**ConceptGraphs**: Open-Vocabulary 3D Scene Graphs for Perception and Planning

https://concept-graphs.github.io/

Qiao Gu†*1, Ali Kuwajerwala†*2, Sacha Morin*2, Krishna Murthy Jatavallabhula*3, Bipasha Sen2, Aditya Agarwal2, Corban Rivera5, William Paul5, Kirsty Ellis2, Rama Chellappa6, Chuang Gan7, Celso Miguel de Melo4, Joshua B. Tenenbaum3, Antonio Torralba3, Florian Shkur11, Liam Paul12, 1University of Toronto, 2Université de Montréal, 3MIT, 4DEVCOM ARL, 5JHU APL, 6JHU, 7UMass

Abstract—For robots to perform a wide variety of tasks, they require a 3D representation of the world that is semantically rich, yet compact and efficient for task-driven perception and planning. Recent approaches have attempted to leverage features from large vision-language models to encode semantics in 3D representations. However, these approaches tend to produce maps with per-point feature vectors, which do not scale well in larger environments, nor do they contain semantic spatial relationships between entities in the environment, which are useful for downstream planning. In this work, we propose ConceptGraphs, an open-vocabulary graph-structured representation for 3D scenes. ConceptGraphs is built by leveraging 2D foundation models and fusing their output to 3D by multi-view association. The resulting representations generalize to novel semantic classes, without the need to collect large 3D datasets or finetune models. We demonstrate the utility of this representation through a number of downstream planning tasks that are specified through abstract (language) prompts and require complex reasoning over spatial and semantic concepts. To explore the full scope of our experiments and results, we encourage readers to visit our project webpage.

†Project Lead *Equal Contribution

I. INTRODUCTION

Scene representation is one of the key design choices that can facilitate downstream planning for a variety of tasks, including mobility and manipulation. Robots need to build these representations online from onboard sensors as they navigate through an environment. For efficient execution of complex tasks such representations should be: scalable and efficient to maintain, as the volume of the scene and the duration of the robot’s operation increases; open-vocabulary, not limited to making inferences about a set of concepts that is predefined at training time; and with a flexible level of detail to enable planning over a range of tasks, from ones that require dense geometric information for mobility and manipulation, to ones that need abstract semantic information and object-level affordance information for task planning. We propose ConceptGraphs, a 3D scene representation method for robot perception and planning that satisfies all the above requirements.
A. Related Work

Closed-vocabulary semantic mapping in 3D. Early works reconstruct the 3D map through online algorithms like simultaneous localization and mapping (SLAM) [1]–[5] or offline methods like structure-from-motion (SfM) [6], [7]. Aside from reconstructing 3D geometry, recent works also use deep learning-based object detection and segmentation models to reconstruct the 3D scene representations with dense semantic mapping [8]–[11] or object-level decomposition [12]–[15]. While these methods achieve impressive results in mapping semantic information to 3D, they are closed-vocabulary and their applicability is limited to object categories annotated in their training datasets.

3D scene representations using foundation models. There have been significant recent efforts [16]–[30] focused on building 3D representations by leveraging foundation models - large, powerful models that capture a diverse set of concepts and accomplish a wide range of tasks [31]–[35]. Such models have excelled in tackling open-vocabulary challenges in 2D vision. However, they require an “internet-scale” of training data, and no 3D datasets exist yet of a comparable size. Recent works have therefore attempted to ground the 2D representations produced by image and language foundation models to the 3D world and show impressive results on open-vocabulary tasks, including language-guided object grounding [17], [18], [24], [26], [36], 3D reasoning [37], [38], robot manipulation [39], [40] and navigation [41], [42]. These approaches project dense per-pixel features from images to 3D to build explicit representations such as pointclouds [17]–[21] or implicit neural representations [16], [22]–[30].

However, such methods have two key limitations. First, assigning every point a semantic feature vector is highly redundant and consumes more memory than necessary, greatly limiting scalability to large scenes. Second, these dense representations do not admit an easy decomposition – this lack of structure makes them less amenable to dynamic updates to the map (crucial for robotics).

3D scene graphs. 3D scene graphs (3DSGs) address the second limitation by compactly and efficiently describing scenes with graph structures, with nodes representing objects and edges encoding inter-object relationships [43]–[47]. These approaches have enabled building real-time systems that can dynamically build up hierarchical 3D scene representations [48]–[50], and more recently shown that various robotics planning tasks can benefit from efficiency and compactness of 3DSGs [51], [52]. However, existing work on building 3D scene graphs has been confined to the closed-vocabulary setting, limiting their applicability to a small set of tasks.

B. Overview of Our Contribution

In this work, we mitigate all the aforementioned limitations and propose ConceptGraphs, an open-vocabulary and object-centric 3D representation for robot perception and planning. In ConceptGraphs, each object is represented as a node with geometric and semantic features, and relationships among objects are encoded in the graph edges. At the core of ConceptGraphs is an object-centric 3D mapping technique that integrates geometric cues from conventional 3D mapping systems, and semantic cues from vision and language foundation models [31], [33], [34], [53]–[55]. Objects are assigned language tags by leveraging large language models (LLMs) [32] and large vision-language models (LVLMs) [55], which provide semantically rich descriptions and enable free-form language querying, all while using off-the-shelf models (no training/finetuning). The scene graph structure allows us to efficiently represent large scenes with a low memory footprint and makes for efficient task planning.

In experiments, we demonstrate that ConceptGraphs is able to discover, map, and caption a large number of objects in a scene. Further, we conduct real-world trials on multiple robot platforms over a wide range of downstream tasks, including manipulation, navigation, localization, and map updates. To summarize, our key contributions are:

- We propose a novel object-centric mapping system that integrates geometric cues from traditional 3D mapping systems and semantic cues from 2D foundation models.
- We construct open-vocabulary 3D scene graphs; efficient and structured semantic abstractions for perception and planning.
- We implement ConceptGraphs on real-world wheeled and legged robotic platforms and demonstrate a number of downstream perception and planning capabilities for complex/abstract language queries.

II. METHOD

ConceptGraphs builds a compact, semantically rich representation of a 3D environment. Given a set of posed RGB-D frames, we run a class-agnostic segmentation model to obtain candidate objects, associate them across multiple views using geometric and semantic similarity measures, and instantiate nodes in a 3D scene graph. We then use an LVLM to caption each node and an LLM to infer relationships between adjoining nodes, which results in edges in the scene graph. This resultant scene graph is open-vocabulary, encapsulates object properties, and can be used for a multitude of downstream tasks including segmentation, object grounding, navigation, manipulation, localization, and remapping. The approach is illustrated in Fig. 2.

A. Object-based 3D Mapping

Object-centric 3D representation: Given a sequence of RGB-D observations \( \mathcal{T} = \{I_1, I_2, \ldots, I_T\} \), ConceptGraphs constructs a map, a 3D scene graph, \( \mathcal{M}_t = (\mathcal{O}_t, \mathcal{E}_t) \), where \( \mathcal{O}_t = \{o_j\}_{j=1}^{J} \) and \( \mathcal{E}_t = \{e_k\}_{k=1}^{K} \) represent the sets of objects and edges, respectively. Each object \( o_j \) is characterized by a 3D point cloud \( p_{o_j} \) and a semantic feature vector \( f_{o_j} \). This map is built incrementally, incorporating each incoming frame \( I_t = (I^{rgb}_t, I^{depth}_t, \theta_t) \) (color image, depth image, pose) into the existing object set \( \mathcal{O}_{t-1} \), by either adding to existing objects or instantiating new ones.

Class-agnostic 2D Segmentation: When processing frame \( I_t \), a class-agnostic segmentation model Seg(\( \cdot \)...)
Fig. 2: ConceptGraphs builds an open-vocabulary 3D scene graph from a sequence of posed RGB-D images. We use generic instance segmentation models to segment regions from RGB images, extract semantic feature vectors for each, and project them to a 3D point cloud. These regions are incrementally associated and fused from multiple views, resulting in a set of 3D objects and associated vision (and language) descriptors. Then large vision and language models are used to caption each mapped 3D objects and derive inter-object relations, which generates the edges to connect the set of objects and form a graph. The resulting 3D scene graph provides a structured and comprehensive understanding of the scene and can further be easily translated to a text description, useful for LLM-based task planning.

is used to obtain a set of masks \( \{m_{t,i}\}_{i=1,...,M} = \text{Seg}(I_{t}^{\text{rgb}}) \) corresponding to candidate objects\(^1\). Each extracted mask \(m_{t,i}\) is then passed to a visual feature extractor (CLIP [31], DINO [53]) to obtain a visual descriptor \(f_{t,i} = \text{Embed}(I_{t}^{\text{rgb}}, m_{t,i})\). Each masked region is projected to 3D, denoised using DBSCAN clustering, and transformed to the map frame. This results in a pointcloud \(p_{t,i}\) and its corresponding unit-normalized semantic feature vector \(f_{t,i}\).

**Object Association:** For every newly detected object \(p_{t,i}\), we compute semantic and geometric similarity with respect to all objects \(o_{t-1,j} = (p_{t-1,j}, f_{t-1,j})\) in the map that shares any partial geometric overlap. The geometric similarity \(\phi_{\text{geo}}(i,j) = \text{nnratio}(p_{t,i}, p_{t-1,j})\) is the proportion of points in point cloud \(p_{t,i}\) that have nearest neighbors in point cloud \(p_{t-1,j}\) within a distance threshold of \(\delta_{\text{m}}\). The semantic similarity \(\phi_{\text{sem}}(i,j) = \frac{f_{t,i} \cdot f_{t-1,j}}{\|f_{t,i}\|_2 \|f_{t-1,j}\|_2 + 1/2}\) is the normalized cosine distance between the corresponding visual descriptors.\(^2\) The overall similarity measure \(\phi(i,j)\) is a sum of both: \(\phi(i,j) = \phi_{\text{sem}}(i,j) + \phi_{\text{geo}}(i,j)\). We perform object association by a greedy assignment\(^3\) strategy where each new detection is matched with an existing object with the highest similarity score. If no match is found with a similarity higher than \(\phi_{\text{sim}}\), we initialize a new object.

**Object Fusion:** If a detection \(o_{t-1,j}\) is associated with a mapped object \(o_{j}\), we fuse the detection with the map. This is achieved by updating the object semantic feature as \(f_{o_j} = (n_{o_j} f_{o_j} + f_{t,i})/(n_{o_j}+1)\), where \(n_{o_j}\) is the number of detections that have been associated to \(o_{j}\) so far; and updating the pointcloud as \(p_{t,i} \cup p_{o_j}\), followed by down-sampling to remove redundant points.

**Node Captioning:** Once the entire image sequence has been processed, a vision-language model, denoted LLM(\(\cdot\)), is used to generate object captions. For each object, the associated image crops from the \(best^4\) 10 views are passed to the language model with the prompt “describe the central object in the image” to generate a set of initial rough captions \(\hat{c}_j = \{\hat{c}_{j,1}, \hat{c}_{j,2}, \ldots, \hat{c}_{j,10}\}\) for each detected object \(o_{j}\). Each set of captions is then refined to the final caption by passing \(\hat{c}_j\) to another language model LLM(\(\cdot\)) with a prompt instruction to summarize the initial captions into a coherent and accurate final caption \(c_j\).

**B. Scene Graph Generation**

Given the set of 3D objects \(O_T\) obtained from the previous step, we estimate their spatial relationships, i.e., the edges \(E_T\), to complete the 3D scene graph. We do this by first estimating potential connectivity among object nodes based on their spatial overlaps. We compute the 3D bounding box IoU between every pair of object nodes to obtain a similarity matrix (i.e., a dense graph), which we prune by estimating a minimum spanning tree (MST), resulting in a refined set of potential edges among the objects. To further determine the semantic relationships, for each edge in the MST, we input the information about the object pair, consisting of object captions and 3D location, to a language model LLM. The prompt instructs the model to describe the likely spatial relationship between the objects, such as “a on b” or “b in a”, along with the underlying reasoning. The model outputs a relationship label with an explanation detailing the rationale.

\(^1\)Without loss of generality, Seg(\(\cdot\)) may be replaced by open-/closed-vocabulary models to build category-specific matching systems.

\(^2\)For the sake of brevity, we only describe the best-performing geometric and semantic similarity measures. For an exhaustive list of alternatives, please see our project website and code.

\(^3\)While we also experimented with optimal assignment strategies such as the Hungarian algorithm, we experimentally determined them to be slower and offer only a minuscule improvement over greedy association.

\(^4\)We maintain a running index of the number of noise-free points each view contributes to the object point cloud.
The use of an LLM allows us to extend the nominal edge type defined above to other output relationships a language model can interpret, such as “a backpack may be stored in a closet” and “sheets of paper may be recycled in a trash can”. This results in an open-vocabulary 3D scene graph $\mathcal{M}_T = (\mathcal{O}_T, \mathcal{E}_T)$, a compact and efficient representation for use in downstream tasks.

C. Robotic Task Planning through LLMs

To enable users to carry out tasks described in natural language queries, we interface the scene graph $\mathcal{M}_T$ with an LLM. For each object in $\mathcal{O}_T$, we construct JSON-structured text descriptions that include information about its 3D location (bounding box) and its node caption. Given a text query, we task the LLM to identify the most relevant object in the scene. We then pass the 3D pose of this object to the appropriate pipeline for the downstream task (e.g., grasping, navigation). This integration of ConceptGraphs with an LLM is easy to implement, and enables a wide range of open-vocabulary tasks by giving robots access to the semantic properties of surrounding objects\(^5\) (see Sec. III).

D. Implementation Details

The modularity of ConceptGraphs enables any appropriate open/closed-vocabulary segmentation model, LLM, or LVLM to be employed. Our experiments use Segment-Anything (SAM) [33] as the segmentation model Seg(·), and the CLIP image encoder [31] as the feature extractor Embed(·). We use LLaVA [55] as the vision-language model LVLM and GPT-4 [32] (gpt-

\(4\)·0613) for our LLM. The voxel size for point cloud downsampling and nearest neighbor threshold $\delta_{\text{nn}}$ are both 2.5cm. We use 1.1 for the association threshold $\delta_{\text{assoc}}$.

We also develop a variant of our system, ConceptGraphs-Detector (CG-D), where we employ an image tagging model (RAM [54]) to list the object classes present in the image and an open-vocabulary 2D detector (Grounding DINO [34]) to obtain object bounding boxes\(^6\). In this variant, we need to separately handle detected background objects (wall, ceiling, floor) by merging them regardless of their similarity scores.

III. EXPERIMENTS

A. Scene Graph Construction

We first evaluate the accuracy of the 3D scene graphs output by the ConceptGraphs system in Table I. For each scene in the Replica dataset [56], we report scene graph accuracy metrics for both CG and the detector-variant CG-D. The open-vocabulary nature of our system makes automated evaluation of the quality of nodes and edges in the scene graph challenging. We instead evaluate the scene graph by engaging human evaluators on Amazon Mechanical Turk (AMT). For each node, we compute precision as the fraction of nodes for which at least 2 of 3 human evaluators deem the node caption correct. We also report the number of valid objects retrieved by each variant by asking evaluators whether they deem each node a valid object. Both CG and CG-D identify a number of valid objects in each scene, and incur only a small number (0-5) of duplicate detections. The node labels are accurate about 70% of the time; most of the errors are incurred due to errors made by the LVLM employed (LLaVA [55]). The edges (spatial relationships) are labeled with a high degree of accuracy (90% on average).

B. 3D Semantic Segmentation

ConceptGraphs focuses on the construction of the open-vocabulary 3D scene graphs for scene understanding and planning. For completeness, in this section, we also use an open-vocabulary 3D semantic segmentation task to evaluate the quality of the obtained 3D maps. To generate the semantic segmentation, given a set of class names, we compute the similarity between the fused semantic feature of each object node and the CLIP text embeddings of the phrase \textit{an image of \{class\}}. Then the points associated with each object are assigned to the class with the highest similarity, which gives a point cloud with dense class labels. In Table II, we report the semantic segmentation results on the Replica [56] dataset, following the evaluation protocol used in ConceptFusion [17]. We also provide an additional baseline, ConceptFusion+SAM, by replacing the Mask2Former used in ConceptFusion with the more performant SAM [33] model. As shown in Table II, the proposed ConceptGraphs performs comparably with or better than ConceptFusion, which has a much larger memory footprint.

C. Object Retrieval based on Text Queries

We assess the capability of ConceptGraphs to handle complex semantic queries, focusing on three key types.

- Descriptive: E.g., \textit{A potted plant}.

\(^5\)For large scenes where the description length of the scene graph exceeds the context length of the LLM, one can easily substitute in alternative (concurrent) LLM planners [52].

\(^6\)We discard the (often noisy) tags produced by the image tagging model, relying instead on our node captions.

<table>
<thead>
<tr>
<th>scene</th>
<th>node prec.</th>
<th>valid objects</th>
<th>duplicates</th>
<th>edge prec.</th>
</tr>
</thead>
<tbody>
<tr>
<td>room0</td>
<td>0.78</td>
<td>54</td>
<td>3</td>
<td>0.91</td>
</tr>
<tr>
<td>room1</td>
<td>0.77</td>
<td>43</td>
<td>4</td>
<td>0.93</td>
</tr>
<tr>
<td>room2</td>
<td>0.66</td>
<td>47</td>
<td>4</td>
<td>1.0</td>
</tr>
<tr>
<td>office0</td>
<td>0.65</td>
<td>44</td>
<td>2</td>
<td>0.88</td>
</tr>
<tr>
<td>office1</td>
<td>0.65</td>
<td>23</td>
<td>0</td>
<td>0.9</td>
</tr>
<tr>
<td>office2</td>
<td>0.75</td>
<td>44</td>
<td>3</td>
<td>0.82</td>
</tr>
<tr>
<td>office3</td>
<td>0.68</td>
<td>60</td>
<td>5</td>
<td>0.79</td>
</tr>
</tbody>
</table>

| Average| 0.71        | -             | -          | 0.88       |

\[\begin{array}{l|l|l|l|l}
 \text{CG} & \text{Average} & \text{node prec.} & \text{valid objects} & \text{ duplicates} & \text{edge prec.} \\
\hline
 \text{room0} & 0.56 & 60 & 4 & 0.87 \\
 \text{room1} & 0.70 & 40 & 3 & 0.93 \\
 \text{room2} & 0.54 & 49 & 2 & 0.93 \\
 \text{office0} & 0.59 & 35 & 0 & 1.0 \\
 \text{office1} & 0.49 & 24 & 2 & 0.9 \\
 \text{office2} & 0.67 & 47 & 3 & 0.88 \\
 \text{office3} & 0.71 & 59 & 1 & 0.83 \\
 \hline
 \text{CG-D} & \text{Average} & \text{node prec.} & \text{valid objects} & \text{ duplicates} & \text{edge prec.} \\
\end{array}\]

\[\text{node precision indicates the accuracy of the label for each node (as measured by a human evaluator)}; \text{valid objects is the number of human-recognizable objects (mturkers used) discovered by our system}; \text{duplicates are the number of redundant detections}; \text{edge precision indicates the accuracy of each estimated spatial relationship (again, as evaluated by an mturker)}\]

<table>
<thead>
<tr>
<th>Method</th>
<th>mAcc</th>
<th>F-mIoU</th>
</tr>
</thead>
<tbody>
<tr>
<td>Privileged</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CLIPSeg (rd64-uni) [57]</td>
<td>28.21</td>
<td>39.84</td>
</tr>
<tr>
<td>LSeg [58]</td>
<td>33.39</td>
<td>51.54</td>
</tr>
<tr>
<td>OpenSeg [59]</td>
<td>41.19</td>
<td>53.74</td>
</tr>
<tr>
<td>Zero-shot</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MaskCLIP [60]</td>
<td>4.53</td>
<td>0.94</td>
</tr>
<tr>
<td>Mask2former + Global CLIP</td>
<td>10.42</td>
<td>13.11</td>
</tr>
<tr>
<td>ConceptFusion [17]</td>
<td>24.16</td>
<td>31.31</td>
</tr>
<tr>
<td>ConceptFusion [17] + SAM</td>
<td>31.53</td>
<td>38.70</td>
</tr>
<tr>
<td>ConceptGraphs [Ours]</td>
<td>40.63</td>
<td>35.95</td>
</tr>
<tr>
<td>ConceptGraphs-Detector [Ours]</td>
<td>38.72</td>
<td>35.82</td>
</tr>
</tbody>
</table>

TABLE II: Open-vocabulary semantic segmentation on the Replica [56] dataset. Privileged methods specifically finetune the pretrained models for semantic segmentation. Zero-shot approaches do not need any finetuning and are evaluated off the shelf.

- Affordance: E.g., Something to use for temporarily securing a broken zipper.
- Negation: E.g., Something to drink other than soda.

We evaluate on the Replica dataset [56] and a real-world scan of the REAL Lab, where we staged a number of items including clothes, tools, and toys. Replicas human evaluators on AMT annotate captions for SAM mask proposals, which serve as both ground truth labels and descriptive queries. We created 5 affordance and negation queries for each scene type (office & room) in Replica and 10 queries of each type for the lab scan, ensuring that each query corresponds to at least one relevant object. We manually select relevant objects as ground truth for each query.

We use two object retrieval strategies: CLIP-based and LLM-based. CLIP selects the object with the highest similarity to the query’s embedding, while the LLM goes through the scenegraph nodes to identify the object with the most relevant caption. Table III shows that CLIP excels with descriptive queries but struggles with complex affordance and negation queries [61]. For example, CLIP accurately retrieves a backpack for the broken zipper query, whereas the LLM correctly identifies a roll of tape. The LLM performs well across the board, but is limited by the accuracy of the node captions, as discussed in Section III-A. Since the lab has a larger variety of objects to choose from, the LLM finds compatible objects for complex queries more reliably there.

<table>
<thead>
<tr>
<th>Method</th>
<th>R@1</th>
<th>R@2</th>
<th>R@3</th>
<th># Queries</th>
</tr>
</thead>
<tbody>
<tr>
<td>Detector (Ours)</td>
<td>38.72</td>
<td>35.82</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

TABLE III: Object retrieval from text queries on the Replica and REAL Lab scenes. We measure the top-1, top-2, and top-3 recall. CLIP refers to object retrieval using cosine similarity, whereas LLM refers to having an LLM parse the scene graph and return the most relevant object.

- Affordance: E.g., Something to use for temporarily securing a broken zipper.
- Negation: E.g., Something to drink other than soda.

We evaluate the performance of ConceptGraphs in a real-world environment, where we carry out navigation experiments in the REAL Lab scene with a Clearpath Jackal UGV. The robot is equipped with a VLP-16 LiDAR and a forward-facing Realsense D435i camera.

The Jackal needs to respond to abstract user queries and navigate to the most relevant object (Figure 1). By using an LVLM [55] to add a description of the current camera image to the text prompt, the robot can also answer visual queries. For example, when shown a picture of Michael Jordan and prompted with something this guy would play with, the robot finds a basketball.

E. Object Search and Traversability Estimation

In this section, we showcase how the interaction between the ConceptGraphs representation and an LLM can enable a mobile robot to access a vast knowledge base of everyday objects. Specifically, we prompt an LLM to infer two additional object properties from ConceptGraphs captions: i) the location where a given object is typically found, and ii) if the object can be safely pushed or traversed by the Jackal robot. We design two tasks around the LLM predictions.

Object search: The robot receives an abstract user query and must navigate to the most relevant object in the ConceptGraphs map. Using an LVLM [55], the robot then checks if the object is at the expected location. If not, it queries an LLM to find a new plausible location given the captions of the other objects in the representation. In our prompt, we nudge the LLM to consider typical containers or storage locations. We illustrate two such queries where the target object is moved in Figure 3.

Traversability estimation: As shown in Figure 4, we design a real-world scenario where the robot finds itself enclaved by objects. In this scenario, the robot must push around multiple objects and create a path to the goal state. While traversability can be learned through experience [62], we show that grounding LLM knowledge in a 3D map can grant similar capabilities to robotic agents.

F. Open-Vocabulary Pick and Place

To illustrate how ConceptGraphs can act as the perception backbone for open-vocabulary mobile manipulation, we conducted a series of experiments with a Boston Dynamics Spot Arm robot. Using an onboard RGBD camera and a ConceptGraphs representation of the scene, the Spot robot responds to the query cuddly quacker by grabbing a duck plush toy and placing it in a nearby box (Figure 1). In the supplementary video, Spot completes a similar grasping maneuver with a mango when prompted with the query something healthy to eat.

G. Localization and Map Updates

ConceptGraphs can also be used for object-based localization and map updates. We showcase this with a 3-DoF (x, y, and yaw) localization and remapping task in the AI2Thor [63], [64] simulation environment, where a
Fig. 3: A Jackal robot answering user queries using the ConceptGraphs representation of a lab environment. We first query an LLM to identify the most relevant object given the user query, then validate with an LVLM if the target object is at the expected location. If not, we query the LLM again to find a likely location or container for the missing object. (Blue) When prompted with something to wear for a space party, the Jackal attempts to find a grey shirt with a NASA logo. After failing to detect the shirt at the expected location, the LLM reasons that it could likely be in the laundry bag. (Red) The Jackal searches for red and white sneakers after receiving the user query footwear for a Ronald McDonald outfit. The LLM redirects the robot to a shoe rack after failing to detect the sneakers where they initially appeared on the map.

Fig. 4: The Jackal robot solving a traversability challenge. All paths to the goal are obstructed by objects. We query an LLM to identify which objects can be safely pushed or traversed by the robot (green) and which objects would be too heavy or hinder the robot’s movement (red). The LLM relies on the ConceptGraphs node captions to make traversability predictions and we add the non-traversable objects to the Jackal costmap for path planning. The Jackal successfully reaches the goal by going through a curtain and pushing a basketball, while also avoiding contact with bricks, an iron dumbbell, and a flower pot.

mobile robot uses a particle filter to localize in a pre-built ConceptGraphs map of the environment. During the observation update step of particle filtering, the robot’s detections are matched against the objects in the map based on the hypothesized pose, in a similar way as described in Section II-A. The matching results are aggregated into a single observation score for weighting the pose hypothesis. During this process, previously observed objects are removed if they are not observed by the robot and new objects can also be added. We provide a demonstration of this localization and map updating approach in the supplementary video material.

H. Limitations
Despite its impressive performance, ConceptGraphs has failure modes that remain to be addressed in future work. First, node captioning incurs errors due to the current limitations of LVLMs like LLaVA [55]. Second, our 3D scene graph occasionally misses small or thin objects and makes duplicate detections. This impacts downstream planning, particularly when the incorrect detection is crucial to planning success. Additionally, the computational and economic costs of our system include multiple LVLM (LLaVA [55]) and one or more proprietary LLM inference(s) when building and querying the scene graph, which may be significant.

IV. CONCURRENT WORK

We briefly review recent and unpublished pre-prints that are exploring themes related to open-vocabulary object-based factorization of 3D scenes. Concurrently to us, [65], [66] have explored open-vocabulary object-based factorization of 3D scenes. Where [65] assumes a pre-built point cloud map of the scene, [66] builds a map on the go. Both approaches associate CLIP descriptors to the reconstruction, resulting in performance comparable to our system’s CLIP variant, which struggles with queries involving complex affordances and negation, as shown in Table III. OGSV [67] is closer to our setting, building an open-vocabulary 3D scene graph from RGB-D images. However, [67] relies on a (closed-set) graph neural network to predict object relationships; whereas ConceptGraphs relies on the capabilities of modern LLMs, eliminating the need to train an object relation model.

V. CONCLUSION

In this paper, we introduced ConceptGraphs, a novel approach to open-vocab object-centric 3D scene representation that addresses key limitations in the existing landscape of dense and implicit representations. Through effective integration of foundational 2D models, ConceptGraphs significantly mitigates memory constraints, provides relational information among objects, and allows for dynamic updates to the scene—three pervasive challenges in current methods. Experimental evidence underscores ConceptGraphs’ robustness and extensibility, highlighting its superiority over existing baselines for a variety of real-world tasks including manipulation and navigation. The versatility of our framework also accommodates a broad range of downstream applications, thereby opening new avenues for innovation in robot perception and planning. Future work may delve into integrating temporal dynamics into the model and assessing its performance in less structured, more challenging environments.
This project was supported in part (KM, AT, JBT) by grants from the Army Research Laboratory (grant W911NF1820218), office of naval research (MURI). FS and LP thank NSERC for funding support. LP acknowledges support from the Canada CIFAR AI Chairs program. All findings, opinions, and conclusions expressed in this manuscript solely reflect the views of the authors; and not that of our research sponsors.

ACKNOWLEDGMENTS

We would like to thank the following individuals for their invaluable contributions:

Qiao Gu, Ali Kuwajerwala, Sacha Morin, and Krishna Murthy for their instrumental role in the development, integration, and deployment of ConceptGraphs. These authors were instrumental in writing the majority of the manuscript.

Qiao spearheaded the implementation of the object-based mapping, localization and map update system, prototyped the object captioning module, and conducted the segmentation experiments.

All wrote initial prototypes of the mapping pipeline, coordinated the real-robot experiments at the REAL Lab, implemented the integration with the LLM planners and object retrieval experiments, and performed a significant amount of the hardware setup.

Sacha was at the forefront of deploying the system on robot navigation, object search, and led the traversability experiment, implementing a variety of crucial robot-side functionalities.

Krishna crafted initial prototypes of the detector-free mapping system, scene graph construction, and vision/language model interfaces, also coordinating the project.

Bipasha Sen and Aditya Agarwal integrated ConceptGraphs with a robotic manipulation platform for open-vocabulary pick and place demonstrations, with Kirsty Ellis assisting them in setting up the manipulation platform.

Corban Rivera and William Paul deployed ConceptGraphs on-board a Spot mini robot, showcasing the mobile manipulation capabilities enabled by our system.

Chuang Gan helped qualitatively evaluate our system against end-to-end learned approaches like 3D-LLM.

Rama Chellappa and Celso de Melo provided advice on real-world demonstrations involving mobile manipulation.

Josh Tenenbaum and Antonio Torralba contributed cognitive science and computer vision perspectives respectively, which shaped the experiments evaluating scene-graph construction accuracy.

Florian Shkurti and Liam Paul provided equal advisory support on this project, contributing to brainstorming and critical review processes throughout, and writing/proofreading significant sections of the paper.

A1. CONTRIBUTION STATEMENT

Qiao Gu, Ali Kuwajerwala, Sacha Morin, and Krishna Murthy were instrumental in the development, integration, and deployment of ConceptGraphs. These authors were responsible for writing the majority of the manuscript.

Qiao spearheaded the implementation of the object-based mapping, localization and map update system, prototyped the object captioning module, and conducted the segmentation experiments.

All wrote initial prototypes of the mapping pipeline, coordinated the real-robot experiments at the REAL Lab, implemented the integration with the LLM planners and object retrieval experiments, and performed a significant amount of the hardware setup.

Sacha was at the forefront of deploying the system on robot navigation, object search, and led the traversability experiment, implementing a variety of crucial robot-side functionalities.

Krishna crafted initial prototypes of the detector-free mapping system, scene graph construction, and vision/language model interfaces, also coordinating the project.

Bipasha Sen and Aditya Agarwal integrated ConceptGraphs with a robotic manipulation platform for open-vocabulary pick and place demonstrations, with Kirsty Ellis assisting them in setting up the manipulation platform.

Corban Rivera and William Paul deployed ConceptGraphs on-board a Spot mini robot, showcasing the mobile manipulation capabilities enabled by our system.

Chuang Gan helped qualitatively evaluate our system against end-to-end learned approaches like 3D-LLM.

Rama Chellappa and Celso de Melo provided advice on real-world demonstrations involving mobile manipulation.

Josh Tenenbaum and Antonio Torralba contributed cognitive science and computer vision perspectives respectively, which shaped the experiments evaluating scene-graph construction accuracy.

Florian Shkurti and Liam Paul provided equal advisory support on this project, contributing to brainstorming and critical review processes throughout, and writing/proofreading significant sections of the paper.

A2. 3D SCENE GRAPH: GENERATING NODE CAPTIONS

Once we build an object-level map of the scene using the methodology described in Sec. II-A, we extract and summarize captions for each object. We first extract upto the 10 most-informative views for each object, by tracking the number of (noise-free) 3D points that each image segment contributes to an object in the map. Intuitively, these views offer the best image views for the object. We run each view through an LVLM, here LLaVA-7B [55], to generate an image caption. We use the same prompt across all images: describe the central object in this image.

We found the captions generated by LLaVA-7B to be incoherent or unreliable across all viewpoints. To alleviate this, we employed GPT-4 as a caption summarizer, to map all of the LLaVA-7B captions to a coherent object tag (or optionally, declare the object as an invalid detection). We use the following GPT-4 system prompt:

Identify and describe objects in scenes. Input and output must be in JSON format. The input field ‘captions’ contains a list of image captions aiming to identify objects. Output ‘summary’ as a concise description of the identified object(s). An object mentioned multiple times is likely accurate. If various objects are repeated and a container/surface is noted such as a shelf or table, assume the (repeated) objects are on that container/surface. For unrelated, non-repeating (or empty) captions, summarize as ‘conflicting captions about [objects]’ and set ‘object_tag’ to ‘invalid’. Output ‘possible_tags’ listing potential object categories. Set ‘object_tag’ as the conclusive identification. Focus on indoor object types, as the input captions are from indoor scans.

Listing 1: GPT-4 system prompt used for caption summarization

A3. LLM PLANNER: IMPLEMENTATION DETAILS

For task planning over 3D scene graphs, we use GPT-4 (gpt-4-0613) with a context length of 8K tokens. We first convert each node in the 3D scene graph into a structured text format (here, a JSON string). Each entry in the JSON list corresponds to one object in the scene, and contains the following attributes:

1) object id: a unique (numerical) object identifier
2) bounding box extents: dimensions of each side of the bounding cuboid
3) bounding box center: centroid of the object bounding cuboid
4) object tag: a brief tag describing the object
5) caption: a one-sentence caption (possibly encoding mode details than present in the object tag

Here is a sample snippet from the scene graph for the room0 scene of the Replica [56] dataset.

```json
{
  id: 2,
  bbox_extent: [2.0, 0.7, 0.6],
  bbox_center: [-0.6, 1.1, -1.2],
  object_tag: wooden dresser or chest of drawers,
  caption: A wooden dresser or chest of drawers,
}
```

We track these statistics throughout the mapping lifecycle; meaning that we do not impose any additional computational overhead to determine the 10 best views per object.

We also prototyped variants of this approach on off-the-shelf LLMs with larger context lengths, such as Claude-2 with a context length of 32K tokens, and found it to work reliably.
Listing 2: Sample text entries in the 3D scene graph

For planning using the above text description of the scene, we instantiate the system prompt for GPT-4 as follows:

```json
{
  "id": 3,
  "bbox_extent": [0.6, 0.5, 0.4],
  "bbox_center": [2.8, -0.4, -0.8],
  "object_tag": "vase",
  "caption": "a white, floral-patterned vase (or possibly a ceramic bowl)
}
```

Listing 3: System prompt for the LLM Planner interface

Here are some example GPT-4 responses on our queries

User: I need a quick gift. Help!

LLM-Planner: {
  "inferred_query": "Find suitable object for a gift."
  "relevant_objects": [0, 6, 7, 8, 15, 23, 29, 31],
  "query_achievable": true,
  "final_relevant_objects": [6],
  "explanation": "The most suitable object for a gift could be the ceramic vase (id 6). Vases are commonly gifted items and this one could potentially be filled with a plant or flower arrangement, making a lovely present."
}

Listing 4: Sample queries and raw outputs from the LLM Planner

A4. EXAMPLE TEXT QUERIES

Here we provide some of the text queries used in our experiments.

Replica Dataset Descriptive Queries: For each room, we randomly selected 20 ground truth annotations collected via Amazon Mechanical Turk (AMT). Here is a sample from room0 and office0.

**office0 Descriptive Queries:**

1) This is a trash can against the wall next to a sofa.
2) A chaise lounge right next to a small table.
3) This is a television.
4) This is a dropped, tiled ceiling in what appears to be a classroom for children.
5) This is a plant and it is next to the screens.
6) This is the back of a chair in front of a screen.
7) A small table in front of a large gray sectional couch.
8) This is an armless chair and it’s opposite a coffee table by the sofa.
9) This is a plug-in and it is on the floor.
10) These are table legs and they are underneath the table.
11) These are chairs and they are next to a table.
12) A diner style table in front of two chairs.
13) These are rocks and they are on the wall.
14) This is the right panel of a lighted display screen.
15) This is a planet and it is on the wall.
16) This is an electronic display screen showing a map, on the wall.
17) This is a couch and it is between a table and the wall.
18) This is a garbage can and it is in front of the wall.
19) This is a rug and it is on the floor.
20) This is a table that is above the floor.

**room0 Descriptive Queries:**

1) This is a pillow and this is on top of a couch.
2) A pillow on top of a white couch.
3) This is a couch and it is under a window.
4) This is a stool and it is on top of a rug.
5) This is a side table under a lamp.
6) This is a ceiling light next to the window.
7) This is an end table and it is below a lamp.
8) These are books and they are on the table.
9) This is a couch and it is in front of the wall.
10) White horizontal blinds in a well lit room.
11) This is a striped throw pillow on the loveseat.
12) The pillow is on top of the chair.
13) This is a window and it is next to a door.
14) This is a hurricane candle and it is on top of a cabinet.
15) This is a vase and it is on top of the table.
16) This is a vent in the ceiling.
17) This is a fish and it is on top of a cabinet.
18) This is a vase and it is on top of the table.
19) This is a trash can against a wall.
20) Two cream colored cushioned chairs with blue pillows adjacent to each other.

**Replica Dataset Affordance Queries for Office Scenes:**
1) Something to watch the news on
2) Something to tell the time
3) Something comfortable to sit on
4) Something to dispose of wastepaper in
5) Something to add light into the room

**Replica Dataset Affordance Queries for Room Scenes:**
1) Somewhere to store decorative cups
2) Something to add light into the room
3) Somewhere to set food for dinner
4) Something I can open with my keys
5) Somewhere to sit upright for a work call

**Replica Dataset Negation Queries for Office Scenes:**
1) Something to sit on other than a chair
2) Something very heavy, unlike a clock
3) Something rigid, unlike a cushion
4) Something small, unlike a couch
5) Something light, unlike a table

**Replica Dataset Negation Queries for Room Scenes:**
1) Something small, unlike a cabinet
2) Something light, unlike a table
3) Something soft, unlike a table
4) Something not transparent, unlike a window
5) Something rigid, unlike a rug

**REAL Lab Scan Descriptive queries:**
1) A pair of red and white sneakers
2) A NASA t-shirt
3) A Rubik’s cube
4) A basketball
5) A toy car
6) A backpack
7) An office chair
8) A pair of headphones
9) A yellow jacket
10) A laundry bag

**REAL Lab Affordance Queries:**
1) Something to use to disassemble or take apart a laptop
2) Something to use for cooling a CPU
3) Something to use for carrying books day to day
4) Something to use for temporarily securing a broken zipper
5) Something to use to help a student understand how a computer works
6) An object that is used in a sport involving rims and nets
7) Something to keep myself from getting distracted by loud noises
8) Something to help explain math proofs to a student
9) Something I can use to protect myself from the harsh winter in Canada
10) Something fun to pass the time with

**REAL Lab Negation Queries:**
1) A toy for someone who dislikes basketball
2) Shoes that you wouldn’t wear to something formal
3) Something to protect me from the rain that’s not an umbrella
4) Shoes that are not red and white
5) Something to make a cape with that’s not green
6) Something to drink other than soda
7) Something to use for exercise other than weights
8) Something to wear unrelated to space or science
9) Something light to store belongings, not a backpack
10) Something to play with that’s not a puzzle or colorful

**A5. Navigation Experiments**

For our navigation experiments with the Jackal robot. Our robot is equipped with a VLP-16 lidar and a forward-facing Realsense D435i camera. We begin by building a point cloud of the REAL Lab using the onboard VLP-16 and Open3d SLAM [68]. The initial Jackal point cloud does not include task-relevant objects and is downprojected to a 2D costmap for navigation using the base Jackal ROS stack.

We then stage two separate scenes with different objects: one for object search and another for traversability estimation. In both cases, we map the scene with an Azure Kinect Camera and rely on RTAB-Map [69] to obtain camera poses and the scene point cloud. We proceed to build a Concept-Graphs representation and register the scene point cloud with the initial Jackal map. For our navigation experiments, we only use the objects $O_T$.

For object search queries, we use the LLM Planner described in Section A3 as part of a simple state machine. The robot first attempts to go look at the 3D coordinates of the most relevant object identified in $O_T$ by the LLM Planner. We then pass the onboard camera image to LLaVA [55] and ask if the target object is in view. If not, we remove the target object from the scene graph and ask the LLM Planner to provide a new likely location for the object in the scene with the following GPT-4 system prompt:

The object described as 'description’ is not in the scene. Perhaps someone has moved it, or put it away. Let’s try to find the object by visiting the likely places, storage or containers that are appropriate for the missing object (eg: a a cabinet for a wineglass, or closet for a broom). The new query is: find a likely container or storage space where someone typically would have moved the object described as ‘description’?

Listing 5: GPT system prompt for object localization.
For traversability estimation, we task GPT to classify a given object as traversable or non-traversable based on its description and possible tags. The system prompt is:

You are a wheeled robot that can push a maximum of 5 pounds or 2.27 kg. Can you traverse through or push an object identified as ‘description’ with possible tags ‘possible_tags’? Specifically, is it possible for you to push the object out of its path without damaging yourself?

Listing 6: GPT-4 system prompt for traversability estimation.

We then take the pointclouds of each non-traversable objects and downproject them in the Jackal costmap before launching the navigation episode. The goal is provided in this case as a specific pose in the room.

For all experiments in this section, we run a local instance of LLaVA onboard on a desktop when needed and otherwise use the GPT-4 API for LLM queries.

A6. LIMITATIONS

As indicated in Sec. III-H, there are a few failure modes of ConceptGraphs that remain to be addressed in subsequent work. In particular, the LLaVA-7B [58] model used for node captioning misclassifies a non-negligible number of small objects as toothbrushes or pairs of scissors. We believe that using more performant vision-language models, including instruction-finetuned variants of LLaVA [70] can alleviate this issue to a large extent. This will, in turn, improve the node and edge precisions of 3D scene graphs beyond what we report in Table I.

In this work, we do not explicitly focus on improving LLM-based planning over 3D scene graphs. We refer the interested reader to concurrent work, SayPlan [52], for insights into how one might leverage the hierarchy inherent in 3D scene graphs, for efficient planning.

REFERENCES


