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 $\pi_{ll}(a_t \mid s_t, g_t)$, 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.

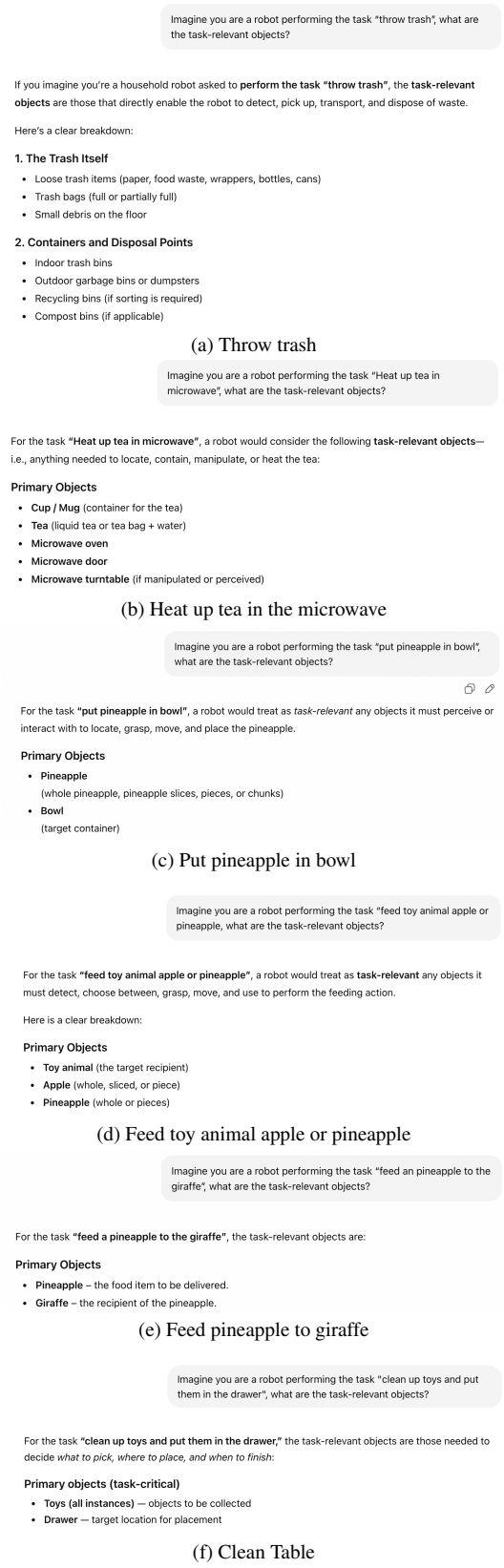

Figure 7. GPT-4 responses for task-relevant object name identification

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.