

Expanding Spatial and Temporal Context for Robotic Imitation Learning With Scene Graphs

Jianing Qian¹, Qinhe Peng^{1*}, Emmanuel Panov^{2*}, Leonor Fermoselle²,

Dinesh Jayaraman^{1**}, Bernadette Bucher^{3**}, Tarik Kelestemur^{2**}

¹University of Pennsylvania, ²RAI Institute, ³University of Michigan

{jianingq, pengqh20, dineshj}@seas.upenn.edu,
{epanov, lfermoselle, tkelestemur}@theaiinstitute.com,
bucherb@umich.edu

Abstract

Imitation learning enables robots to learn how to execute tasks via observation. However, real-world environments like homes and offices are often severely partially observed due to their large spatial scales. In addition, many tasks involve executing a series of subtasks requiring autonomous robots to reason over extended time horizons. To address these challenges, we propose using scene graphs as an explicit and structured memory mechanism in imitation learning. By maintaining a dynamic scene graph that captures object-centric relationships and their evolution over time, our method allows the agent to retain relevant historical context during task execution to efficiently reason over incrementally accrued scene information. Our experiments on simulated mobile manipulation and real-world tabletop manipulation demonstrate that our approach substantially improves policy performance, particularly in settings that demand long-term reasoning and robust generalization under partial observability. Code and videos: <https://sites.google.com/view/objgraph>.

1. Introduction

Imitation-learned policies have shown great promise for executing complex physical robotic skills, but largely over small spatial scales and short time horizons [12, 17, 63, 64]. At larger spatial and temporal contexts, their performance frequently deteriorates [24, 37]. A key source of difficulty in such tasks is partial observability. Consider, for example, a mobile manipulation task where it needs to move a chair to a different room. If a robot only has access to onboard sensor information during task execution, it is commonly not possible to fully observe a scene while executing actions.

Nevertheless, performing such tasks well often requires the robot to reason over information that is not currently in its field of view.

This motivates maintaining some form of memory for robotic agents operating in partially observed settings. What might such a memory look like? Modern computer vision approaches permit building and maintaining comprehensive metric maps aggregating information from all previous views of the scene, but such an overcomplete memory creates new issues for imitation policy learning: how should such large maps be processed in a neural policy, and how can we train such a policy sample-efficiently?

We propose to construct and maintain a novel scene memory representation for long spatial and temporal context imitation learning. When a policy is conditioned on this representation, it can make decisions based on all task-relevant information in the scene, even if it was observed in the distant past. We draw inspiration from the ample evidence to suggest that humans instead maintain abstract task-relevant *semantic* memories when accomplishing long-horizon tasks [7, 14, 21, 28, 30, 53]. Mirroring this, we use scene graphs to store scene information compactly for large environments in a manner that can be easily updated with dynamic environment changes. Scene graphs sparsely represent critical scene information over large spaces, even up to the scale of multi-floor buildings. Updating a scene graph from movement in the environment is straightforward since scene information is clearly discretized and organized in a graph structure. Furthermore, designing the scene graph to only track task-relevant information can enable efficient learning and execution of targeted actions even in large, cluttered scenes. However, no prior work directly leverages a scene graph as a model input in imitation learning. We fill this gap: we propose an approach to construct and dynamically update task-relevant scene graphs to provide the necessary spatial and temporal context for training and execut-

*Equal contribution.

**Equal advising.

ing imitation learning neural policies in partially observed settings.

2. Related Work

We begin by summarizing prior efforts on generating scene graphs using vision foundation models. We then review approaches that incorporate scene graphs into robot policy learning. Finally, we discuss imitation learning methods that introduce memory mechanisms to improve performance on long-horizon tasks.

Semantic Scene Graph with Vision Foundation Models.

Scene graphs [31, 32, 55, 58] have grown popular as a structured and interpretable image representation. In a scene graph, salient objects are represented as graph nodes, and edges represent their spatial relationships. The concept of the scene graph can naturally be extended from 2D to 3D, where input to the scene graph generation pipelines is no longer restricted to image observations, but also 3D maps of an environment. Previous works [1, 9, 18, 62] introduce methods to build 3D semantic scene graphs, either from videos, 3D mesh models, or 3D point clouds of the visual environment. Given the recent advances of vision-language foundation models [4, 29, 33, 40, 42, 65], many works propose to leverage their generalization ability in new environments and ground the output 2D representations from those models in 3D point clouds or maps. Chang et al. [5, 6], Gu et al. [20], Jatavallabhula et al. [25] propose frameworks to generate semantic 3D scene representations by prompting VLMs.

Most related to our approach is CLIO [36], which builds on a hierarchical 3D scene graph [22] to provide a framework to infer the appropriate granularity and sub-graph given user-input task instructions. Unlike CLIO, which only demonstrates simple object picking tasks, we focus on how to enable long-horizon task execution with a 3D semantic scene graph. Many other prior works also utilize scene graphs or object-centric representations to improve VLA models [3], guide long-horizon task planning [2, 39, 54, 57, 59], and actively explore the environment or navigate [13, 26, 43, 44, 60]. Unlike us, none of them show how modern neural imitation learning architectures could be modified to directly ingest dynamically maintained scene graphs as inputs. Similar to us, [49] also constructs scene graphs to aid visual imitation learning. However, they use scene graphs to map human demonstrations to robot ones, while we directly encode and input the scene graph to the downstream diffusion policies.

Imitation Learning with Memory. Imitation learning methods such as Diffusion Policy (DP) [10], Action Chunking Transformers (ACT) [66], and several variants of these methods [19, 61, 68] over the years have shown that we can learn dexterous and multi-modal tasks with a few hundred demonstrations. Later, by scaling the data and the net-

work capacity [23, 46, 56], we see that we can learn multi-task policies that are conditioned on the language or image goals. However, these models are still not able to handle long-horizon and large spatial information, as the majority of these works address tabletop manipulation, where partial observability is not a major issue. These challenges are particularly important for mobile manipulation, where the robot does not have an immediate view of its full environment. To this end, [52] introduced a variant of DP that targeted long-context visual imitation learning. They introduce an auxiliary loss to the network to avoid learning spurious correlations in the long-context data. On the other hand, [34] learns to incorporate state information by incorporating ControlNet. The efficient scene memory representation in our approach encodes both large-scale spatial and temporal context as input representations to DP. Critically, our memory representation is explicitly constructed, not learning over spatial information and time.

3. Method

We study the problem of imitation learning in partially observed environments. Our training dataset consists of M expert demonstrations, $D = \{\tau_i\}_{i=1}^M$, where each trajectory τ_i is a sequence of observation-action pairs $\tau_i = [(I_0, a_0), \dots, (I_T, a_T)]$ where I is an RGB-D image. A key challenge in our setting is that the observation at each timestep, I_t , does not fully capture the underlying state s_t of the environment.

This partial observability is a common and challenging scenario in robot learning. For example, on a mobile manipulation platform, visual observations are typically gathered from body-mounted cameras with a limited field-of-view that see only a very small portion of the relevant environment, such as a home or an office building. Even in table-top manipulation, wrist-mounted cameras are popular as they abstract away many details of the robot morphology, enable fast data collection, and induce robustness to extraneous information. However, in both cases, the robot must contend with incomplete and noisy sensory observations, making robust decision-making difficult.

Consider a mobile robot performing indoor household tasks, such as tidying a room or heating a cup of tea. With onboard sensors and limited perceptual coverage, the robot must act based on partial observations, without direct access to the full environment state. These settings are naturally modeled as Partially Observable Markov Decision Processes (POMDPs), where the agent must infer the hidden state of the world from a history of past observations and actions.

In principle, optimal policies π^* in these partially observed settings could require access to the entire observation-action history. However, learning such a full history-dependent policy is typically intractable, especially

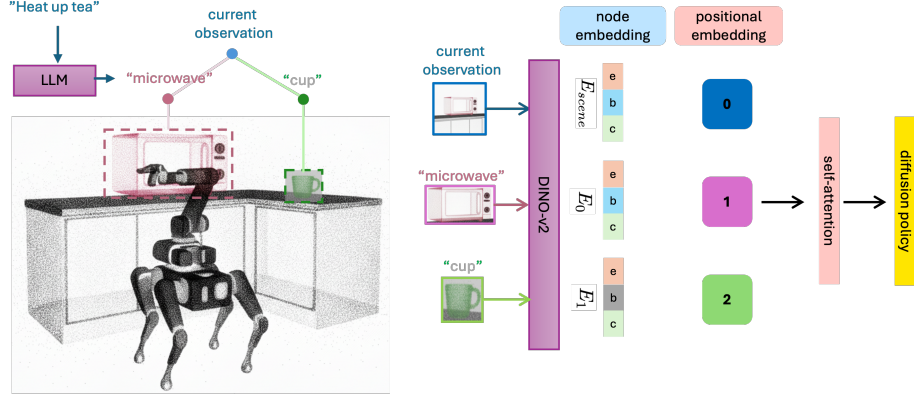


Figure 1. We maintain a scene graph where each task-relevant object is represented as a node $n_i = [e_i, b_i, c_i]$, consisting of a learned visual embedding e_i , its 2D bounding box b_i when visible, and its 3D centroid c_i in a world-aligned frame. Using a natural-language task description, large language models and vision foundation models identify and track relevant objects, allowing the graph to be incrementally updated as the robot interacts with the environment. This provides a compact, memory-efficient representation suitable for downstream imitation-learning policies. In the figure, the microwave is currently in view, while the cup is out of view (shown as a grayed-out area). We update b_i only when the object is visible.

in long-horizon tasks where the observation history grows rapidly and extracting relevant information becomes increasingly difficult [15, 27, 35, 50].

3.1. Task-Driven Scene Graph

To capture historical information compactly, we propose a structured memory representation in the form of a dynamically updated task-relevant scene graph, designed to support effective robotic decision-making under partial observability.

For our purposes, a scene graph refers to a graph-based abstraction in which task-relevant objects are represented as a set of nodes $N = \{n_i\} = \{[e_i, b_i, c_i]\}$, each consisting of an embedding e_i , its 2D bounding box coordinates b_i (if it is currently in view, empty otherwise), and its 3D centroid c_i in a world coordinate frame (anchored through odometry in mobile manipulation, or just attached to the robot base in tabletop manipulation). Importantly, while we do not explicitly parameterize edges as separate learnable entities, relational structure is implicitly encoded through shared coordinate frames and joint processing of node attributes. In particular, object centroids c_i define a common spatial reference frame, allowing pairwise relationships (e.g., relative position and distance) to be inferred directly from node features.

To identify and track potentially task-relevant objects, we leverage large language models and vision foundation models based on a concise natural language description of the task, as adopted from [41]. As the task unfolds, we incrementally update and maintain the scene graph to reflect the current state of the environment in a task-aware and memory-efficient way suitable as the state representation input to a neural imitation policy. We discuss these steps in detail in the following paragraphs.

Task-Relevant Object Enumeration. In realistic indoor

environments, exhaustively identifying and tracking all objects is computationally infeasible and unnecessary. To build a scene graph that is both compact and effective for policy learning, we focus on identifying only the task-relevant objects based on the provided task description.

Given a natural language task instruction, we use a large language model (e.g., GPT-5) to extract a list of noun entities (details in Sec. 6) that are likely to be important for the task. Once this list of task-relevant entity words is generated, we look to localize and track all objects matching any item on that list, as we describe below.

Scene Graph Construction and Maintenance. For the first image frame I_0 in an episode, we use Grounding DINO [33] to detect and localize objects matching the task-relevant entity list, producing a set of segmentation masks $M_{t=0} = \{m_0^j\}$. These detections initialize the nodes in the scene graph.

Node Representation. At any time step t , given segmentation masks $M_t = \{m_t^j\}$ for the current observation, we represent each object as a node by extracting object-centric features. For each mask m_t^j , we compute an appearance embedding e_j by average-pooling DINO-v2 [40] features within the masked region. We also extract a tight 2D bounding box b_j and estimate a 3D centroid c_j via depth backprojection using camera parameters and odometry. This yields a node representation $n_j = \{e_j, b_j, c_j\}$ capturing both semantic and spatial information.

Temporal Maintenance. To maintain the scene graph over time, we combine tracking of existing nodes with detection of newly observed objects. For each node already present in the graph, we initialize a visual tracker [8], enabling persistent identity across frames. When an object moves out of view, its 2D bounding box b_i is set to zero while its 3D centroid c_i is retained, assuming no movement unless re-observed. When the object reappears, both b_i and

c_i are updated accordingly.

Since the environment is only partially observed, new task-relevant objects may appear over time. To account for this, we run object detection in parallel at each time step. Each newly detected candidate n_j is compared against existing nodes n_i by computing mask overlap. If no existing node exceeds a threshold τ_m , the detection is initialized as a new node and added to the scene graph.

This design differs from a flat set of object tokens in two key ways. First, nodes maintain persistent identities across time and are incrementally updated, forming a temporally consistent structure. Second, spatial relationships between objects are preserved through their shared coordinate representation, enabling relational reasoning without requiring explicitly parameterized edges.

See Algorithm 1 for a detailed pseudocode of the scene graph update procedure.

3.2. Policy Learning

We now encode the scene graph as input to a neural diffusion policy.

Scene Graph-Conditioned Transformer Policy. Given the set of task-relevant object nodes $N = \{n_i\} = \{[e_i, b_i, c_i]\}$, we construct a structured representation of the current scene and use it to condition a transformer-based policy.

Scene and Object Encoding. For each observation I , we first extract a global scene representation e_{scene} using the CLS token of a DINO-v2 ViT encoder. We associate this with a scene-level node defined by a bounding box $b_{\text{scene}} = [0., 1., 0., 1.]$ in normalized image coordinates and a 3D coordinate $c_{\text{scene}} = [0., 0., 0.]$ representing the origin of the global frame.

For each object node $n_i \in N$, we compute its visual embedding e_i as described earlier. The geometric attributes, including the 2D bounding box b_i and 3D centroid c_i , are encoded via small MLPs. The final node embedding is given by

$$E_i = [e_i, \text{MLP}(b_i), \text{MLP}(c_i)].$$

Similarly, the scene-level embedding is

$$E_{\text{scene}} = [e_{\text{scene}}, \text{MLP}(b_{\text{scene}}), \text{MLP}(c_{\text{scene}})].$$

Scene Graph Representation. The full scene graph representation is given by the set of node embeddings

$$R_{sg} = [E_{\text{scene}}, E_1, \dots, E_i, \dots],$$

where E_{scene} serves as a root node and E_i correspond to task-relevant objects. This representation preserves object identity and spatial structure, while enabling relational reasoning through joint processing of all nodes.

Transformer Encoding. We input R_{sg} as a sequence of tokens to a transformer encoder. Positional embeddings are

Algorithm 1 Scene Graph Maintenance Pseudocode

Require: RGB image I_{rgb} , depth image I_d , camera intrinsics E_i , camera extrinsics E_e , task entities $\mathcal{N}_{\text{task}}$, overlap threshold τ_m

Ensure: Scene graph $\mathcal{V}$

```

1: Function GenerateSceneGraph( $I_{\text{rgb}}, I_d, E_i, E_e, \mathcal{N}_{\text{task}}, \tau_m$ )
   // update scene graph at every timestep t
2: for  $v_i \in \mathcal{V}$  do
3:   if ISVISIBLE( $v_i, I_{\text{rgb}}$ ) then
4:      $(e_i, b_i, c_i) \leftarrow \text{UPDATEFROMTRACKER}(v_i)$ 
5:   else
6:      $b_i = (0, 0, 0, 0)$ , keep  $c_i$  unchanged
7:   end if
8: end for
9:  $M \leftarrow \text{DETECTOBJECTS}(I_{\text{rgb}}, \mathcal{N}_{\text{task}})$ 
10: for  $m^j \in M$  do
11:    $v_i \leftarrow \text{OVERLAPWITHEXISTING}(m^j, \mathcal{V}, \tau_m)$ 
12:   if not  $v_i$  then
13:      $b_j \leftarrow \text{GETBBBOX}(m^j)$ 
14:      $e_j \leftarrow \text{EXTRACTDINOFEATURES}(I_{\text{rgb}}, b_j)$ 
15:      $c_j \leftarrow \text{COMPUTE3DCENTROID}(I_d, b_j, E_i, E_e)$ 
16:      $\mathcal{V} \leftarrow \mathcal{V} \cup \{(e_j, b_j, c_j)\}$ 
17:   end if
18: end for
19: return  $\mathcal{V}$ 

```

used to encode the structural roles of nodes (e.g., a distinct embedding for the scene-level root). The transformer processes all tokens jointly via self-attention, allowing interactions between objects and the global scene context. The resulting features are aggregated through an MLP head to produce a compact scene representation, which conditions the downstream policy.

In our current tasks, the scene graph typically forms a shallow hierarchical structure (i.e., a scene-level root and object nodes). Despite this simplicity, it captures the object-centric state and spatial relationships necessary for decision-making.

Imitation Learning with Diffusion. We use the scene graph representation R_{sg} to condition a diffusion-based policy that generates short-horizon action sequences.

During training, expert actions are corrupted using a forward diffusion process, and a denoising model $\epsilon_\theta(a_k, R_{sg}, k)$ is trained to predict the injected noise under the DDPM objective:

$$\mathcal{L}(\theta) = \mathbb{E}_{\mathbf{a}_0, k \sim \mathcal{U}(1, K), \epsilon \sim \mathcal{N}(0, \mathbf{I})} [\|\epsilon - \epsilon_\theta(a_k, R_{sg}, k)\|^2].$$

At test time, the policy initializes from Gaussian noise and iteratively applies reverse diffusion (using a DDIM sampler) to produce a coherent action sequence conditioned on the current scene graph.

4. Experiments

In this section, we evaluate our method for learning long-horizon imitation policies under partial observability. We address three key questions: (1) Does our method outperform prior imitation learning baselines for long-horizon tasks? (2) How well does it generalize across diverse table-top and mobile manipulation tasks? (3) How important are the individual components of our approach?

To answer these, we conduct experiments in both simulation and real-world settings and comprehensively evaluate performance.

4.1. Simulated Mobile Manipulation

We first evaluate our approach on three simulated mobile manipulation tasks (Fig. 3) with the Boston Dynamics Spot robot in MuJoCo [51]:

1. **Throw Trash:** The robot must walk to a box, pick it up, and throw it into a trash can. The locations of both the box and the trash can are randomized in each episode. Each episode contains around 100 timesteps.
2. **Heat-up Tea:** The robot walks to a microwave, opens the door, walks to a cup, picks it up, and uses its body to nudge the microwave door open before placing the cup inside. Each episode contains around 300 timesteps.
3. **Heat-up Tea Long:** A more challenging variant of Heat-up Tea where the microwave and the cup are located further apart, increasing the navigation complexity. Each episode contains around 1000 timesteps.

These tasks are designed to be long-horizon and composed of multiple subtasks involving both navigation and manipulation.

The Spot robot is equipped with two cameras: one mounted on the front of the body and another on the end effector. For each task, we collect 400 scripted demonstrations. To enable mobile manipulation behaviors in simulation, we train a Deep Reinforcement Learning (DRL) agent for the low-level locomotion controller. Please see Supplementary Material for more details.

The low-level policy, denoted as $\pi_{ll}(a_t|s_t, g_t)$, receives proprioceptive information s_t and base velocity commands g_t , and outputs joint angle targets for the legs. Arm joint angles are sent directly to the arm’s PD controller. To ensure robust whole-body execution, we randomly sample arm configurations during training so that the robot learns to walk while the arm is in various poses. More details on the low-level controller and the data collection process are provided in the Supplementary Material.

We collect demonstrations such that, at any given time, the robot operates in either locomotion mode or manipulation mode. In locomotion mode, the arm joints are frozen, and only the velocities of the legs are controlled; in manipulation mode, the legs are fixed while the arm is active. In addition to joint and leg velocities, we also record a mode

indicator.

During policy training, our network is designed to predict both the joint angle targets for the legs and arm, as well as the mode indicator. During data collection, we apply a sigmoid function to the raw mode signal to generate a smooth, continuous target. This soft target is then used to train the policy, enabling smoother transitions between locomotion and manipulation during execution. During execution, we apply a masking mechanism to activate the leg or arm control outputs based on the predicted mode.

Baselines. We compare our method with three baselines:

1. **RGB:** Vanilla diffusion policy that takes in RGB images from two cameras mounted on the Spot and proprioceptive state as input;
2. **Visible-Objects:** Diffusion policy that takes instantaneous object-centric representations as input. Compared to our method, these object-centric representations are constructed using only the current camera views.
3. **PTP:** To overcome the difficulties of learning long-horizon imitation learning tasks, [52] proposes a method to input more history frames as well as predict both past and future actions. We implement a variant of diffusion policy where we increase the history frames from 1 to 10 and predict both the past 10 and future 32 actions.

Ablations. In our scene graph design, each object node n_i encodes not only the visual features of a task-relevant object but also its 2D bounding box coordinates and 3D centroid. To assess the contribution of these components to the overall performance, we conduct a series of ablation studies in addition to the main baselines. Specifically, we evaluate two variants:

1. **Ours w/o 3D**, which removes the 3D centroid c_i from the node representation. Objects that are currently visible are represented by their object embedding e_i and bounding box b_i , and those that are not are represented solely as e_i . In other words, there is no memory of *where* in the scene they were observed, just a memory of the fact that they exist in the scene and of their appearance.
2. **Ours w/o object locations**, which removes all location information associated with each object node i.e., each object is represented only by its appearance embedding e_i .

Performance Metrics. Each simulated task consists of two or three subtasks. To clearly show task progress, we report results using stacked bar plots (Fig. 2). The bottom segment of each stack indicates the success rate of the full task, while higher segments show success rates for completing earlier subtasks. A gray background bar denotes the fraction of evaluations that fail to complete any subtasks.

Results. The results for the simulated tasks are shown in Fig. 2. We observe that errors in early subtasks often propagate and compound over time, as later stages depend on the resulting scene configuration. This effect is particu-

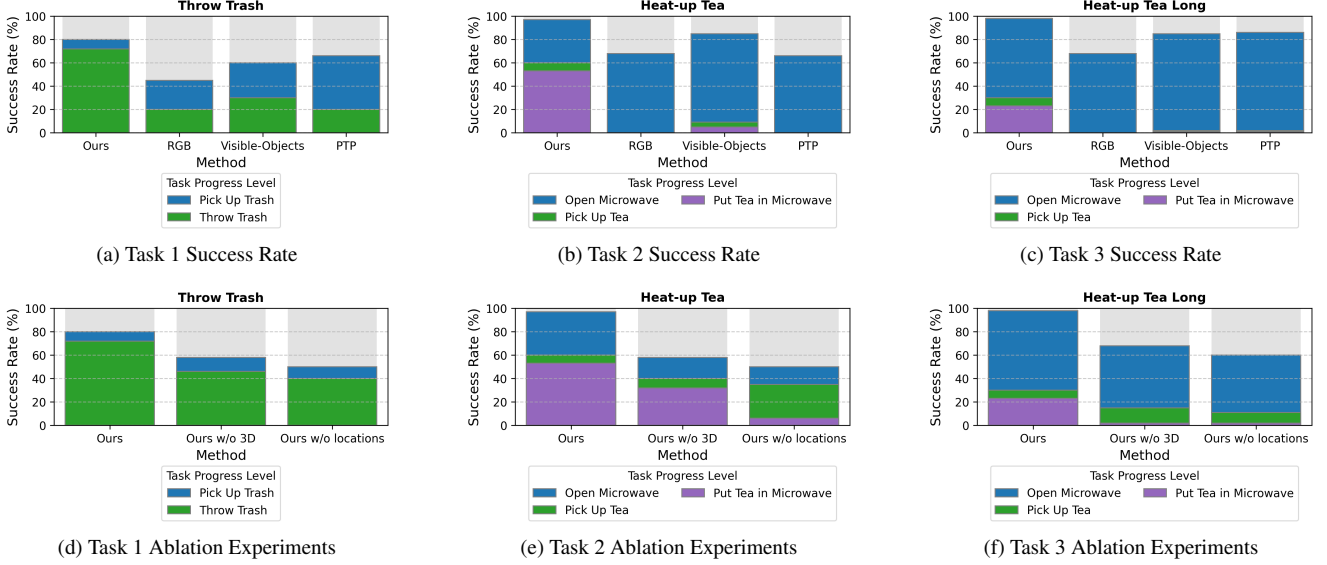


Figure 2. Success rates of different methods and their ablations across three simulated tasks are represented by stacked bar plots. The lowest bar in each stacked column represents the success rate of that task, while the other bars show success rates for completing earlier subtasks that lead up to the final tasks. The gray bar in the back represents the number of evaluation runs that complete zero subtasks.



Figure 3. Illustrations of the simulated mobile manipulation tasks. Early pronounced under partial observability, where recovering from earlier mistakes requires remembering previously observed states. Our method mitigates this issue by maintaining object-centric memory, leading to more consistent performance across long-horizon tasks.

For Heat-up Tea and Heat-up Tea Long in particular, the third step—nudging open the microwave door and placing the cup inside—depends heavily on how the microwave was opened during the very first subtask. Without an explicit memory mechanism, baseline methods struggle to relocate

the microwave door hundreds of steps later. The PTP baseline further highlights this limitation: simply increasing the temporal context improves short-horizon reasoning (e.g., on the first subtask) but does not address the fundamental challenges of long-horizon manipulation.

To understand which design choices most contribute to performance, we evaluate two ablations: **Ours w/o 3D** and **Ours w/o object locations**. The results in Fig. 2d, Fig. 2e, and Fig. 2f show that removing the 3D centroid or both 3D and 2D location information does not significantly affect performance on the first subtask. However, the success rates drop sharply for the later subtasks and the final task. This suggests that in long-horizon settings, providing explicit spatial information (3D centroids and 2D bounding boxes) is crucial—it helps the policy more reliably localize and track task-relevant objects over time.

4.2. Real World Tabletop Manipulation

Next, we evaluate our method in a partially observed tabletop setup. In these experiments, we use the 7-DoF Franka Emika Panda robotic arm. Unlike the original setup, we remove the two side-mounted cameras, retaining only the wrist-mounted ZED Mini RGB-D camera. As we have noted above, this is a commonly used setup in robotic manipulation: several prior works [11, 16] rely solely on wrist-mounted cameras because they permit simpler, morphology-agnostic data collection, and the resulting policies are naturally invariant to many task-irrelevant aspects of the scene because they mostly do not even appear within the image observations. However, it also introduces new challenges (see Fig. 5): as the wrist-mounted camera moves

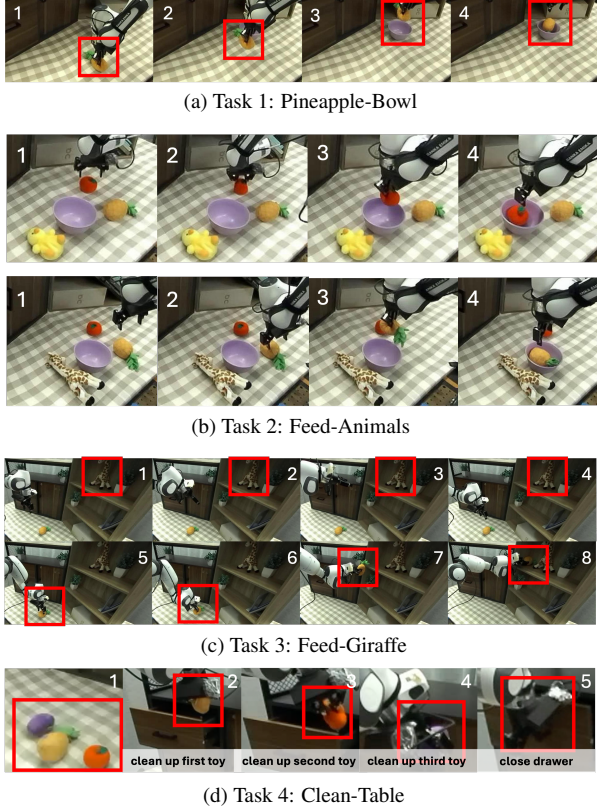


Figure 4. Illustration of the real-world tabletop manipulation tasks.

during task execution, it often provides limited and changing viewpoints. The policies must therefore rely on observations accumulated throughout the trajectory to reason effectively about the scene.

Data Collection and Setup. We set up a tabletop environment that contains a variety of everyday objects and several cabinets mounted on the countertop. A ZED Mini RGB-D camera is mounted on the robot’s wrist and serves as the sole visual sensor, providing RGB-D image streams for both data collection and policy learning. For each task, as illustrated in Fig. 4, we collect 300 teleoperated demonstrations. The task suite is designed to increase in difficulty progressively, allowing us to evaluate the method’s ability to handle partial observability and long-horizon challenges. Additional implementation and environment details are provided in the Supplementary Material. Our tasks are:

1. Put Pineapple in Bowl (**Pineapple-Bowl**): The robot must pick up a plush pineapple and place it into a bowl. The main challenge is partial observability: with a wrist-mounted camera, the bowl leaves the field of view once the pineapple is lifted, requiring the policy to maintain a memory of its location.
2. Feed Two Animals (**Feed-Animals**): A pineapple and an apple are placed on the table, and either a giraffe or a chicken appears at the start of the episode. The robot must choose the correct fruit based on the animal

(pineapple for giraffe, apple for chicken), testing its ability to retain and act on initial visual context.

3. Feed Giraffe (**Feed-Giraffe**): A giraffe is positioned on one of three bookshelf levels. The robot must first scan the shelf to locate it, then pick up a pineapple and place it at the corresponding level. This requires long-horizon reasoning and persistent memory of the target location.
4. Clean Table (**Clean-Table**): One to three fruit toys are placed on the table. The robot must collect all objects and place them into a drawer, and only close the drawer once the table has been fully cleared. This requires tracking which objects have already been moved and ensuring the policy does not prematurely terminate the task.

Results. See Fig. 6 for quantitative results. We compare our method against all baselines and ablations, excluding PTP because it rarely progresses beyond the first task stage in simulated tasks.

For the **Pineapple-Bowl** task, our method achieves the highest success rate in picking up the pineapple and performs on par with **Ours w/o 3D** and **Ours w/o object locations** for placing it in the bowl, as it maintains a consistent memory of the bowl’s location even when being out of view. **Visible-Objects** outperforms **RGB**, indicating that object-centric representations improve grasping and placement accuracy, consistent with prior OCR-based work [48, 67]. However, both **RGB** and **Visible-Objects** frequently misplace the pineapple when the bowl is far from its initial position.

For the **Feed-Animals** task, we observe a large performance gap between our method and the baselines. Both **RGB** and **Visible-Objects** tend to pick up a random fruit regardless of the target animal, reflecting an inability to retain and act on initial visual context under partial observability.

Feed-Giraffe is a challenging task, requiring the robot to scan the bookshelf, locate the giraffe, and place the pineapple at the correct level. This long-horizon dependency makes it difficult for memory-less policies: while both our method and **Visible-Objects** can pick up the pineapple, baselines often either fail to retrieve it or lose track of the giraffe’s location during placement.

For the **Clean-Table** task, the robot must remove all objects before closing the drawer, requiring it to keep track of which items have already been cleared. Policies without explicit memory often prematurely close the drawer or leave objects behind, indicating difficulty in maintaining task progress over time. In contrast, our method achieves more reliable completion by maintaining a persistent representation of object states throughout the episode.

Error Propagation and Runtime Analysis. The above results suggest that performance is primarily limited by long-horizon reasoning rather than perception. To better understand system behavior, we analyze error sources and runtime.



Figure 5. **Partial-Observation Tabletop Setup.** We evaluate our method using a 7-DoF Franka Emika Panda arm with only a wrist-mounted ZED Mini RGB-D camera, removing the side cameras. While this setup simplifies data collection and makes policies invariant to many task-irrelevant scene features, it introduces limited and changing viewpoints. Policies must therefore leverage observations accumulated over the trajectory to reason effectively about the scene.

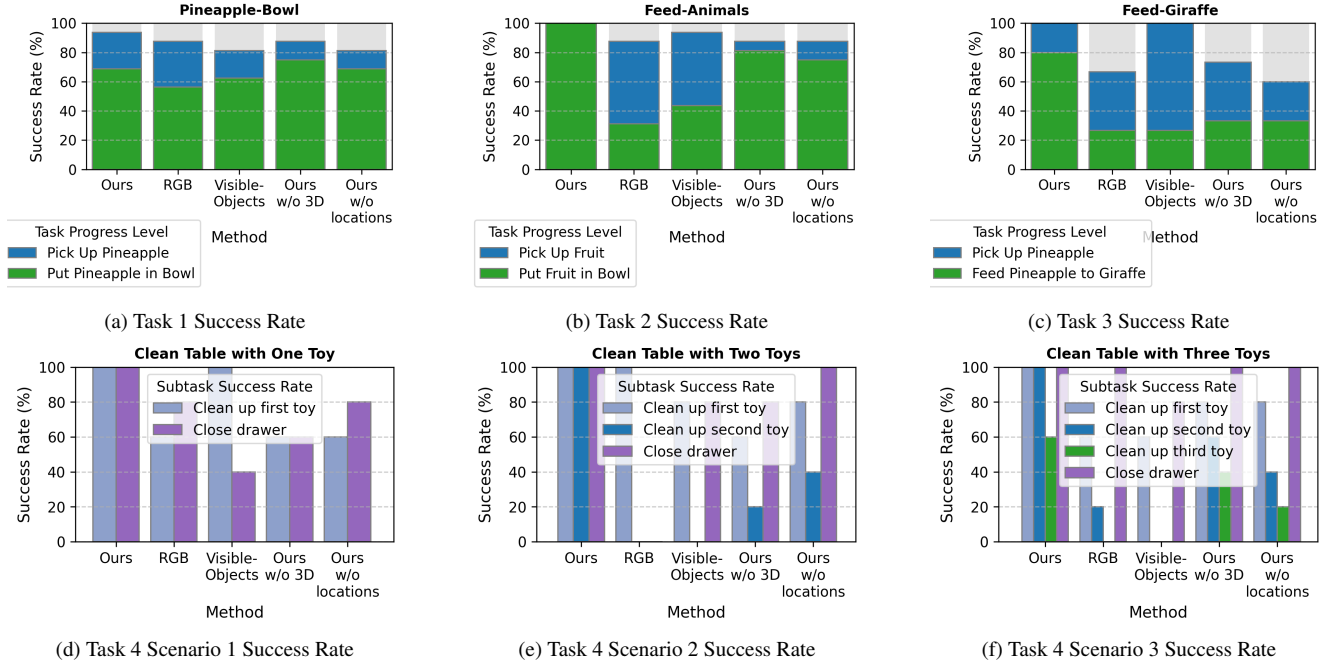


Figure 6. Success rates of different methods and their ablations on the first three real-world tabletop manipulation tasks are shown as stacked bar plots. Each column corresponds to a method, and each segment represents the success rate of a subtask. The bottom segment denotes the success rate of completing the full task, while higher segments indicate success on intermediate subtasks. The gray segment represents runs that fail to complete any subtask. For the fourth task, we report results separately for three scenarios: cleaning up one, two, or three toys.

Failure Analysis. Our system combines LLM-based task parsing, object segmentation, tracking, and policy inference. The LLM is queried once per task to identify relevant objects, which are segmented in the first frame using Grounded-SAM and tracked with XMem. In simulation, we observe no failures from perception or tracking. In real-world experiments (64 episodes), segmentation fails in 4 cases due to occlusion or ambiguous boundaries. We observe no failures from LLM-based object enumeration or from tracking after initialization. Instead, most failures arise from the imitation policy in long-horizon settings, consistent with the challenges observed in tasks such as **Feed-Giraffe** and **Clean-Table**.

Runtime. Expensive components are used sparingly: the LLM is queried once per task, and GroundingDINO runs once per episode (0.53 s). During execution, mask propagation takes 0.19 s per frame. The diffusion policy runs at 3.5-3.8 Hz on a real robot using a single NVIDIA 3090 GPU (around 260-285 ms per step), enabling real-

time closed-loop control. The primary runtime cost during execution is mask propagation, while other components are amortized over the episode.

5. Limitations and Conclusions

We introduce task-relevant scene graphs as a compact, dynamically updated memory for imitation learning under partial observability. By encoding only relevant objects, the representation enables reasoning over past observations beyond the robot’s field of view. While effective for long-horizon tasks, it depends on accurate object identification and may struggle in highly cluttered or dynamic environments. This suggests scene graphs as a promising interpretable memory for future vision-language-action models.

Acknowledgments

This work was supported in part by the National Science Foundation (NSF) under CAREER Award #2239301 and SLES Award #2331783, the Office of Naval Research (ONR) under Grant N00014-22-1-2677, and DARPA under Grant HR00112490421 (TIAMAT).

References

- [1] Iro Armeni, Zhi-Yang He, JunYoung Gwak, Amir Zamir, Martin Fischer, Jitendra Malik, and Silvio Savarese. 3d scene graph: A structure for unified semantics, 3d space, and camera. *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 5663–5672, 2019.
- [2] Katrina Ashton, Chahyon Ku, Shrey Shah, Wen Jiang, Kostas Daniilidis, and Bernadette Bucher. HELIOS: Hierarchical Exploration for Language-grounded Interaction in Open Scenes. *ArXiv*, abs/2509.22498, 2025.
- [3] Rokas Bendikis, Daniel Dijkman, Markus Peschl, Sanjay Haresh, and Pietro Mazzaglia. Focusing on what matters: Object-agent-centric tokenization for vision language action models. *ArXiv*, abs/2509.23655, 2025.
- [4] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2021.
- [5] Haonan Chang, Kowndinya Boyalakuntla, Shiyang Lu, Siwei Cai, Eric Pu Jing, Shreesh Keskar, Shijie Geng, Adeeb Abbas, Lifeng Zhou, Kostas E. Bekris, and Abdeslam Boularias. Context-aware entity grounding with open-vocabulary 3d scene graphs. *ArXiv*, abs/2309.15940, 2023.
- [6] Yun Chang, Leonor Fermoselle, Duy Ta, Bernadette Bucher, Luca Carlone, and Jiuguang Wang. ASHiTA: Automatic Scene-grounded Hierarchical Task Analysis. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025.
- [7] Ye Chen, Peter Beech, Ziwei Yin, Shanshan Jia, Jiayi Zhang, Zhaofei Yu, and Jian K Liu. Decoding dynamic visual scenes across the brain hierarchy. *PLOS Computational Biology*, 20(8):e1012297, 2024.
- [8] Ho Kei Cheng and Alexander G. Schwing. Xmem: Long-term video object segmentation with an atkinson-shiffrin memory model. In *European Conference on Computer Vision*, 2022.
- [9] Shun cheng Wu, Johanna Wald, Keisuke Tateno, Nassir Navab, and Federico Tombari. Scenegraphfusion: Incremental 3d scene graph prediction from rgb-d sequences. *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7511–7521, 2021.
- [10] Cheng Chi, Zhenjia Xu, Siyuan Feng, Eric Cousineau, Yilun Du, Benjamin Burchfiel, Russ Tedrake, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. *The International Journal of Robotics Research*, page 02783649241273668, 2023.
- [11] Cheng Chi, Zhenjia Xu, Chuer Pan, Eric Cousineau, Benjamin Burchfiel, Siyuan Feng, Russ Tedrake, and Shuran Song. Universal manipulation interface: In-the-wild robot teaching without in-the-wild robots. *ArXiv*, abs/2402.10329, 2024.
- [12] Sudeep Dasari and et al. Transformers for one-shot imitation learning. In *CoRL*, 2020.
- [13] Venkata Naren Devarakonda, Raktim Gautam Goswami, Ali Umut Kaypak, Naman Patel, Rooholla Khorrambakht, Prashanth Krishnamurthy, and Farshad Khorrami. Orionnav: Online planning for robot autonomy with context-aware llm and open-vocabulary semantic scene graphs. *ArXiv*, abs/2410.06239, 2024.
- [14] Adrien Doerig, Tim C Kietzmann, Emily Allen, Yihan Wu, Thomas Naselaris, Kendrick Kay, and Ian Charest. Visual representations in the human brain are aligned with large language models, 2024.
- [15] Shi Dong, Benjamin Van Roy, and Zhengyuan Zhou. Simple agent, complex environment: Efficient reinforcement learning with agent states. *Journal of Machine Learning Research*, 23(255):1–54, 2022.
- [16] Haritheja Etukuru, Norihito Naka, Zijin Hu, Seungjae Lee, Julian Mehu, Aaron Edsinger, Chris Paxton, Soumith Chintala, Lerrel Pinto, and Nur Muhammad Mahi Shafiullah. Robot utility models: General policies for zero-shot deployment in new environments. *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pages 8275–8283, 2024.
- [17] Pete Florence, Lucas Manuelli, and Russ Tedrake. Implicit behavioral cloning. *RSS*, 2022.
- [18] Paul Gay, Stuart James, and Alessio Del Bue. Visual graphs from motion (vgfm): Scene understanding with object geometry reasoning. In *Asian Conference on Computer Vision*, 2018.
- [19] Ankit Goyal, Valts Blukis, Jie Xu, Yijie Guo, Yu-Wei Chao, and Dieter Fox. Rvt-2: Learning precise manipulation from few demonstrations. *arXiv preprint arXiv:2406.08545*, 2024.
- [20] Qiao Gu, Ali Kuwajerwala, Sacha Morin, Krishna Murthy Jatavallabhula, Bipasha Sen, Aditya Agarwal, Corban Rivera, William Paul, Kirsty Ellis, Ramalingam Chellappa, Chuang Gan, Celso Miguel de Melo, Joshua B Tenenbaum, Antonio Torralba, Florian Shkurti, and Liam Paull. Conceptgraphs: Open-vocabulary 3d scene graphs for perception and planning. *ArXiv*, abs/2309.16650, 2023.
- [21] Zihong He, Weizhe Lin, Hao Zheng, Fan Zhang, Matt W Jones, Laurence Aitchison, Xuhai Xu, Miao Liu, Per Ola Kristensson, and Junxiao Shen. Human-inspired perspectives: A survey on ai long-term memory. *arXiv preprint arXiv:2411.00489*, 2024.
- [22] Nathan Hughes, Yun Chang, Siyi Hu, Rajat Talak, Rumaisa Abdulhai, Jared Strader, and Luca Carlone. Foundations of spatial perception for robotics: Hierarchical representations and real-time systems. *ArXiv*, abs/2305.07154, 2023.
- [23] Physical Intelligence, Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, et al. $\pi_{0.5}$: a vision-language-action model with open-world generalization. *arXiv preprint arXiv:2504.16054*, 2025.
- [24] Eric Jang and et al. Bc-z: Zero-shot task generalization with robotic imitation learning. *CoRL*, 2022.

- [25] Krishna Murthy Jatavallabhula, Ali Kuwajerwala, Qiao Gu, Mohd. Omama, Tao Chen, Shuang Li, Ganesh Iyer, Soroush Saryazdi, Nikhil Varma Keetha, Ayush Kumar Tewari, Joshua B. Tenenbaum, Celso M. de Melo, M. Krishna, Liam Paull, Florian Shkurti, and Antonio Torralba. Conceptfusion: Open-set multimodal 3d mapping. *ArXiv*, abs/2302.07241, 2023.
- [26] Hanxiao Jiang, Binghao Huang, Ruihai Wu, Zhuoran Li, Shubham Garg, Hooshang Nayyeri, Shenlong Wang, and Yunzhu Li. Roboexp: Action-conditioned scene graph via interactive exploration for robotic manipulation. *ArXiv*, abs/2402.15487, 2024.
- [27] Nan Jiang. On value functions and the agent–environment boundary. *arXiv preprint arXiv:1905.13341*, 2019.
- [28] M Jaleed Khan, Filip Ilievski, John G Breslin, and Edward Curry. A survey of neurosymbolic visual reasoning with scene graphs and common sense knowledge. *Neurosymbolic Artificial Intelligence*, 1:NAI–240719, 2025.
- [29] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C. Berg, Wan-Yen Lo, Piotr Dollár, and Ross B. Girshick. Segment anything. *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 3992–4003, 2023.
- [30] Abhilasha A Kumar. Semantic memory: A review of methods, models, and current challenges. *Psychonomic bulletin & review*, 28(1):40–80, 2021.
- [31] Yikang Li, Wanli Ouyang, Bolei Zhou, Kun Wang, and Xiaogang Wang. Scene graph generation from objects, phrases and region captions. *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 1270–1279, 2017.
- [32] Xin Lin, Changxing Ding, Jinquan Zeng, and Dacheng Tao. Gps-net: Graph property sensing network for scene graph generation. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3743–3752, 2020.
- [33] Shilong Liu, Zhaoyang Zeng, Tianhe Ren, Feng Li, Hao Zhang, Jie Yang, Chun yue Li, Jianwei Yang, Hang Su, Jun-Juan Zhu, and Lei Zhang. Grounding dino: Marrying dino with grounded pre-training for open-set object detection. *ArXiv*, abs/2303.05499, 2023.
- [34] Xiao Liu, Fabian Weigend, Yifan Zhou, and Heni Ben Amor. Enabling stateful behaviors for diffusion-based policy learning. *arXiv preprint arXiv:2404.12539*, 2024.
- [35] Xiuyuan Lu, Benjamin Van Roy, Vikranth Dwaracherla, Morteza Ibrahimi, Ian Osband, Zheng Wen, et al. Reinforcement learning, bit by bit. *Foundations and Trends in Machine Learning*, 16(6):733–865, 2023.
- [36] Dominic Maggio, Yun Chang, Nathan Hughes, Matthew Trang, Dan Griffith, Carlyn Dougherty, Eric Cristofalo, Lukas Schmid, and Luca Carlone. Clío: Real-time task-driven open-set 3d scene graphs. 2024.
- [37] Ajay Mandlekar and et al. What matters in learning from offline human demonstrations for robot manipulation. In *CoRL*, 2021.
- [38] Mayank Mittal, Pascal Roth, James Tigue, Antoine Richard, Octi Zhang, Peter Du, Antonio Serrano-Muñoz, Xinjie Yao, René Zurbügg, Nikita Rudin, et al. Isaac lab: A gpu-accelerated simulation framework for multi-modal robot learning. *arXiv preprint arXiv:2511.04831*, 2025.
- [39] Zhe Ni, Xiao-Xin Deng, Cong Tai, Xin-Yue Zhu, Xiang Wu, Y. Liu, and Long Zeng. Grid: Scene-graph-based instruction-driven robotic task planning. *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 13765–13772, 2023.
- [40] Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Q. Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, Mahmoud Assran, Nicolas Ballas, Wojciech Galuba, Russ Howes, Po-Yao (Bernie) Huang, Shang-Wen Li, Ishan Misra, Michael G. Rabbat, Vasu Sharma, Gabriel Synnaeve, Huijiao Xu, Hervé Jégou, Julien Mairal, Patrick Labatut, Armand Joulin, and Piotr Bojanowski. Dinov2: Learning robust visual features without supervision. *ArXiv*, abs/2304.07193, 2023.
- [41] Jianing Qian, Yunshuang Li, Bernadette Bucher, and Dinesh Jayaraman. Task-oriented hierarchical object decomposition for visuomotor control. In *Conference on Robot Learning*, 2024.
- [42] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning*, 2021.
- [43] Zachary Ravichandran, Lisa Peng, Nathan Hughes, J. Daniel Griffith, and Luca Carlone. Hierarchical representations and explicit memory: Learning effective navigation policies on 3d scene graphs using graph neural networks. *2022 International Conference on Robotics and Automation (ICRA)*, pages 9272–9279, 2021.
- [44] Sonia Raychaudhuri, Duy Ta, Katrina Ashton, Angel X Chang, Jiuguang Wang, and Bernadette Bucher. Zero-shot object-centric instruction following: Integrating foundation models with traditional navigation. *ArXiv*, abs/2411.07848, 2024.
- [45] Nikita Rudin, David Hoeller, Philipp Reist, and Marco Hutter. Learning to walk in minutes using massively parallel deep reinforcement learning. In *Conference on robot learning*, pages 91–100. PMLR, 2022.
- [46] Vaibhav Saxena, Matthew Bronars, Nadun Ranawaka Arachchige, Kuancheng Wang, Woo Chul Shin, Soroush Nasiriany, Ajay Mandlekar, and Danfei Xu. What matters in learning from large-scale datasets for robot manipulation. *arXiv preprint arXiv:2506.13536*, 2025.
- [47] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.
- [48] Junyao Shi, Jianing Qian, Yecheng Jason Ma, and Dinesh Jayaraman. Composing pre-trained object-centric representations for robotics from “what” and “where” foundation models. *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 15424–15432, 2024.
- [49] Maximilian Sieb, Xian Zhou, Audrey Huang, Oliver Kroemer, and Katerina Fragkiadaki. Graph-structured visual imitation. In *Conference on Robot Learning*, 2019.

- [50] Amit Sinha and Aditya Mahajan. Agent-state-based policies in pomdps: Beyond belief-state mdps. *arXiv preprint arXiv:??*, 2023.
- [51] Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ international conference on intelligent robots and systems*, pages 5026–5033. IEEE, 2012.
- [52] Marcel Torne, Andy Tang, Yuejiang Liu, and Chelsea Finn. Learning long-context diffusion policies via past-token prediction. *arXiv preprint arXiv:2505.09561*, 2025.
- [53] Endel Tulving. Episodic and semantic memory. In *Organization of Memory*, pages 381–403. Academic Press, 1972.
- [54] Kaijun Wang, Liqin Lu, Mingyu Liu, Jianuo Jiang, Zeju Li, Bolin Zhang, Wancai Zheng, Xinyi Yu, Hao Chen, and Chunhua Shen. Odyssey: Open-world quadrupeds exploration and manipulation for long-horizon tasks. *ArXiv*, abs/2508.08240, 2025.
- [55] Danfei Xu, Yuke Zhu, Christopher Bongsoo Choy, and Li Fei-Fei. Scene graph generation by iterative message passing. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3097–3106, 2017.
- [56] Ge Yan, Jiyue Zhu, Yuquan Deng, Shiqi Yang, Ri-Zhao Qiu, Xuxin Cheng, Marius Memmel, Ranjay Krishna, Ankit Goyal, Xiaolong Wang, et al. Manifold: A general robot manipulation policy via consistency flow training. *arXiv preprint arXiv:2509.01819*, 2025.
- [57] Zhijie Yan, Shufei Li, Zuoxu Wang, Lixiu Wu, Han Wang, Jun-Yan Zhu, Lijiang Chen, and Jihong Liu. Dynamic open-vocabulary 3d scene graphs for long-term language-guided mobile manipulation. *IEEE Robotics and Automation Letters*, 10:4252–4259, 2024.
- [58] Jianwei Yang, Jiasen Lu, Stefan Lee, Dhruv Batra, and Devi Parikh. Graph r-cnn for scene graph generation. In *European Conference on Computer Vision*, 2018.
- [59] Ruochu Yang, Yu Zhou, Fumin Zhang, and Mengxue Hou. Interleaved llm and motion planning for generalized multi-object collection in large scene graphs. *ArXiv*, abs/2507.15782, 2025.
- [60] Naoki Yokoyama, Sehoon Ha, Dhruv Batra, Jiuguang Wang, and Bernadette Bucher. Vlfm: Vision-language frontier maps for zero-shot semantic navigation. *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 42–48, 2023.
- [61] Yanjie Ze, Gu Zhang, Kangning Zhang, Chenyuan Hu, Muhan Wang, and Huazhe Xu. 3d diffusion policy: Generalizable visuomotor policy learning via simple 3d representations. *arXiv preprint arXiv:2403.03954*, 2024.
- [62] Shoulong Zhang, Shuai Li, Aimin Hao, and Hong Qin. Knowledge-inspired 3d scene graph prediction in point cloud. In *Neural Information Processing Systems*, 2021.
- [63] Tingfan Zhang, Zoe McCarthy, Eric Jang, and Sergey Levine. Deep imitation learning for complex manipulation tasks from virtual reality teleoperation. In *IROS*, 2018.
- [64] Yunzhu Zhang and et al. Perceiver-actor: A multi-task transformer for robotic manipulation. In *CoRL*, 2021.
- [65] Youcai Zhang, Xinyu Huang, Jinyu Ma, Zhaoyang Li, Zhaochuan Luo, Yanchun Xie, Yuzhuo Qin, Tong Luo, Yaqian Li, Siyi Liu, Yandong Guo, and Lei Zhang. Recognize anything: A strong image tagging model. *ArXiv*, abs/2306.03514, 2023.
- [66] Tony Z Zhao, Vikash Kumar, Sergey Levine, and Chelsea Finn. Learning fine-grained bimanual manipulation with low-cost hardware. *arXiv preprint arXiv:2304.13705*, 2023.
- [67] Yifeng Zhu, Zhenyu Jiang, Peter Stone, and Yuke Zhu. Learning generalizable manipulation policies with object-centric 3d representations. *ArXiv*, abs/2310.14386, 2023.
- [68] Yifeng Zhu, Zhenyu Jiang, Peter Stone, and Yuke Zhu. Learning generalizable manipulation policies with object-centric 3d representations. *arXiv preprint arXiv:2310.14386*, 2023.

Expanding Spatial and Temporal Context for Robotic Imitation Learning With Scene Graphs

Supplementary Material

6. Additional Information about Our Task-Driven Scene Graph

Concretely, our scene graph is implemented as a *two-level tree* in which the root node is represented by the CLS token extracted from the DINO-v2 encoder applied to the current image observation. The second level of the tree consists of the set of task-relevant object nodes described in Section 3.1. Although this structure lacks the deeper hierarchical layers (e.g. room $\rightarrow$ objects), we find that a shallow, two-tier topology is fully sufficient for the mobile manipulation tasks considered in this work, where the primary need is to reason about objects with respect to the robot’s current view. Importantly, this formulation remains fully compatible with richer multi-level hierarchies as explored in CLIO [36] and HODOR [41]; future extensions of our system could incorporate room-level or task-semantic nodes by simply adding additional parent layers above our current root node.

From the perspective of the downstream transformer encoder, the parent (root) node is always supplied as the *first token* in the sequence. Consistent with standard transformer architectures, we attach a positional embedding to each node token, including both the root and all object nodes. This positional encoding gives the model an explicit mechanism to distinguish the structural role of each node and allows the transformer to reliably identify the root as the global context provider. As a result, even in the absence of explicit relational edges, the model learns to interpret the object nodes as children conditioned on a shared scene-level parent, enabling effective reasoning over this minimal yet structured two-level scene graph.

Prompt and Entity Lists for Our Tasks In this paper, we use natural task descriptions to prompt a large language model (LLM) for a list of relevant objects. Identifying the right entities (e.g., “microwave” and “cup”) given task prompts (e.g., “Heat up a cup of tea”) is very easy for modern LLMs, so we used very simple prompts: “Imagine you are a robot performing the task “(insert-task-name)”, what are the task-relevant objects?” With GPT-4, this works with 100% accuracy across all our tasks. We include the results from GPT-4 for all tasks in Fig 7(a)-(f).

7. Technical Details for Collecting Demonstrations in Simulated Mujoco Environment

Low-Level Controller. During demonstration collection, we train a locomotion controller in simulation. Following

the MDP formulation of [45], we employ Proximal Policy Optimization (PPO) [47] within IsaacLab [38] to learn a robust quadruped walking policy.

The resulting low-level controller, denoted $\pi_{ll}(a_t | s_t, g_t)$, receives proprioceptive observations s_t along with a desired base velocity command g_t , and outputs joint-angle targets for the legs. The arm joints are controlled separately via a PD controller. To improve the robustness of the learned locomotion behavior, we randomize the arm configuration during training, enabling the robot to walk stably even when the arm is placed in arbitrary poses.

Mode Switching. During data collection, the robot operates in one of two regimes: *locomotion mode* and *manipulation mode*. In locomotion mode, the arm joints are frozen and π_{ll} controls only the leg velocities; in manipulation mode, the base remains stationary while the arm executes joint-space actions. Along with joint states and leg velocities, we record a mode indicator that specifies the robot’s operational regime.

To avoid abrupt discontinuities at switching boundaries—e.g., when the robot transitions between navigating and manipulating—we encode the mode using a temporally smoothed signal. For each mode transition occurring at time t , we generate a transition-specific contribution

$$\tilde{m}_k^{(t)} = \begin{cases} \sigma(k - t), & \text{if switching into manipulation,} \\ -\sigma(k - t), & \text{if switching into locomotion,} \end{cases}$$

where σ denotes the logistic sigmoid. The final mode value at timestep k is obtained by summing all contributions from past transitions:

$$m_k = \sum_{t \in \mathcal{T}} \tilde{m}_k^{(t)},$$

where $\mathcal{T}$ denotes the set of all transition times. This produces a smooth, continuous, and bidirectional encoding of the robot’s operational regime, stabilizing training and avoiding artifacts caused by instantaneous mode flips.

During training, the high-level policy predicts this smoothed mode value m_k jointly with all other arm and leg action outputs, enabling the policy to learn coordinated transitions between locomotion and manipulation behaviors. During test time, we convert the predicted mode value into a discrete operational mode via thresholding: if $m_k > 0.5$, the robot is considered to be in manipulation mode; otherwise, it is treated as being in locomotion mode.

Imagine you are a robot performing the task "throw trash", what are the task-relevant objects?

If you imagine you're a household robot asked to **perform the task "throw trash"**, the **task-relevant objects** are those that directly enable the robot to detect, pick up, transport, and dispose of waste.

Here's a clear breakdown:

1. The Trash Itself

- Loose trash items (paper, food waste, wrappers, bottles, cans)
- Trash bags (full or partially full)
- Small debris on the floor

2. Containers and Disposal Points

- Indoor trash bins
- Outdoor garbage bins or dumpsters
- Recycling bins (if sorting is required)
- Compost bins (if applicable)

(a) Throw trash

Imagine you are a robot performing the task "Heat up tea in microwave", what are the task-relevant objects?

For the task "**Heat up tea in microwave**", a robot would consider the following **task-relevant objects**—i.e., anything needed to locate, contain, manipulate, or heat the tea:

Primary Objects

- **Cup / Mug** (container for the tea)
- **Tea** (liquid tea or tea bag + water)
- **Microwave oven**
- **Microwave door**
- **Microwave turntable** (if manipulated or perceived)

(b) Heat up tea in the microwave

Imagine you are a robot performing the task "put pineapple in bowl", what are the task-relevant objects?

For the task "**put pineapple in bowl**", a robot would treat as **task-relevant** any objects it must perceive or interact with to locate, grasp, move, and place the pineapple.

Primary Objects

- **Pineapple**
(whole pineapple, pineapple slices, pieces, or chunks)
- **Bowl**
(target container)

(c) Put pineapple in bowl

Imagine you are a robot performing the task "feed toy animal apple or pineapple", what are the task-relevant objects?

For the task "**feed toy animal apple or pineapple**", a robot would treat as **task-relevant** any objects it must detect, choose between, grasp, move, and use to perform the feeding action.

Here is a clear breakdown:

Primary Objects

- **Toy animal** (the target recipient)
- **Apple** (whole, sliced, or piece)
- **Pineapple** (whole or pieces)

(d) Feed toy animal apple or pineapple

Imagine you are a robot performing the task "feed an pineapple to the giraffe", what are the task-relevant objects?

For the task "**feed a pineapple to the giraffe**", the task-relevant objects are:

Primary Objects

- **Pineapple** – the food item to be delivered.
- **Giraffe** – the recipient of the pineapple.

(e) Feed pineapple to giraffe

Imagine you are a robot performing the task "clean up toys and put them in the drawer", what are the task-relevant objects?

For the task "**clean up toys and put them in the drawer**," the task-relevant objects are those needed to decide *what to pick, where to place, and when to finish*:

Primary objects (task-critical)

- **Toys (all instances)** – objects to be collected
- **Drawer** – target location for placement

(f) Clean Table

8. Technical Details for Real World Tabletop Manipulation Experiments

We utilize a 7-DoF Franka robotic arm operating under a continuous joint-control action space at 15 Hz. A ZED Mini camera is mounted on the robot’s wrist, and the captured wrist images are encoded using DINO-v2. The resulting CLS token is incorporated as an additional input to the policy during training. Operating under velocity control, the robot’s action space comprises a 6-DoF joint velocity vector alongside a single gripper command representing open or close actions, yielding a 7-dimensional continuous action output from the policy. All policies are trained for 200 epochs using a single NVIDIA 3090 GPU. For each task, we collected 300 demonstrations. Qualitative comparisons across the three tasks can be found in the supplementary materials, which include a presentation containing demonstration videos for each method.

Figure 7. GPT-4 responses for task-relevant object name identification