^{u,v,w}) = \begin{cases} \infty & \text{if } t^{i,j,k} \neq t^{u,v,w} \\ 1 - \text{dot}(f^{i,j,k}, f^{u,v,w}) & \text{if } t^{i,j,k} = t^{u,v,w} \neq \text{spatial} \\ \text{SRD}(f^{i,j,k}, f^{u,v,w}) & \text{if } t^{i,j,k} = t^{u,v,w} = \text{spatial} \end{cases} \quad (1)$$

, where the $|e^{i,j}|$ and $|e^{u,v}|$ are the number of relationships inside $e^{i,j}$ and $e^{u,v}$; *SRD* refers to a Spatial Relationship Decoder. Check Sec. 3.3 and Appendix. C for more details.

3.2 Context-Aware Open-Vocabulary Entities Grounding

The problem we address can be formally defined using the open-vocabulary scene graph concept as follows: Given a scene, represented as S, our objective is to localize an entity, referred to as s, using natural language, represented as L_q, within the context of the scene S. Essentially, we seek to

establish a mapping π such that $s = \pi(L_q|S)$. An RGBD scan of the scene I, user linguistic input L_u, and position input P_u are provided to facilitate this process. Significantly, the query language L_q may encompass entity types and relationship descriptions not previously included in the scene graph construction phase.

Our proposed procedure can be separated into two main stages. The first stage involves the construction of the scene graph. From the user input L_u and the RGBD scan I, we construct an open-vocabulary scene graph (OVSG) for the entire scene, denoted as G_s. This is a one-time process that can be reused for every subsequent query. When a new query is introduced, we also construct an OVSG using the query L_q, denoted as G_q. Once we have both scene graphs G_s and G_q, we proceed to the second stage, which is the graph matching stage. Here, we match the query scene graph, G_q, with a sub-graph from the whole scene graph, G_s. The queried entity is situated within the matched sub-graph.

3.3 3D Scene Graph Building

Type definition Prior to delving into the scene graph construction procedure, we first delineate the categories of node types and edge types this paper pertains to. The term *Object* signifies static elements within a scene, such as sofas, tables, and so forth. The term *Agent* is attributed to dynamic, interactive entities in the scene, which could range from humans to robots. *Region* indicates a specific area, varying in scale from the surface of a tabletop to an entire room or building. Regarding relationships, *spatial* describes positional relationships between two entities, such as Tom being in the kitchen. Conversely, *abstract* relationships are highly adaptable, enabling us to elucidate relationships between an agent and an object (for instance, a cup belonging to Mary) or the affordance relationship between two objects, such as a key being paired with a door.

Input process The inputs for G_s consist of an RGBD-scan set I, a user language input L_u, and a user position input P_u. The L_u input assigns names to agents and regions and provides descriptions of abstract relationships. P_u provides the locations for the agent and region, and it can be autonomously generated using existing algorithms like DSGS [7]. Since this process is not the focus of our study, we assume P_u is pre-determined in this paper. The input I is an RGBD-scan of the entire scene, which is fed into the Open-Vocabulary 3D Instance Retrieval (OVIR-3D) [19] system, a fusion system operating at the instance level. OVIR-3D returns a set of objects, each denoted by a position p^i and a Detic feature f^i_{Detic}.

G_q accepts a language query L_q as its input. An exemplary query, as depicted in Fig.1, is "I want to find Tom's bottle in the laboratory". To parse this language, we utilize a large language model (LLM), such as GPT-3.5 or LLAMA. Utilizing a meticulously engineered prompt (refer to Appendix. B for more details), we can interpret different entities within the query.

Feature encoding As specified in Eq. 1, the calculation of the similarity between nodes and edges relies heavily on their features. This operation of computing features is termed the feature encoding process. Primarily, let's delve into feature encoding for G_s. In G_s, the feature for the object is conveniently provided by the OVIR-3D system. Conversely, both the agent node and the region node come with a name l. We utilize a Sentence-BERT [20] to encode a prototype "A is l". The sentence transformer is selected for its generality, enabling it to manage rare names effectively.

Concerning abstract relationships, each one is accompanied by a language description l, which we encode using GloVe [21]. We opt for GloVe over the sentence transformer because GloVe can effectively identify subtle variations in abstract relationships. For instance, GloVe can discern that "like" is closer to "favorite" than "neutral", which is beyond the capability of the sentence transformer. Lastly, for spatial relationships, since their inputs are positions that must be aligned with language-based spatial descriptions, we have devised an encoder-decoder structure. Owing to space constraints, we have included the specifics of this spatial encoder-decoder in Appendix C. In G_s, the spatial relationship feature is encoded from a pose pair utilizing a Spatial Relationship Encoder (SRE).

As for G_q, we employ the language description parsed by the LLM to encode features for all entities and relationships. We use the Sentence-BERT to encode the name of the agent and region. The object description and spatial relationship description are both encoded by a CLIP text encoder. The abstract relationship is again encoded by GloVe.

3.4 Sub-graph Matching

After obtaining the OVSG for the entire scene and the query, denoted as G_s and G_q respectively, the problem of grounding L_q within the scene S becomes locating G_q within G_s. Generally, the subgraph-matching problem is NP-hard, prompting us to make several assumptions to simplify this problem. In this study, we assume that our G_q is a star-graph, signifying that a central node exists and all other nodes are exclusively linked to this central node. (If G_q is not a star-graph, we will extract a sub-star-graph from it, and use this sub-graph as our query graph.)

The pipeline of sub-graph matching is illustrated on the right side of Fig. 1. This a two-step procedure: candidate proposal and re-ranking. Let's denote the center of G_q as v_q^c. Initially, we traverse all nodes, v_s^i, in V_s, ranking them based on their distance to v_q^c, computed with Eq. 1. Subsequently, we extract the local subgraph, G_s^i, surrounding each candidate, v_s^i. These extracted subgraphs serve as our candidate subgraphs. In the second phase, we re-rank these candidates using a graph-similarity metric, $\tau(G_q, G_s^i)$. To evaluate graph similarity, we examine three distinct methodologies: Likelihood, Jaccard coefficient, and Szymkiewicz-Simpson index.

Likelihood Assuming the features of nodes and edges all originate from a normal distribution, we can define the likelihood of nodes and edges being identical as follows: $L(v^i = v^j) = exp(\frac{-\text{dist}(f^i, f^j)}{\sigma_v})$ for nodes and $L(e^{i,j} = e^{u,v}) = exp(\frac{-\text{dist}(f^{i,j}, f^{u,v})}{\sigma_e})$ for edges. Here σ_v and σ_e are balancing parameters. From this, we can derive the graph-level likelihood τ_L as:

$$\tau_L(G_q, G_s^i) = L(v_q^c = v_{s^i}^c) \times \underset{(k,\pi(k)) \in C_M^N}{\text{argmax}} \Pi_k \left[L(v_q^k = v_{s^i}^{\pi(k)}) \cdot L(e_q^{c,k} = e_{s^i}^{c,\pi(k)}) \right] \tag{2}$$

where $v_{s^i}^c$ is the center node of G_s^i. $(k, \pi(k))$ refers to a possible node-level data association between G_q and G_s^i. $M = \max(|V_q|, |V_{s^i}|), N = \min(|V_q|, |V_{s^i}|)$. C_M^N refers to all possible associations. The idea behind this formula is to iterate over all possible node-level associations and select the one that maximizes the overall likelihood that G_q matches with G_s^i.

Jaccard-coefficient & Szymkiewicz–Simpson In addition to the likelihood index, we also consider other widely used graph similarity indices such as the Jaccard and Szymkiewicz–Simpson indices. Both indices measure the similarity between two sets.

We adopt a similar method as in [8], generating a set $S(G)$ for each graph G by combining nodes and edges, such that $|S(G)| = |V| + |E|$. The Jaccard coefficient τ_J and Szymkiewicz–Simpson index τ_S are then defined as follows:

$$\tau_J(G_q, G_s^i) = \frac{|S(G_q) \cap S(G_s^i)|}{|S(G_q)| + |S(G_s^i)| - |S(G_q) \cap S(G_s^i)|}, \tau_S(G_q, G_s^i) = \frac{|S(G_q) \cap S(G_s^i)|}{min(|S(G_q)|, |S(G_s^i)|)} \tag{3}$$

Given that we already know $|S(G_q)|$ and $|G_s^i|$, we simply need to compute $|S(G_q) \cap S(G_s^i)|$, which consists of nodes or edges that belong to both G_q and G_s^i. We can define this union by applying distance thresholds τ_v and τ_e for node and edge separately:

$$S(G_q) \cap S(G_s^i) = \{(v_q^k, v_{s^i}^{\pi(k)}) | dist(f_q^k, f_{s^i}^{\pi(k)}) < \epsilon_v\} + \{(e_q^k, e_{s^i}^{\pi(k)}) | dist(e_q^k, e_{s^i}^{\pi(k)}) < \epsilon_e\} \tag{4}$$

Here, π is a data association between G_q and G_s^i, where $\pi(k) = \text{argmin}_{\pi(k)}(dist(s_k, s_{\pi(k)}))$. ϵ_v and ϵ_e are threshold parameters.

The differences between τ_L, τ_J, and τ_S can be understood as follows: τ_L describes the maximum likelihood among all possible matches between G_q and G_s^i. Both τ_J and τ_S use thresholds ϵ_v, ϵ_e to convert the node and edge matches to binary, and they measure the overall match rate with different normalization.

4 System Evaluation

Our experiments with the OVSG framework aimed to answer these research questions: 1) How does our context-aware grounding method compare to the OVIR-3D approach? 2) How well does OVSG handle open-vocabulary queries? 3) What differences do our graph similarity-based methods show? 4) How well does OVSG perform inside a real robot environment?

4.1 Dataset, Metrics & Baselines

ScanNet We are using the validation set of ScanNet for evaluation, consisting of 312 scenes. Since ScanNet only includes objects, we emulated agents, induced their abstract relationships to objects, captured spatial relationships between objects, and extracted object features via OVIR-3D before integrating the dataset into our evaluation pipeline. Resource limitations prevented manual labeling of scenes; hence, we synthetically generated 43000 queries for evaluation (details in Appendix A).

DOVE-G To accommodate the richness of open-vocabulary queries, we introduced a custom dataset—DOVE-G (Dataset for Open-Vocabulary Entity Grounding). This is to facilitate users to query for objects within a scene using natural language. For each scene within DOVE-G, we manually labeled the ground truth and created 50 natural language queries (L_q). To augment this query set, we harnessed LLMs to generate four additional sets of natural language queries. This approach yielded a total of 250 queries for each scene, and cumulatively, we have 4000 queries to evaluate OVSG's performance. With this setup, we set out to assess how our OVSG framework performs with open-vocabulary queries, one of our key research questions, providing a critical testbed for its effectiveness in handling diverse natural language expressions.

Metric	Top1 Scores				Top3 Scores			
	OVIR-3D	OVSG-J	OVSG-S	OVSG-L	OVIR-3D	OVSG-J	OVSG-S	OVSG-L
Match Rate	13.80	20.73	26.35	**31.19**	34.72	29.49	37.18	**38.47**
IoU	0.14	0.21	0.25	**0.29**	0.35	0.29	0.36	**0.38**

Table 1: Overall Performance of the OVSG framework on ScanNet

Metric	Top1 Scores				Top3 Scores			
	OVIR-3D	OVSG-J	OVSG-S	OVSG-L	OVIR-3D	OVSG-J	OVSG-S	OVSG-L
Match Rate	35.20	25.21	45.42	**51.10**	63.66	47.15	64.86	**65.78**
IoU	0.28	0.18	0.34	**0.39**	0.49	0.39	0.49	**0.50**

Table 2: Overall Performance of the OVSG framework on DOVE-G

Evaluation Metrics For each query, we calculated IoU scores between the bounding boxes of the correct match (ground truth) and the top candidates returned by the system. We also measured the match rate for each scene, which refers to the percentage of queries where the bounding box of the predicted candidate overlapped with the ground truth with an IoU score of 0.5 or more. This threshold can be adjusted to alter the strictness of the match criteria. We reported the Top1 and Top3 match rates and average IoU scores for each scene, reflecting the performance of our system in the Top-k results returned for each query.

Baselines We evaluate four different methods in our experiments. The SOTA open-vocabulary grounding method, OVIR-3D, is our baseline as it will not leverage any inter-notion relations, providing a comparative measure for the effectiveness of contextual information integration in the other methods. The rest of the methods perform Context-Aware Entity Grounding by employing subgraph matching techniques (refer to Subsection 3.4). We denote those three graph similarity method

variants as OVSG-J, OVSG-S, and OVSG-L (for Jaccard coefficient, Szymkiewicz-Simpson index, and Likelihood, respectively).

4.2 Performance

ScanNet Concerning the ScanNet dataset, we averaged the scores for the 312 scenes and presented the results in Table 1. From an overall perspective, leveraging contextual information for grounding entities proved useful as the graph similarity variants (OVSG-J, OVSG-S, OVSG-L) outperformed the SOTA semantic segmentation method (OVIR-3D). Especially, the OVIR-3D method performed poorly for scenes of type bookstore or library, or conference room where there are multiple instances of the same entity. Hence, a lack of context/specification for the target object being queried makes the object retrieval process less accurate. The standout was the OVSG-L method, yielding an average Top1 Match Rate and IoU of 31.19% and 0.29, and Top3 rates of 38.47% and 0.38.

DOVE-G Table 2 summarizes the performance of all the methods across every scene in DOVE-G, averaged over all five sets of natural language queries. OVSG-L consistently outperforms other methods across all scenes (refer to Table 7 in Appendix D). OVSG-J and OVSG-S also show competitive results in some scenes. Despite these bright spots, both methods are generally overshadowed by OVSG-L's superior performance. The OVIR-3D model does manage to stay competitive with the other three methods, particularly in the Top3 categories. Because scenes in DOVE-G do not contain a large number of identical instances as in a stack of several books in a library scene, the Top3 results returned by the system usually contain the ground truth match.

Vocab Set	Top1 Match Rate				Top3 Match Rate			
	OVIR-3D	OVSG-J	OVSG-S	OVSG-L	OVIR-3D	OVSG-J	OVSG-S	OVSG-L
#1	35.14	23.71	42.29	**57.14**	53.14	40.86	55.71	**57.14**
#2	35.71	27.43	45.43	**51.71**	58.86	46.86	62.57	**62.57**
#3	32.29	26.00	43.43	**48.86**	58.86	47.71	60.57	**62.00**
#4	33.14	20.29	42.29	**44.86**	55.71	42.86	57.14	**56.57**
#5	42.86	29.71	52.57	**57.43**	62.57	52.29	65.71	**65.43**
Overall	35.83	25.43	45.20	**50.00**	57.83	46.12	60.34	**60.74**
	Top1 IoU				Top3 IoU			
	OVIR-3D	OVSG-J	OVSG-S	OVSG-L	OVIR-3D	OVSG-J	OVSG-S	OVSG-L
#1	0.31	0.18	0.36	**0.42**	0.50	0.39	0.51	**0.53**
#2	0.31	0.22	0.38	**0.44**	0.53	0.43	0.55	**0.56**
#3	0.29	0.21	0.37	**0.43**	0.52	0.43	0.54	**0.56**
#4	0.30	0.17	0.36	**0.40**	0.51	0.39	0.52	**0.52**
#5	0.38	0.24	0.44	**0.50**	0.57	0.47	0.59	**0.60**
Overall	0.32	0.20	0.38	**0.44**	0.53	0.42	0.54	**0.56**

Table 3: Performance comparison against five different varied open-vocabulary sets

Relationship Set	Top1 Match Rate				Top3 Match Rate			
	OVIR-3D	OVSG-J	OVSG-S	OVSG-L	OVIR-3D	OVSG-J	OVSG-S	OVSG-L
#1	35.14	19.50	36.00	**41.38**	54.12	40.88	53.88	**58.12**
#2	35.71	21.50	36.75	**44.62**	59.12	47.62	62.38	**62.88**
#3	32.29	23.25	38.25	**42.12**	58.88	49.88	61.12	**59.88**
#4	33.14	18.75	36.50	**40.38**	56.38	43.12	55.38	**56.38**
#5	42.86	22.75	43.25	**47.88**	62.38	52.38	65.12	**65.38**
Overall	35.83	21.15	38.15	**43.28**	58.18	46.76	59.58	**60.53**
	Top1 IoU				Top3 IoU			
	OVIR-3D	OVSG-J	OVSG-S	OVSG-L	OVIR-3D	OVSG-J	OVSG-S	OVSG-L
#1	0.30	0.17	0.31	**0.36**	0.50	0.38	0.48	**0.53**
#2	0.30	0.19	0.32	**0.39**	0.54	0.43	0.55	**0.56**
#3	0.29	0.20	0.33	**0.38**	0.53	0.45	0.54	**0.54**
#4	0.30	0.16	0.30	**0.36**	0.52	0.40	0.50	**0.52**
#5	0.38	0.20	0.37	**0.42**	0.57	0.48	0.58	**0.60**
Overall	0.32	0.18	0.33	**0.38**	0.53	0.43	0.53	**0.55**

Table 4: Performance comparison against five different varied relationship sets

4.3 Open-Vocabulary Analysis

Having presented insights on our system's performance on natural language queries for DOVE-G (as shown in Table 2), we proceed to deepen our investigation into the system's resilience across diverse query sets. To accomplish this, we instead average the results from all scenes for each of the five vocabulary sets (refer to Table 3). By doing so, we aim to provide a robust evaluation of our system's performance across a variety of query structures and word choices, simulating the varied ways in which users may interact with our system. In addition to experimenting with object vocabulary variations (a 'coffee maker' to 'espresso machine' or 'coffee brewer'), and altering the order of entity referencing in the query, we also studied the impact of changing relationship vocabulary. In this experimental setup, the LLM is not bound to map relationships to a pre-determined set as before. Instead, the graph-based query contains a variety of relationship vocabulary. To illustrate, consider the queries "A is to the left back corner of B" and "A is behind and left to B". Previously, these relationships would map to a fixed relation like 'left and behind'. Now, 'front and left' as interpreted by the LLM can variate to 'leftward and ahead', 'northwest direction', or 'towards the front and left', offering a broader range of relationship descriptions. The evaluation results for these query sets are presented in Table 4.

Varying object names Across all evaluated vocabulary sets, OVSG-L demonstrates the highest Top1 and Top3 match rates, outperforming the remaining methods. This pattern also persists for scores in the IoU category. Notably, OVSG-L's match rates span from 44.86% to 57.43% for Top1, and 56.57% to 65.43% for Top3. All in all, contextual understanding of the target again proves to

improve results from 35.83% (OVIR-3D) to 50% (OVSG-L) for Top1 match rate and 0.32 to 0.44 for the Top1 IoU.

Varying relationships Compared to Table 3, we observe a noticeable decrease in performance for the methods under the OVSG framework. This is likely due to the increased complexity introduced by the varied word choices for edges (relationships) in the sub-graph being matched. Despite this, two of the OVSG methods still outperform the OVIR-3D method, with the OVSG-L method delivering the strongest results.

object	shoe	bottle	chair	trash can#1	trash can#2	drawer	cloth
success rate %	100.0	100.0	100.0	100.0	100.0	100.0	0.0

Table 5: Success rate of object navigation task

Method	scene1	scene2	scene3
OVIR-3D (%)	0.0	0.75	0.33
OVSG (%)	0.88	0.75	0.75

Table 6: Success rate of manipulation task

4.4 Robot application

Manipulation In order to exemplify the utility of OVSG in real-world manipulation scenarios, we devised a complex pick-and-place experiment. In this task, the robot is instructed to select one building block and position it on another. The complexity of the task stems from the multitude of blocks that are identical in both shape and color, necessitating the use of spatial context for differentiation. Each task consists of a picking action and a placing action. We formulated nine distinct tasks for this purpose (please refer to Appendix. E for detailed setup). The effectiveness of the manipulation task was evaluated by comparing the success rate achieved by OVIR-3D and our newly proposed OVSG-L. The outcome of this comparative study is depicted in the accompanying table. The results demonstrate that our innovative OVSG-L model significantly enhances the object grounding accuracy in manipulation tasks involving a high prevalence of identical objects. This improvement highlights the potential of OVSG-L in complex manipulation scenarios, paving the way for further exploration in the field of robotics.

Navigation We conducted a system test on a ROSMASTER R2 Ackermann Steering Robot for an object navigation task. The detailed setup can be found in Appendix. F. We provided queries for seven different objects within a lab scene, using three slightly different languages to specify each object. These queries were then inputted into OVSG, and the grounded positions of the entities were returned to the robot. We considered the task successful if the robot's final position was within 1 meter of the queried objects. The results are presented in Table. 5. From the table, it is evident that the proposed method successfully located the majority of user queries. However, there was one query that was not successfully located: "The cloth on a chair in the office." In this case, we found OVIR-3D incorrectly recognized the cloth as part of a chair, resulting in the failure to locate it.

5 Conclusion & Limitation

While our proposed OVSG system has made significant progress, three key limitations currently hinder its deployment. First, OVSG heavily relies on an open-vocabulary fusion system like OVIR-3D, which may lead to missed queries if the system fails to identify an instance. Second, the current language processing system's strong dependence on LLMs exposes it to inaccuracies, as any failure in parsing the query language may yield incorrect output. Third, as discussed in Section 3.4, calculating graph likelihood by multiplying nodes and edges likelihoods may not be optimal, as likelihoods from distinct types might carry varying levels of importance and distribution. Accurately balancing these factors remains a challenge for future research, as our efforts with a GNN have not yielded satisfactory results.

Despite the aforementioned areas for improvement, we observe that OVSG significantly improves context-aware entity grounding compared to existing open-vocabulary semantic methods. Since OVSG only requires natural language as the query input, we believe it holds great potential for seamless integration into numerous existing robotics systems.

References

[1] A. Dai, A. X. Chang, M. Savva, M. Halber, T. Funkhouser, and M. Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE*, 2017.

[2] E. Olson. Apriltag: A robust and flexible visual fiducial system. In *2011 IEEE International Conference on Robotics and Automation*, pages 3400–3407, 2011. doi:10.1109/ICRA.2011.5979561.

[3] K. He, G. Gkioxari, P. Dollár, and R. Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017.

[4] M. Fisher, M. Savva, and P. Hanrahan. Characterizing structural relationships in scenes using graph kernels. The first paper that uses 3d scene-graph.

[5] 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. 10 2019. URL http://arxiv.org/abs/1910.02527.

[6] 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*, 50:4921–4933, 12 2020. ISSN 21682275. doi:10.1109/TCYB.2019.2931042.

[7] A. Rosinol, A. Gupta, M. Abate, J. Shi, and L. Carlone. 3d dynamic scene graphs: Actionable spatial perception with places, objects, and humans. 2 2020. URL http://arxiv.org/abs/2002.06289.

[8] J. Wald, H. Dhamo, N. Navab, and F. Tombari. Learning 3d semantic scene graphs from 3d indoor reconstructions. 4 2020. URL http://arxiv.org/abs/2004.03967.

[9] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR, 2021.

[10] C. Jia, Y. Yang, Y. Xia, Y.-T. Chen, Z. Parekh, H. Pham, Q. Le, Y.-H. Sung, Z. Li, and T. Duerig. Scaling up visual and vision-language representation learning with noisy text supervision. In *International Conference on Machine Learning*, pages 4904–4916. PMLR, 2021.

[11] X. Zhai, X. Wang, B. Mustafa, A. Steiner, D. Keysers, A. Kolesnikov, and L. Beyer. Lit: Zero-shot transfer with locked-image text tuning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18123–18133, 2022.

[12] B. Li, K. Q. Weinberger, S. Belongie, V. Koltun, and R. Ranftl. Language-driven semantic segmentation. *arXiv preprint arXiv:2201.03546*, 2022.

[13] G. Ghiasi, X. Gu, Y. Cui, and T.-Y. Lin. Scaling open-vocabulary image segmentation with image-level labels. In *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXXVI*, pages 540–557. Springer, 2022.

[14] J. Xu, S. De Mello, S. Liu, W. Byeon, T. Breuel, J. Kautz, and X. Wang. Groupvit: Semantic segmentation emerges from text supervision. *arXiv preprint arXiv:2202.11094*, 2022.

[15] X. Gu, T.-Y. Lin, W. Kuo, and Y. Cui. Open-vocabulary object detection via vision and language knowledge distillation. In *International Conference on Learning Representations*.

[16] X. Zhou, R. Girdhar, A. Joulin, P. Krähenbühl, and I. Misra. Detecting twenty-thousand classes using image-level supervision. In *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part IX*, pages 350–368. Springer, 2022.

[17] A. S. M. N. D. W. A. D. A. M. A. A. M. D. Z. S. X. W. X. Z. T. K. N. H. Matthias Minderer, Alexey Gritsenko. Simple open-vocabulary object detection with vision transformers. *ECCV*, 2022.

[18] L. H. Li, P. Zhang, H. Zhang, J. Yang, C. Li, Y. Zhong, L. Wang, L. Yuan, L. Zhang, J.-N. Hwang, et al. Grounded language-image pre-training. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10965–10975, 2022.

[19] L. Shiyang, C. Haonan, J. Eric, B. Abdeslam, and B. Kostas. Ovir-3d: Open-vocabulary 3d instance retrieval without training on 3d data. 6 2023. URL https://ovir-3d.github.io/.

[20] N. Reimers and I. Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 11 2019. URL https://arxiv.org/abs/1908.10084.

[21] J. Pennington, R. Socher, and C. D. Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.

A Synthetic Query Generation for ScanNet

In the ScanNet dataset, each scene comes with ground-truth labels for its segmented instances or objects. We began by calculating the spatial relationships between these ground-truth objects or entities. Subsequently, agents were instantiated into the scene, and abstract relationships were randomly established between the agents and the entities present in the scene. After generating the OVSG for each scene, our next step involved the creation of graph-based queries (refer to syntax and details in Appendix B) for evaluation purposes. For each of these queries, we randomly selected reference entities from the OVSG that shared a relationship with the target entity. This formed the basis of the synthetic generation of the graph-based queries for the ScanNet dataset.

B Prompt Engineering for Query Parse

The goal is to make LLM understand and process natural language queries in a structured way. The dialogue examples first start with a user input, containing a query in natural language. The LLM then deconstructs the query into a set of 'notions'. Notions are the key elements that exist within the query; they can be specific objects like a 'cup' or 'cushion', or agents like 'Tom' or 'Mary'. The LLM then determines the 'target' entity the user is interested in.

The LLM also identifies the relationship between the notions, mapping these relationships onto predefined categories, such as spatial or abstract relationships. This mapping helps the LLM to understand the dynamics between different notions. Each relationship is expressed in a structured graph form. This consistent structure helps in processing the data and can make the LLM's responses more predictable and easier to control.

After breaking down the query, the LLM then presents the structured interpretation of the user's request, recapping the notions, target, and relationships identified.

An example natural language query (L_q) Consider this natural language query as an example: "Could you point out Zoro's go-to cup, which we usually keep to the right of our espresso machine, on the left of the trash can, and in front of the coffee kettle?"

In this query, the user is asking about the location of a cup, which has three different spatial relationships with other reference entities and one abstract relationship with a user named Zoro.

Corresponding graph-based query

```
target @ cup {object}
zoro {user} – like [like] – cup {object}
cup {object} – right to [spatial] – espresso machine {object}
cup {object} – left to [spatial] – trash can {object}
coffee kettle {object} – behind [spatial] – cup {object}
```

The breakdown of the graph-based query is as follows:

target @ cup {object}: This line specifies the target object, which is a cup.

zoro {user} – like [like] – cup {object}: This line represents a relationship between a user named Zoro (user) and the cup (an object) that Zoro likes the cup (Zoro's favorite). In our current implementation, like is a relation of type *like*.

cup {object} – right to [spatial] – espresso machine {object}: This line represents a spatial relationship between the cup (an object) and the espresso machine (an object). The cup is positioned to the right of the espresso machine.

cup {object} – left to [spatial] – trash can {object}: This line represents a spatial relationship between the cup (an object) and the trash can (an object). The cup is positioned to the left of the trash can.

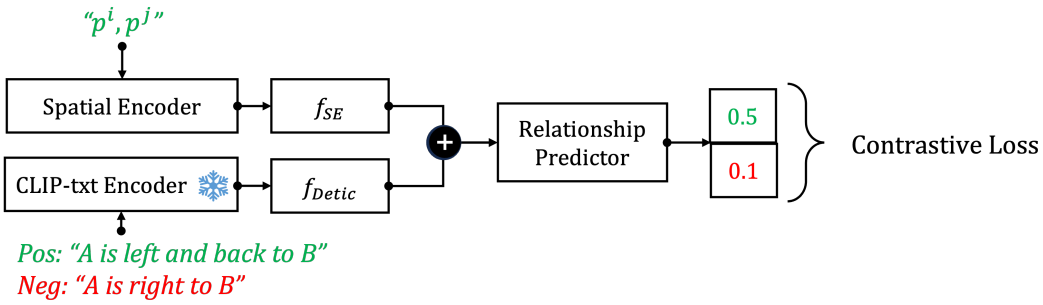

Figure 2: Training Pipeline of Spatial Relationship Prediction.

coffee kettle {object} – behind [spatial] – cup {object}: This line describes a spatial relationship between the coffee kettle (an object) and the cup (an object). The coffee kettle is positioned behind the cup.

C Spatial Relationship Prediction Pipeline

The purpose of the Spatial Relationship Encoder is to establish a connection between a pair of poses and their corresponding language description that elucidates the spatial relationship. The architecture of the Spatial Relationship Prediction pipeline is illustrated in Fig. 2.

Designed to accommodate both pose representation and text inputs, the Spatial Relationship Prediction system incorporates two separate encoders. The first, a spatial encoder, is comprised of a 4-layer Multilayer Perceptron (MLP). For textual data, we utilize a CLIP-txt Encoder. Each of these encoders outputs a vector of 512 dimensions. These vectors are subsequently concatenated and supplied to the distance predictor.

Throughout the training process, we input a pair of poses along with a positive language label and a negative language label. A contrastive loss is implemented to differentiate between the distances of the positive and negative samples.

The positive and negative samples are produced via a synthetic pipeline, for which we have predefined 90 distinct spatial relationship descriptions. To augment the language labels during training, we employ a random drop technique and introduce noise.

D Performance of the OVSG Framework on Various Scenes in DOVE-G

In Table 7, we present the performance of our OVSG framework on natural language scene queries in DOVE-G.

E Manipulation Experiment Setup

Robot Setup All evaluations were conducted using a Kuka IIWA 14 robot arm equipped with a Robotiq 3-finger adaptive gripper. The arm was augmented with an Intel Realsense D435 camera, which was utilized to capture the depth and color information of the scene in an RGB-D format, offering a resolution of 1280 x 720. The gripper operated in "Pinch Mode," whereby the two fingers on the same side of the gripper bent inward.

To initiate the process, the robot arm was employed to position the camera above the table, orienting it in a downward direction. Subsequently, the RGB-D data, along with a query specifying the object to be picked and a target object for placement, were inputted into the OVSG system.

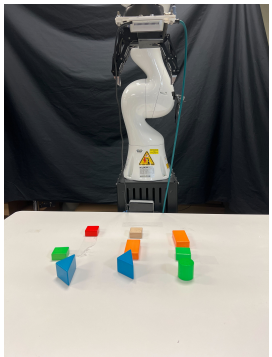

Figure 3: ROSMASTER R2 Ackermann Steering Robot.

Scene	Top1 Match Rate				Top3 Match Rate			
	OVIR-3D	OVSG-J	OVSG-S	OVSG-L	OVIR-3D	OVSG-J	OVSG-S	OVSG-L
Room #1	44.4	36.8	61.2	**79.6**	88.4	72.8	98.4	**99.6**
Kitchenette	31.0	24.0	36.0	**49.0**	65.0	51.0	71.0	**71.0**
Bathroom	44.8	23.2	57.2	**58.0**	59.6	41.2	66.0	**69.6**
Kitchen	44.8	36.0	51.2	**55.6**	53.6	50.8	55.2	**56.8**
Room #2	18.0	26.0	40.0	**41.6**	62.0	48.0	**65.2**	62.8
Computer Lab	38.0	30.8	38.0	**41.6**	**54**	41.2	50.4	49.2
Room #3	32.8	16.8	40.4	**40.4**	**47.2**	34.4	**47.2**	**47.2**
Hallway	28.0	8.4	28.4	**33.2**	40.0	34.4	40.0	**40.0**
Overall	35.2	25.2	45.4	**51.1**	63.6	47.1	64.8	**65.7**
	Top1 IoU				Top3 IoU			
	OVIR-3D	OVSG-J	OVSG-S	OVSG-L	OVIR-3D	OVSG-J	OVSG-S	OVSG-L
Room #1	0.26	0.18	0.36	**0.47**	0.55	0.39	0.60	**0.61**
Kitchenette	0.28	0.23	0.34	**0.42**	0.59	0.47	0.62	**0.62**
Bathroom	0.30	0.13	0.36	**0.36**	0.41	0.30	0.44	**0.46**
Kitchen	0.35	0.28	0.38	**0.42**	0.47	0.42	0.46	**0.48**
Room #2	0.15	0.21	0.31	**0.34**	0.51	0.41	**0.53**	0.52
Computer Lab	0.32	0.22	0.33	**0.39**	**0.49**	0.39	0.45	0.46
Room #3	0.34	0.13	0.37	**0.39**	0.50	0.36	0.49	**0.50**
Hallway	0.27	0.10	0.24	**0.31**	0.37	0.34	0.38	**0.41**
Overall	0.28	0.18	0.34	**0.39**	0.49	0.39	0.49	**0.50**

Table 7: Performance of the OVSG framework on natural language scene queries in DOVE-G

Upon acquiring the bounding box of the query object, the robot gripper was directed to move towards the center coordinates of the target box by utilizing the ROS interface of the robot arm.

Block building task To evaluate the application of the proposed method in real-world manipulation tasks, we designed a block-building task. The task is to pick one building block from a set of building blocks and place it on another building block. The picking block and placing block are separately specified by a different natural language query. The difficulty of this task is that each building block has many repeats around it so that we have to use spatial context to specify the building block. And we need to succeed twice in a row to complete a task.

F Navigation Experiment Setup

Robot Setup All evaluations were conducted using a ROSMASTER R2 Ackermann Steering Robot. For perception, we utilized an Astra Pro Plus Depth Camera and a YDLidar TG 2D lidar sensor, both mounted directly onto the robot. The robot is equipped with a built-in Inertial Measurement Unit (IMU) and wheel encoder. The Astra camera provides a video stream at a resolution of 720p at 30 frames per second, and the lidar operates with a sampling frequency of 2000 Hz and a scanning radius of approximately 30 meters. The overall configuration of the setup is depicted in Figure. 4.

Demonstrations and Execution Prior to the evaluation process, we employed an Intel RealSense D455 camera and ORB-SLAM3 to generate a comprehensive map of the environment. This generated both the RGB-D and pose data, which could be subsequently fed into the Open-vocabulary pipeline. For the demonstration of locating with the Open-Vocabulary 3D Scene Graph (OVSG), we developed a 3D to 2D conversion tool. This tool takes the point cloud from the comprehensive 3D map and converts it into a 2D map by selecting a layer of points at the height of the lidar. The resultant 2D map could then be utilized by the ROSMASTER R2 Ackermann Steering robot for navigation. To achieve goal-oriented navigation, we incorporated the Robot Operating System (ROS) Navigation stack and inte-

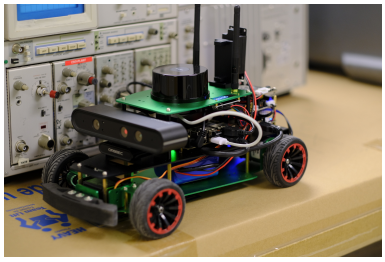

Figure 4: ROSMASTER R2 Ackermann Steering Robot.

Object	Query Id	Query
shoe	1	A shoe that is in front of the monitor and in the office.
	2	A shoe positioned both before the monitor and within the office.
	3	A shoe rests in front of the monitor. The shoe is inside the office.
bottle	4	The bottle that is right to Tom's keyboard
	5	The bottle to the right of Tom's keyboard.
	6	The bottle positioned to the right of a keyboard which belongs to Tom.
chair#1	7	The chair that is behind the TV.
	8	The chair is situated behind the TV
	9	The chair with a TV in front of it.
chair#2	10	The chair that is in front of the car.
	11	The chair rests before a car.
	12	The chair with a car behind it.
trash can#1	13	The trash can that is behind the refrigerator.
	14	The trash can which can be found behind the refrigerator.
	15	The trash can that is situated at the rear of the refrigerator.
trash can#2	16	The trash can that is in the lab and under a table.
	17	The trash can is located in the lab and beneath a table.
	18	The trash can is situated within the lab, positioned under a table.
drawer	19	The drawer that is behind a box in the office.
	20	The drawer is positioned behind a box in the office.
	21	The drawer is situated at the rear of a box within the office.
cloth	22	The cloth that is on a chair
	23	The cloth is resting on a chair
	24	The cloth is positioned atop a chair.

Table 8: Queries for navigation task

grated it with the Timed Elastic Band (TEB) planner. The initial step involved establishing a pose within the environment. Subsequently, the Adaptive Monte Carlo Localization (AMCL) leveraged lidar scan inputs and IMU data to provide a robust estimate of the robot's pose within the map. The move base node, a key component of the ROS navigation stack, used the converted map and the item's position provided by the OVSG and conversion tool to formulate a comprehensive global plan targeting the goal position. Concurrently, the TEB local planner consolidated information about ROSMASTER R2's kinematics and lidar input to generate a short-term trajectory. The result was a locally optimized, time-efficient plan that adhered to the robot's pre-set velocity and acceleration limits. The plan also included obstacle avoidance capabilities, enabling the robot to identify and circumvent barriers detected by the lidar system.

Object navigation task To evaluate the application of OVSG in real-world navigation problems, a language-based object navigation task is proposed. We selected seven different objects inside a laboratory. Each object is paired with three different queries. All queries for three objects are listed in Table. 8.

G 50 Sample Natural Language Queries for Scenes in DOVE-G

In Table 9, we provide a list of 50 sample queries for scenes in DOVE-G.

H More on Scenes in DOVE-G

In Figure 5 and Figure 6, we display eight different scenes included in our DOVE-G dataset.

Query No.	Natural Language Query
1	Locate the vanity sink, which is positioned to the right side of a door latch.
2	Identify the hand basin that has a face cleanser situated in front of it.
3	Is there a hand basin that has both a facial scrub and hand soap placed in front of it?
4	I'm looking for a wash-hand basin with a facial scrub directly in front of it and a door latch to its right.
5	Locate the shower jet that Nami loves, with a mirrored surface to its right and a hair cleanser in front of it.
6	Can you find the shower sprayer with a face cleanser positioned behind it?
7	Search for the shower sprayer that has a facial scrub behind it and a vanity mirror to its right.
8	Identify the travel suitcase located to the right of a backpack and ahead of a water bottle.
9	Look for a travel suitcase Zoro dislikes, it should be to the right of a book and a water bottle.
10	Can you find a book positioned to the left of a backpack?
11	Nami's preferred book should be positioned in front of a chair, can you find it?
12	Can you identify Zoro's liked book that's situated ahead of a water bottle and another book?
13	Locate a Carry-on Luggage for me, please. It should have both a water bottle and a rucksack in front.
14	Is there a trolley bag with a water bottle and a backpack up front, and also a desk chair in its rear?
15	Find an ergonomic chair for me, but it has to have a textbook situated to its left.
16	Can you find the workbook that Luffy dislikes and is right of a desk chair?
17	Is there a headrest that has a reference book positioned in front of it?
18	Where's the cushion with a coursebook and an ergonomic chair up front?
19	I'm searching for a backpack with a coursebook on its left.
20	Where's the table fan with a desk chair behind it?
21	Can you find a table fan with a computer chair and a reference book behind it?
22	Where's the reading lamp that's to the left of a travel bag?
23	Can you spot the reading lamp that's to the left of a travel bag and behind a computer chair?
24	Is there a reading lamp that's behind a reference book?
25	Can you find a pedestal fan with a desk chair and a coursebook behind it?
26	Locate the headrest that's disliked by Nami, but Luffy is indifferent to.
27	How about a cushion that Nami dislikes, Luffy is neutral to, and Zoro takes a liking to?
28	Where's the reading lamp that's to the left of a knapsack?
29	Can you spot the reading lamp that's to the left of a knapsack and behind an ergonomic chair?
30	Is there a desk lamp that's behind a workbook?
31	Where's the reading lamp that Nami is fond of?
32	Can you locate a backpack with a table fan to its left?
33	Where's the knapsack with a tower fan on its left and Luffy behind?
34	Can you find a travel bag that has a water bottle on its left?
35	I'm searching for a backpack with a textbook on its left.
36	Where's the pedestal fan with a computer chair behind it?
37	Where is the cup that's nestled to the right of the coffee maker, to the left of the coffee kettle, and in front of the poster?
38	Identify the toy that's to the right of the espresso machine and to the left of the trash can.
39	Where's the doll with a cup positioned behind it?
40	Can you show me the water bottle that Luffy loves and has a coffee cup behind it?
41	Locate the water bottle that's to the left of the checkerboard.
42	I want to know about the coffee cup that Zoro loves, which is also behind the keyboard that Nami is behind.
43	Can you identify the chair that the CPU machine is behind?
44	Locate the chair that Nami likes and is also behind the CPU machine.
45	Can you identify the teacup that Luffy loves and is behind the CPU machine?
46	Is there a computer chair that Zoro doesn't prefer?
47	Identify a coursebook located ahead of a water bottle.
48	Is there a workbook sandwiched between two water sipper bottles?
49	Nami's preferred coursebook should be positioned in front of a desk chair, can you find it?
50	Zoro's liked reading book, is it placed in front of a water beverage bottle?

Table 9: List of 50 sample queries for scenes in DOVE-G

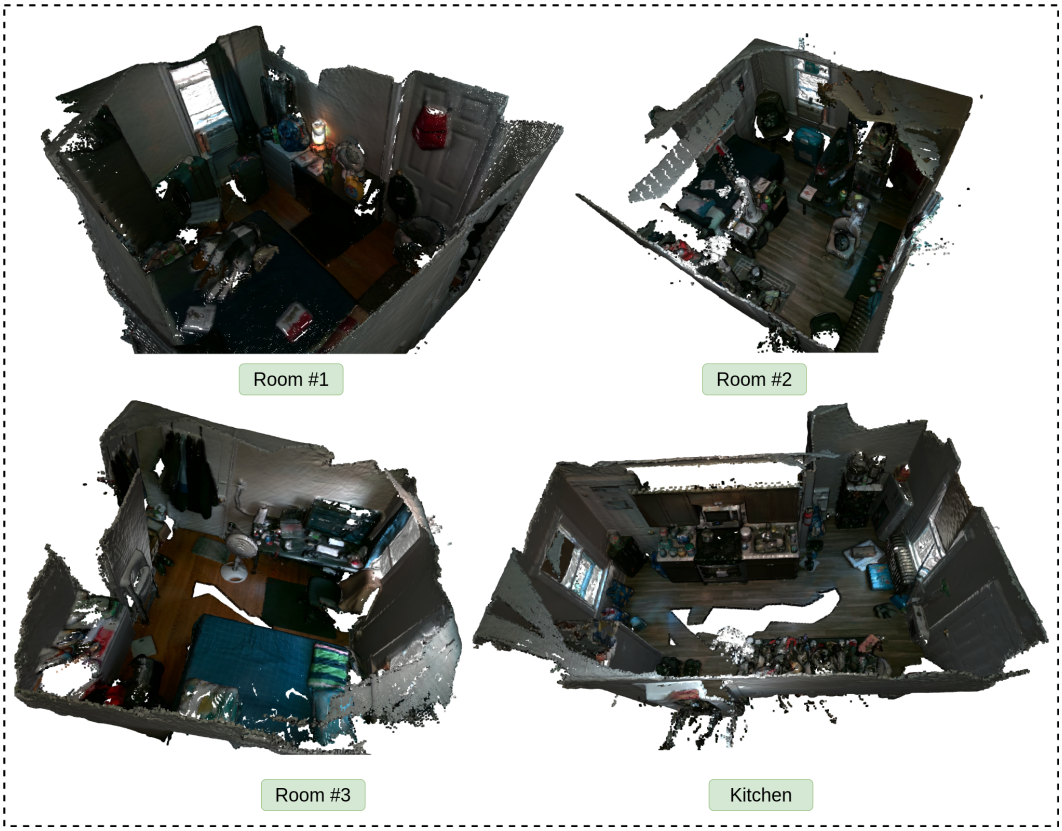

Figure 5: Four of the scenes in DOVE-G

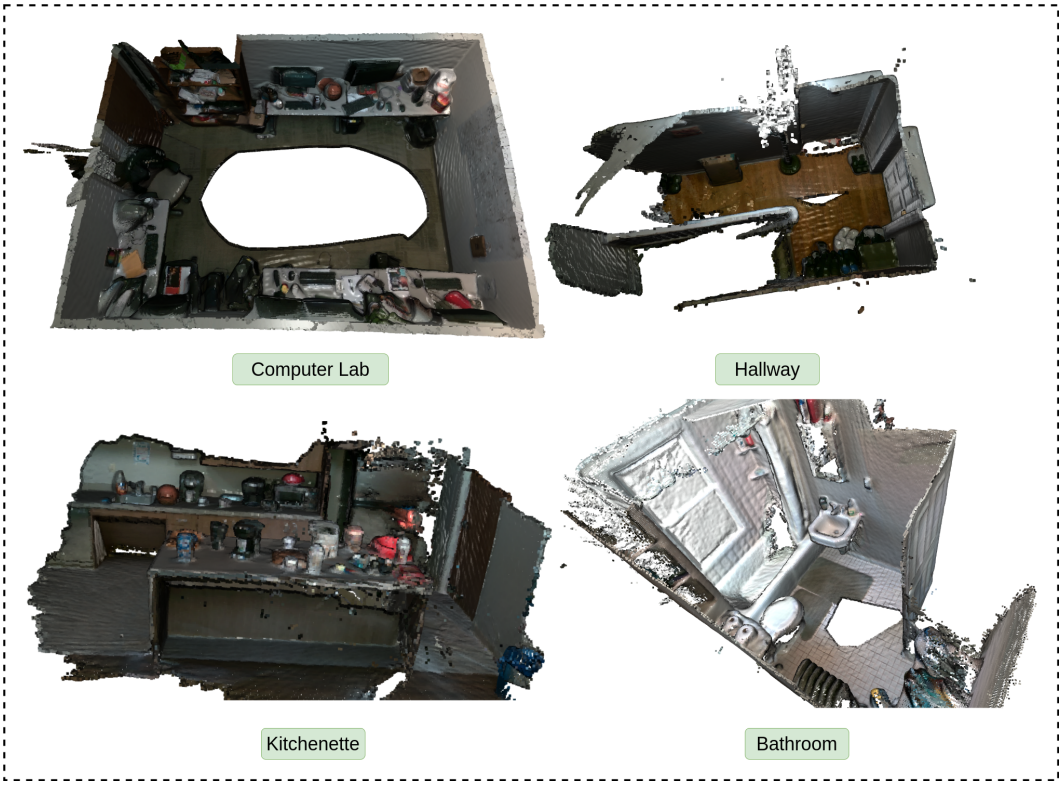

Figure 6: Four of the other scenes in DOVE-G