TASK-GRAPH-GUIDED DEMONSTRATION GENERATION FOR MULTI-ROBOT MANIPULATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Learning visuomotor policies for long-horizon multi-robot manipulation requires demonstrations that capture both reliable object interaction and inter-robot coordination. However, collecting coordinated demonstrations through human teleoperation can be costly and increasingly difficult as the number of robots grows. Script-based demonstrations can reduce reliance on human operators, but they impose synchronization that introduces unnecessary waiting. We present Graph2Demo, a task-graph-guided demonstration-generation framework that combines symbolic planning with residual reinforcement learning. Given a task specification in PDDL, Graph2Demo utilizes a task graph to encode the plan into per-agent subtask sequences and inter-agent dependencies, allowing robots to progress independently and synchronize when needed. To guide subtask execution, the task graph provides target poses for a pretrained pose-reaching base policy and subtask-specific rewards for learning a residual policy to correct the action for object interaction. By combining the base policy and the residual policy, we could generate demonstrations for multi-robot manipulation tasks. Averaged across the 11 tasks from the RoboFactory benchmark, diffusion policies trained on Graph2Demo demonstrations achieve higher success rates than those trained on demonstrations from the baselines, highlighting the potential of task-graph-guided demonstration generation for multi-robot manipulation. Code is available anonymously at `https://anonymous.4open.science/r/Graph2Demo-3D72/`.

1 INTRODUCTION

Behavior cloning (BC) has enabled effective visuomotor policy learning from demonstrations, with applications in single-arm manipulation (Mandlekar et al., 2021; Chi et al., 2023), fine-grained bimanual manipulation (Zhao et al., 2023; Jiang et al., 2025), and bimanual mobile manipulation (Fu et al., 2024). However, extending these successes to long-horizon tasks involving multiple robots places additional demands on the demonstrations: they must capture not only how each robot manipulates objects, but also how its actions depend on the progress of other robots. For example, in multi-robot assembly, robots may fetch components independently but must coordinate when handing them to one another or jointly lifting heavy loads. The resulting challenge is to obtain demonstrations that capture both reliable object interaction and the inter-robot dependencies required for coordinated task execution.

Collecting coordinated demonstrations through human teleoperation requires significant effort and specialized infrastructure, scaling poorly as the number of robots increases (Mandlekar et al., 2023; Chen et al., 2026b). Automated demonstration generation via scripted controllers and motion planners offers an appealing alternative to mitigate human collection costs (Mu et al., 2025; Qin et al., 2025). However, scripted trajectories could potentially inherit hand-designed execution order and synchronization rules. These execution rules introduce idle periods and rigid waiting behaviors into the generated trajectories. Reinforcement learning (RL) provides a way to improve execution through interaction and feedback, and learned policies can serve as demonstration generators for subsequent policy training (Xu et al., 2024). However, learning long-horizon multi-robot manipulation from scratch requires discovering both fine-grained physical manipulation and inter-robot dependencies, resulting in severe sample inefficiency and exploration bottlenecks (Cheng & Xu, 2023; Zhou et al., 2024; Dalal et al., 2024).

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

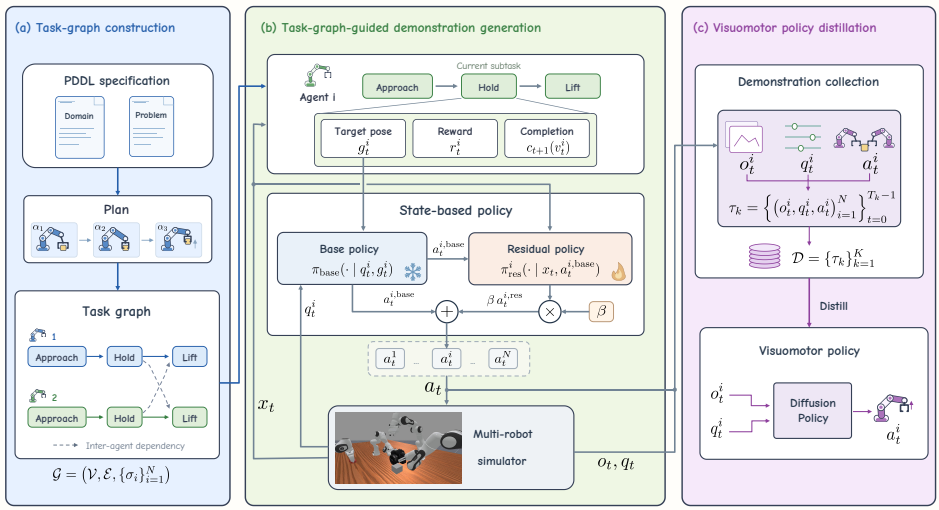


Figure 1: Overview of Graph2Demo. (a) Task-graph construction: a symbolic planner produces a plan from a PDDL specification, which is converted into a task graph encoding per-agent subtask sequences and inter-agent dependencies. (b) Task-graph-guided demonstration generation: each agent's current subtask specifies a target pose, reward, and completion check. A frozen pose-reaching base policy supplies goal-directed actions, while a learned residual policy adds action corrections scaled by β. The combined actions are executed in the multi-robot simulator, whose feedback guides subtask progression and policy learning. (c) Visuomotor policy distillation: demonstrations collected from the learned state-based policies record images, proprioceptive states, and executed actions for all agents, and are used to train per-agent visuomotor diffusion policies.

We present Graph2Demo, a task-graph-guided demonstration-generation framework for multi-robot manipulation (Figure 1). Given a task specification in the Planning Domain Definition Language (PDDL) (Ghallab et al., 1998), a symbolic planner produces an action sequence, and then we convert it into a graph of per-agent subtasks and inter-agent dependencies. Predicate-based checks determine when each subtask is complete and provide intermediate rewards for learning. Each robot advances to its next subtask once the graph's dependencies are satisfied. The graph also selects end-effector targets derived from object annotations, following RoboTwin and RoboFactory (Mu et al., 2025; Qin et al., 2025). A pretrained pose-reaching policy supplies goal-directed motion, while residual RL (Johannink et al., 2019) learns action corrections for object interaction. We then collect trajectories from the resulting state-based policies to train visuomotor diffusion policies (Chi et al., 2023).

We evaluate Graph2Demo on all 11 RoboFactory tasks (Qin et al., 2025), spanning one to four robots. Diffusion policies trained on demonstrations collected by Graph2Demo achieve higher success rates and require fewer completion steps. We further design baselines for a fine-grained analysis of the factors contributing to its success, and conduct an ablation study to demonstrate the necessity of our residual RL.

2 Related Work

2.1 Demonstration generation for manipulation

Automated demonstration generation reduces the need for human data collection by adapting existing demonstrations or synthesizing new task executions. Existing methods generate new demonstrations by transforming recorded trajectories to match new object configurations, including in bimanual tasks (Mandlekar et al., 2023; Jiang et al., 2025), and can also synthesize corresponding observations (Xue et al., 2025). Scripted and planning-based methods generate demonstrations from task procedures and geometric constraints (Mu et al., 2025; Chen et al., 2026a; Qin et al., 2025).

Another approach trains robot policies with reinforcement learning (RL) and collects their rollouts as demonstrations for policy training (Xu et al., 2024; Li et al., 2026). Rather than learning control from scratch, residual RL learns corrections to existing manipulation policies (Xiao et al., 2026; Zhu et al., 2025) or planning-based controllers (Jin et al., 2026) to generate demonstrations. Graph2Demo also uses RL to train demonstration-generating policies, but focuses on long-horizon tasks requiring multi-robot coordination. A task graph guides subtask learning and allows each robot to advance only when its prerequisites are satisfied. We evaluate the generated demonstrations by training visuomotor policies on them and measuring their task success.

2.2 Task planning for long-horizon tasks

Task planning breaks long-horizon manipulation into manageable subtasks. Early methods combined classical symbolic planning with motion planning to account for geometric constraints (Kaelbling & Lozano-Pérez, 2011; Garrett et al., 2017; 2020). More recent methods use large language models (LLMs) to choose robot skills and determine their execution order based on feasibility estimates or environment feedback (Ahn et al., 2022; Huang et al., 2022). Hybrid approaches such as LLM+P (Liu et al., 2023) translate natural-language problems into PDDL and delegate plan search to classical planners. Beyond sequencing predefined skills, symbolic and language-guided planning can also guide reinforcement learning of manipulation skills (Cheng & Xu, 2023; Dalal et al., 2024; Zhou et al., 2024).

With multiple robots, planners must also allocate subtasks to each agent, and coordinate the execution under temporal and spatial constraints. Existing methods use LLMs for task decomposition and allocation (Kannan et al., 2023), combine LLM-based decomposition with symbolic planning (Zhang et al., 2025), or use graphs for dependency-aware scheduling (Obata et al., 2025; Su et al., 2026) and asynchronous execution (Huang et al., 2025). Our method brings planning-guided learning and dependency-aware execution together to train multi-robot demonstration generators. Its PDDL-derived task graph defines subtask rewards and gates each robot's progress on prerequisite completion; it does not provide a collision-checked motion schedule.

2.3 Residual policy learning in robotics

Residual policies learn action corrections on top of a base controller via reinforcement learning, leveraging the base controller's capabilities while compensating for its limitations (Silver et al., 2018; Johannink et al., 2019). The base controller can be a policy initialized via behavior cloning, while the residual policy is further trained using visual observations and sparse task rewards (Alakuijala et al., 2021). ResiP (Ankile et al., 2025) applies per-step residual actions to chunked actions from a diffusion policy, improving execution reliability over long horizons in a closed-loop manner. Beyond direct control, residual policy learning also facilitates demonstration generation by refining learned policies or planning priors (Xiao et al., 2026; Zhu et al., 2025; Jin et al., 2026). In this work, Graph2Demo employs a frozen pose-reaching policy as its base and trains a separate residual policy for each robot to learn object interactions. A task graph then selects pose targets, enabling the same reaching base to be reused throughout multi-robot tasks.

3 Preliminaries

3.1 PDDL fundamentals

Classical planning seeks a plan $\rho = (\alpha_1, \ldots, \alpha_L)$ of length L that transforms an initial state s_0 into a state satisfying a goal condition g. The Planning Domain Definition Language (PDDL) (Ghallab et al., 1998) is a formal language for specifying such problems. A reusable domain $\mathcal{D} = (\mathcal{T}, \Phi, \mathcal{A})$ specifies types $\mathcal{T}$, predicates Φ, and actions $\mathcal{A}$, while a problem instance $\mathcal{P} = (\mathcal{O}, s_0, g)$ specifies typed objects $\mathcal{O}$, the initial state s_0, and the goal condition g.

A symbolic state s is represented by the set of grounded predicates that hold. Under the closed-world assumption, any predicate absent from the state is false. An action $\alpha \in \mathcal{A}$ is characterized by a precondition set $\mathrm{pre}(\alpha)$, an add-effect set $\mathrm{eff}^+(\alpha)$, and a delete-effect set $\mathrm{eff}^-(\alpha)$, which are sets of grounded predicates. When $\mathrm{pre}(\alpha) \subseteq s$, the action is applicable and induces the symbolic

transition

$$f(s, \alpha) = \big(s \setminus \text{eff}^-(\alpha)\big) \cup \text{eff}^+(\alpha)\,. \tag{1}$$

A plan is valid if, for each $l = 1, \ldots, L$, action α_l is applicable in s_{l-1}, the state evolves as $s_l = f(s_{l-1}, \alpha_l)$, and the final state satisfies the goal condition, $g \subseteq s_L$.

3.2 Problem setup

We study long-horizon manipulation tasks in which N robots cooperate to achieve a shared goal. We assume all tasks share the same PDDL domain $\mathcal{D}$ and each task has a problem instance $\mathcal{P}$ specifying the task objects, symbolic initial state, and goal condition. We also assume access to a simulator that provides state information during RL training and demonstration generation. Our objective is to generate multi-robot demonstrations for learning visuomotor policies. To this end, we train state-based policies using reinforcement learning and collect demonstrations from their execution. These demonstrations are then used to train visuomotor policies to perform the tasks using visual and proprioceptive observations.

4 Method

4.1 Overview

Our framework generates demonstrations for learning visuomotor policies on long-horizon multi-robot tasks. Given a PDDL task specification, the PDDL planner produces an action sequence. This action sequence is then converted into a task graph, encoding per-agent subtask sequences and inter-agent dependencies. This task graph coordinates multi-robot execution through predicate-based completion checks and specifies targets for low-level control. A pretrained pose-reaching policy provides the basic motion capability, while a residual policy is trained to correct the action for object interaction. We then collect trajectories from the learned state-based policies to train visuomotor policies.

4.2 Task graph representation

We use a graph to schedule the subtasks for each agent. This graph allows agents to proceed independently while encoding dependencies among subtasks.

Task-graph representation. A task graph is defined by

$$\mathcal{G} = \big(\mathcal{V}, \mathcal{E}, \{\sigma_i\}_{i=1}^{N}\big), \tag{2}$$

where $\mathcal{V}$ is the set of subtask nodes, $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ contains directed dependency edges, and σ_i is the ordered subtask sequence for agent i.

Given a PDDL plan $\rho = (\alpha_1, \ldots, \alpha_L)$, we convert it into a task graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \{\sigma_i\}_{i=1}^{N})$. In our formulation, each PDDL action involves one or more agents. For a PDDL action α, we create one subtask node $v \in \mathcal{V}$ for each participating agent. A joint action therefore produces multiple subtask nodes with shared preconditions and effects. We then scan the actions in plan order and add a directed edge $(u, v) \in \mathcal{E}$ when an effect of a preceding node u establishes a precondition of node v. These edges encode dependencies within and across agents: an edge $(u, v) \in \mathcal{E}$ requires subtask u to be completed before the agent advances to subtask v. Each sequence σ_i contains the nodes assigned to agent i in plan order. Figure 2 illustrates how per-agent sequences and inter-agent dependencies permit independent progress while restricting transitions that require other agents' subtasks to be completed. Task graphs for all 11 tasks are provided in Appendix B.

Task-graph-guided RL. At timestep t, agent i is assigned subtask $v_t^i = \sigma_i[\ell_t^i]$, where ℓ_t^i denotes the current subtask index. Each agent produces a continuous actions a_t^i, forming the joint action $a_t = (a_t^1, \ldots, a_t^N)$ that transitions the simulator from state x_t to x_{t+1}. The simulator then evaluates completion and rewards for each agent's current subtask.

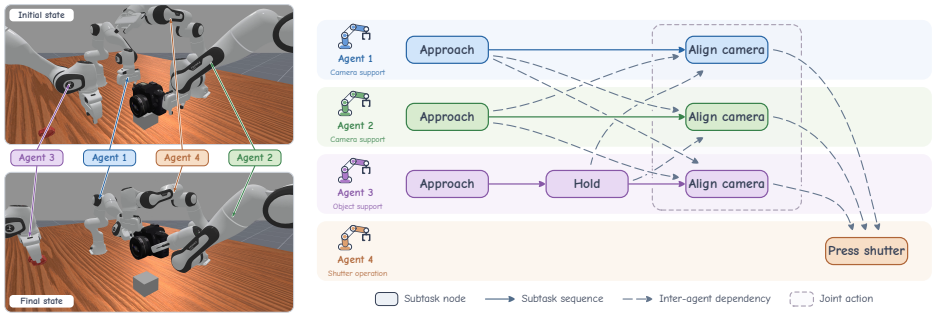


Figure 2: Example task graph for Take Photo. Left: visualization of the initial and final states. Right: each row shows one agent's subtask sequence. Solid arrows indicate subtask order, and dashed arrows indicate inter-agent dependencies. The dashed box groups the three *Align camera* nodes into a joint action involving the two camera-support agents and the object-support agent. The shutter-operation agent executes *Press shutter* after all three alignment nodes are completed.

A subtask v is completed if all its effect predicates $p \in \text{eff}(v)$ are satisfied.[1] Let $\chi(p, x_t)$ denote whether the predicate checker marks predicate p as satisfied in state x_t. Meanwhile, the agent receives a predicate reward for satisfying the predicate, denoted as $r(p, x_t)$. The subtask-level completion signal is denoted as $F(v, x)$, and the reward is denoted as $R(v, x)$:

$$F(v, x) = \bigwedge_{p \in \text{eff}(v)} \chi(p, x), \qquad R(v, x) = \sum_{p \in \text{eff}(v)} r(p, x). \tag{3}$$

We track the completion of each subtask v with a Boolean variable $c_t(v)$, which is initialized to be false and updated as $c_{t+1}(v_t^i) = c_t(v_t^i) \vee F(v_t^i, x_{t+1})$. After the current subtask $\sigma_i[\ell_t^i]$ is completed, the agent cannot advance to the next subtask $\sigma_i[\ell_t^i{+}1]$ until the predecessor subtasks $\text{Pred}(\sigma_i[\ell_t^i{+}1])$ are all completed, where $\text{Pred}(v) = \{u : (u, v) \in \mathcal{E}\}$. For $\ell_t^i < |\sigma_i|$, the index is updated as

$$\ell_{t+1}^i = \ell_t^i + \mathbf{1}\left[c_{t+1}(\sigma_i[\ell_t^i]) \wedge \bigwedge_{u \in \text{Pred}(\sigma_i[\ell_t^i+1])} c_{t+1}(u)\right], \tag{4}$$

where $\mathbf{1}[\cdot]$ denotes the indicator function. All agents begin at the first subtask node of their sequence, with $\ell_0^i = 1$ and all completion flags $c_0(v)$ set to false for all $v \in \mathcal{V}$. The episode terminates successfully once every agent has completed its final subtask.

Agents participating in the same joint action enter their corresponding subtasks at the same timestep, synchronizing the start of joint execution. They receive the same predicate reward because their nodes share the same effects, encouraging cooperation within the joint action. Together, these mechanisms coordinate joint actions while allowing independent subtasks to proceed asynchronously.

4.3 Task-Graph-Guided Residual RL

Learning manipulation from subtask rewards requires discovering both how to reach relevant poses and how to interact with objects. We separate these capabilities using a pretrained pose-reaching policy and a learned residual policy. The base policy provides goal-directed motion, while the residual policy learns action corrections for object interaction.

Residual action formulation. For each agent i, we combine a base action with a weighted residual action:

$$a_t^i = a_t^{i,\text{base}} + \beta a_t^{i,\text{res}}, \tag{5}$$

where β controls the magnitude of the residual action. The base action is sampled from a shared base policy $a_t^{i,\text{base}} \sim \pi_{\text{base}}(\cdot \mid q_t^i, g_t^i)$, where q_t^i is the proprioceptive state of agent i and g_t^i is its target

[1] For a node v corresponding to PDDL action α, we define $\text{eff}(v) = \text{eff}^+(\alpha) \cup \{\neg p : p \in \text{eff}^-(\alpha)\}$, the set of positive and negated effect predicates.

pose. The residual action is sampled from an individual residual policy $a_t^{i,\text{res}} \sim \pi_{\text{res}}^i(\cdot \mid x_t, a_t^{i,\text{base}})$, conditioned on the simulator state x_t and the base action $a_t^{i,\text{base}}$.

Base policy target. Object interaction requires appropriate end-effector targets. PSL (Dalal et al., 2024) estimates target positions from object segmentation and calibrated depth images. RoboFactory (Qin et al., 2025) follows RoboTwin's (Mu et al., 2025) annotation-based approach, computing grasp and manipulation target poses from annotated contact and functional points and their associated directions. Following RoboFactory, we use these object annotations to specify end-effector targets for object interaction.

The task graph selects among these annotated poses using the near predicate. If the predicate $\text{near}(\texttt{gripper}, \texttt{obj})$ appears in the effects of the current subtask, it means that the gripper has to move to a pose near the `obj` to complete the subtask. Naturally, we can use the near predicate in the effects of the subtask to select the target pose. In the initial state, each gripper is near an initial pose. If a subtask sets a new near predicate, it must negate the previous near predicate. In this way, exactly one near predicate must hold for each gripper. Let H denote the annotation-based mapping from a target object and the current simulator state to a target end-effector pose. At each timestep t, we first resolve the current near target `obj` for each agent i, calculate the target $g_t^i = H(\texttt{obj}, x_t)$, and then feed it to the base policy to generate the base action $a_t^{i,\text{base}} \sim \pi_{\text{base}}(\cdot \mid q_t^i, g_t^i)$.

Base policy training. We pretrain a shared base policy on a pose-reaching task using PPO (Schulman et al., 2017). Conditioned on the robot's proprioceptive state and a target end-effector pose, the policy outputs joint-position increments executed by a PD controller. The base policy is trained once from scratch and reused across robot agents and downstream manipulation tasks.

Residual policy training. For each downstream task, we train residual policies while keeping the pretrained base policy frozen. Actions are composed as in Equation 5. We train the residual policies simultaneously for all agents via IPPO (De Witt et al., 2020) with separate actor and critic networks. Each agent optimizes the reward of its current subtask, $r_t^i = R(\sigma_i[\ell_t^i], x_{t+1})$. Agents that share a joint action receive the same reward, encouraging cooperation, whereas agents assigned to different subtasks receive subtask-specific rewards to accelerate training.

4.4 Visual policy distillation

We distill the state-based policy into visuomotor diffusion policies using teacher-student distillation (Rusu et al., 2015). For each task, we collect K demonstrations using the state-based policy, forming a dataset $\mathcal{D} = \{\tau_k\}_{k=1}^K$. Each demonstration records observations and actions for all agents, $\tau_k = \{(o_t^i, q_t^i, a_t^i)_{i=1}^N\}_{t=0}^{T_k-1}$, where T_k is the length of the demonstration, o_t^i is the image observation from agent i's camera, q_t^i is its proprioceptive state, and a_t^i is the executed action defined in Equation 5. We train a separate diffusion policy for each agent to imitate these actions conditioned on its visual observations and proprioceptive state.

5 Experiments

5.1 Experimental setup

Benchmark. We evaluate on all 11 tasks from RoboFactory (Qin et al., 2025), a multi-agent manipulation benchmark built on the ManiSkill3 simulator (Tao et al., 2024). The tasks involve varying numbers of Franka Emika Panda arms, each with seven degrees of freedom and a parallel-jaw gripper. Successful collaboration requires coordinating object manipulation while respecting task dependencies, avoiding collisions in shared workspaces, and synchronizing actions when required.

Demonstration sources. We compare Graph2Demo with four scripted demonstration sources derived from RoboFactory (Qin et al., 2025). **RoboFactory scripts** (**Scripts**) use the original task-specific scripts and motion planner, preserving their execution order and synchronization rules. **Scripts with action noise** (**+ Noise**) add Gaussian noise with standard deviation 0.01 to the seven arm-joint action components during demonstration collection, while retaining the clean scripted

Table 1: Success rates of diffusion policies trained on Graph2Demo and baseline demonstrations on RoboFactory. Each task entry reports the percentage (%) of successful episodes out of 100 evaluation episodes. Since asynchronous execution adds no motion overlap beyond the original scripts, so + Async and + Noise & Async reuse the results from Scripts and + Noise. All results are reported as the average over 3 independent runs, accompanied by their standard deviations.

		Demonstration sources				
# of robots	Task	Scripts	+ Noise	+ Async	+ Noise & Async	Graph2Demo (ours)
1	Pick Meat	96.3 (2.5)	95.7 (2.1)	96.3 (2.5)	95.7 (2.1)	**99.7** (0.6)
	Stack Cube	75.3 (3.1)	56.7 (4.5)	75.3 (3.1)	56.7 (4.5)	**98.7** (1.5)
	Strike Cube	**98.3** (1.2)	91.3 (5.1)	**98.3** (1.2)	91.3 (5.1)	83.3 (4.7)
2	Pass Shoe	0.0 (0.0)	0.3 (0.6)	0.0 (0.0)	0.3 (0.6)	**82.0** (7.0)
	Place Food	**91.3** (3.2)	90.3 (3.5)	82.3 (7.4)	57.7 (8.1)	85.0 (2.0)
	Lift Barrier	94.7 (0.6)	87.3 (2.5)	94.7 (0.6)	87.3 (2.5)	**97.7** (1.5)
	Two Robots Stack Cube	68.7 (5.7)	63.3 (2.1)	34.3 (3.8)	54.7 (4.0)	**86.7** (3.5)
3	Camera Alignment	97.7 (1.5)	97.7 (1.2)	**99.0** (0.0)	**99.0** (0.0)	53.0 (3.0)
	Three Robots Stack Cube	0.3 (0.6)	18.0 (8.5)	2.7 (1.2)	38.0 (6.1)	**65.7** (2.3)
4	Take Photo	20.3 (5.5)	53.0 (10.1)	0.7 (1.2)	**79.3** (3.1)	48.3 (3.8)
	Long Pipeline Delivery	0.0 (0.0)	12.0 (4.6)	0.0 (0.0)	0.0 (0.0)	**48.3** (5.1)
	Average	58.5 (1.3)	60.5 (0.8)	53.1 (0.8)	60.0 (1.2)	**77.1** (0.9)

actions as the policy-training targets. **Scripts with asynchronous execution** (**+ Async**) overlap collision-free robot motions to reduce unnecessary waiting; for example, in Two Robots Stack Cube, one robot can begin moving while the other retreats, rather than waiting for the retreat to finish. **Scripts with action noise and asynchronous execution** (**+ Noise & Async**) combine both modifications. Together, these baselines test whether action noise, reduced synchronization overhead, or their combination can account for differences in the visuomotor policy performance relative to Graph2Demo. For each task and demonstration-generation method, we collect 150 successful demonstrations. We assess demonstration quality by training diffusion policies on each demonstration source with the same settings.

Implementation details. For PDDL planning, we use Fast Downward (Helmert, 2006) as the planner through the Unified Planning Python library (Micheli et al., 2025). For task-graph-guided residual reinforcement learning, we use 1,024 parallel environments with a training budget of 100 million environment transitions per task. For visuomotor learning, we train a separate diffusion policy (Chi et al., 2023) for each robot. We use the same policy architecture and training configuration across demonstration-generation methods, training each policy for 12,000 gradient steps with a batch size of 64. At each action step, each policy outputs seven arm-joint control commands and one gripper command. Each policy conditions on 3 observation steps, predicts an 8-step action chunk, and executes the 6 steps before replanning. Detailed training hyperparameters are provided in Appendix A.

5.2 Main results

Visuomotor policy performance. Table 1 shows that policies trained on Graph2Demo demonstrations achieve the highest average success rate across RoboFactory. Averaged across all 11 tasks, Graph2Demo achieves 77.1% mean success rate, compared with 60.5% for the strongest scripted demonstration source, Scripts + Noise. Graph2Demo yields the highest mean success rate on 7 of the 11 tasks, including 99.7% on Pick Meat and 98.7% on Stack Cube. On challenging multi-robot tasks, it substantially outperforms the best scripted baseline, achieving 48.3% versus 12.0% on Long Pipeline Delivery and 65.7% versus 38.0% on Three Robots Stack Cube. However, these gains are not uniform. Graph2Demo trails the best scripted baseline by a large margin on Camera Alignment and Take Photo.

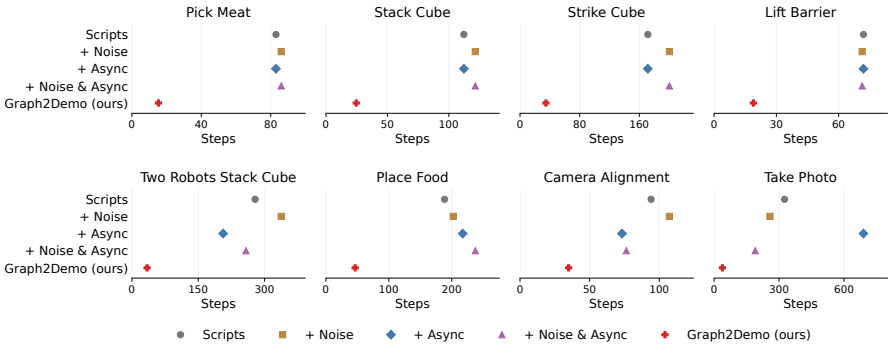


Figure 3: Mean steps to success for diffusion policies trained on Graph2Demo and baseline demonstrations. We average completion steps over successful episodes from 100 evaluation runs. For single-robot tasks and Lift Barrier, asynchronous execution adds no motion overlap beyond the original scripts, so + Async and + Noise & Async reuse the results from Scripts and + Noise, respectively.

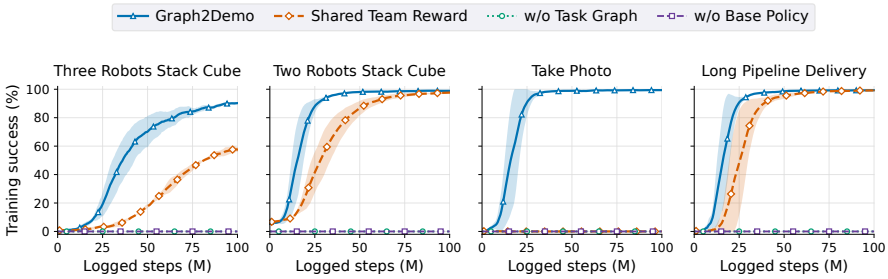


Figure 4: Training success rates of task-graph-guided residual RL and ablations on 4 tasks.

Execution efficiency. Figure 3 compares mean completion steps over successful evaluation episodes for eight tasks. Graph2Demo-trained policies have substantially lower mean completion steps than every scripted baseline on all eight tasks. For example, on Two Robots Stack Cube, the mean decreases from 206.3 steps for the fastest scripted baseline to 34.4 steps for Graph2Demo, while success rate increases from 30% to 87%. However, shorter successful rollouts do not always coincide with a higher success rate. On Camera Alignment and Take Photo, Graph2Demo requires fewer completion steps but has a lower success rate than the strongest scripted variants.

5.3 Analysis

Task-graph-guided residual RL ablation. Figure 4 compares the training success rates of the task-graph-guided residual RL and three ablations on 4 complex tasks. **Graph2Demo** denotes the full task-graph-guided residual RL used in data generation. **Shared Team Reward** replaces subtask rewards with a shared reward averaged across agents. **w/o Task Graph** removes subtask-specific rewards and provides only the final task-completion reward. **w/o Base Policy** removes the pretrained pose-reaching policy and learns control actions from scratch.

The full task-graph-guided residual RL reaches approximately 90% training success on Three Robots Stack Cube and nearly 100% on the other three tasks, whereas removing either the task graph or the base policy results in zero success rates across all four tasks. These results suggest that both components are important for learning on long-horizon multi-robot manipulation tasks, where the task graph supplies intermediate objectives for long-horizon coordination, while the base policy provides goal-directed motion on which manipulation behavior can be learned. Sharing rewards across agents exhibits slower improvement in the success rates, suggesting that learning from subtask rewards can improve sample efficiency.

Table 2: Mean demonstration length (in steps) for Graph2Demo and scripted baselines across all 11 RoboFactory tasks. Graph2Demo yields the shortest demonstrations on average for every task. Dashes indicate tasks for which asynchronous execution leaves the original scripts unchanged.

Sources	Pick Meat	Stack Cube	Strike Cube	Pass Shoe	Place Food	Lift Barrier	Two Robots Stack Cube	Camera Alignment	Three Robots Stack Cube	Take Photo	Long Pipeline Delivery
Scripts	70.49	104.45	172.61	365.85	202.87	69.05	249.62	97.79	424.29	187.13	721.71
+ Noise	70.85	103.63	172.17	363.92	197.88	68.91	250.12	97.78	441.29	194.21	715.26
+ Async	—	—	—	327.57	212.71	—	202.36	73.38	315.06	152.76	618.99
+ Noise & Async	—	—	—	331.07	203.81	—	204.15	73.23	323.01	153.55	615.91
Graph2Demo	15.13	24.39	35.45	71.01	46.68	19.34	46.71	36.75	81.09	43.45	111.75

Effect of action noise. Adding action noise during demonstration collection improves average downstream policy success under both the original and asynchronous execution schedules. As shown in Table 1, noise increases the average success rate from 58.5% to 60.5% for the original scripts and from 53.1% to 60.0% for asynchronous scripts. Several three- and four-robot tasks benefit substantially: on Three Robots Stack Cube, success rises from 0.3% to 18.0% under the original schedule and from 2.7% to 38.0% under asynchronous execution; on Long Pipeline Delivery, it increases from 0.0% to 12.0% under the original schedule. However, these gains are accompanied by lower success on some single- and two-robot tasks.

Effect of asynchronous execution. Asynchronous execution alone is not sufficient to improve downstream policy success. Without action noise, it decreases the average success rate from 58.5% to 53.1%. With action noise, it improves success on some tasks, notably Three Robots Stack Cube (18.0% to 38.0%) and Take Photo (53.0% to 79.3%), but the overall average remains slightly lower (60.5% to 60.0%). Besides, Figure 3 shows that asynchronous execution can shorten successful rollouts: on Two Robots Stack Cube, mean completion steps decrease from 278.5 to 206.3 without noise and from 337.6 to 257.8 with noise. Overall, asynchronous execution can shorten successful rollouts by reducing unnecessary waiting, but its effect on downstream policy success rates varies across tasks and noise settings.

Effect of demonstration length. Table 2 shows that Graph2Demo produces the shortest mean demonstration length on all 11 tasks. This is expected, as RL can minimize trajectory length if the completion reward is discounted. This pattern is also mirrored in Figure 3, where policies trained on these demonstrations also require fewer steps to complete than all scripted baselines. However, shorter demonstrations do not necessarily translate into higher success rates: on Camera Alignment and Take Photo, Graph2Demo has both shorter demonstrations and shorter successful rollouts, yet the success rates are lower than the strongest scripted baseline.

6 Conclusion and Future Work

We presented Graph2Demo, a task-graph-guided demonstration-generation framework for multi-robot manipulation. Graph2Demo converts PDDL plans into task graphs that encode per-agent subtask sequences and inter-agent dependencies, guiding residual reinforcement learning with subtask-specific rewards and target pose selection. By combining a pretrained pose-reaching policy with learned residual corrections, the framework generates demonstrations for training visuomotor diffusion policies. Across the 11 RoboFactory tasks, these demonstrations yield the highest average success rate of the downstream policies compared with the scripted baselines. While task graph could effectively guide residual RL training, it is fixed once generated. Future work could enable online graph revision based on the current state to either recover from failures or adapt to changes in the environment. Beyond demonstration generation, task graphs could also guide reinforcement fine-tuning of visuomotor policies in simulation, using predicate checks computed from simulator state to provide subtask-level rewards for policy refinement beyond behavior cloning.

AI USE STATEMENT

In this work, we used generative AI tools to assist with code development, code reviewing, language polishing, and visualization. We have not used generative AI tools for idea and hypotheses generation. We have reviewed all AI-assisted work, e.g., we checked all the polished content against the empirical results. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REFERENCES

Michael Ahn, Anthony Brohan, Noah Brown, Yevgen Chebotar, Omar Cortes, Byron David, Chelsea Finn, Chuyuan Fu, Keerthana Gopalakrishnan, Karol Hausman, et al. Do as I can, not as I say: Grounding language in robotic affordances. In *6th Conference on Robot Learning (CoRL)*, volume 205 of *Proceedings of Machine Learning Research*, pp. 287–318. PMLR, 2022.

Minttu Alakuijala, Gabriel Dulac-Arnold, Julien Mairal, Jean Ponce, and Cordelia Schmid. Residual reinforcement learning from demonstrations. *arXiv preprint arXiv:2106.08050*, 2021. URL `https://arxiv.org/abs/2106.08050`.

Lars Ankile, Anthony Simeonov, Idan Shenfeld, Marcel Torne, and Pulkit Agrawal. From imitation to refinement-residual RL for precise assembly. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 01–08. IEEE, 2025.

Tianxing Chen, Zanxin Chen, Baijun Chen, Zijian Cai, Yibin Liu, Zixuan Li, Qiwei Liang, Xianliang Lin, Yiheng Ge, Zhenyu Gu, et al. RoboTwin 2.0: A scalable data generator and benchmark with strong domain randomization for robust bimanual robotic manipulation. In *International Conference on Machine Learning (ICML)*, 2026a.

Yincong Chen, Ranpeng Qiu, Zihao Li, Yanan Zhou, Guoqiang Ren, and Weiming Zhi. Robots that collaborate: Sequential asymmetric imitation for learning coupled robot policies. *arXiv preprint arXiv:2606.16490*, 2026b.

Shuo Cheng and Danfei Xu. LEAGUE: Guided skill learning and abstraction for long-horizon manipulation. *IEEE Robotics and Automation Letters*, 2023.

Cheng Chi, Siyuan Feng, Yilun Du, Zhenjia Xu, Eric Cousineau, Benjamin Burchfiel, and Shuran Song. Diffusion Policy: Visuomotor policy learning via action diffusion. In *Proceedings of Robotics: Science and Systems (RSS)*, 2023.

Murtaza Dalal, Tarun Chiruvolu, Devendra Chaplot, and Ruslan Salakhutdinov. Plan-Seq-Learn: Language model guided RL for solving long horizon robotics tasks. In *International Conference on Learning Representations (ICLR)*, 2024.

Christian Schroeder De Witt, Tarun Gupta, Denys Makoviichuk, Viktor Makoviychuk, Philip HS Torr, Mingfei Sun, and Shimon Whiteson. Is independent learning all you need in the StarCraft multi-agent challenge? *arXiv preprint arXiv:2011.09533*, 2020.

Zipeng Fu, Tony Z. Zhao, and Chelsea Finn. Mobile ALOHA: Learning bimanual mobile manipulation with low-cost whole-body teleoperation. In *Conference on Robot Learning (CoRL)*, 2024.

Caelan Reed Garrett, Tomás Lozano-Pérez, and Leslie Pack Kaelbling. FFRob: Leveraging symbolic planning for efficient task and motion planning. *The International Journal of Robotics Research*, 2017.

Caelan Reed Garrett, Tomás Lozano-Pérez, and Leslie Pack Kaelbling. PDDLStream: Integrating symbolic planners and blackbox samplers via optimistic adaptive planning. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 30, pp. 440–448, 2020.

Malik Ghallab, Adele Howe, Craig Knoblock, Drew McDermott, Ashwin Ram, Manuela Veloso, Daniel Weld, and David Wilkins. PDDL-the planning domain definition language. Technical report, Yale Center for Computational Vision and Control, 1998.

Malte Helmert. The Fast Downward planning system. *Journal of Artificial Intelligence Research*, 26:191–246, 2006.

Philip Huang, Ruixuan Liu, Shobhit Aggarwal, Changliu Liu, and Jiaoyang Li. APEX-MR: Multi-robot asynchronous planning and execution for cooperative assembly. In *Proceedings of Robotics: Science and Systems (RSS)*, 2025.

Wenlong Huang, Fei Xia, Ted Xiao, Harris Chan, Jacky Liang, Pete Florence, Andy Zeng, Jonathan Tompson, Igor Mordatch, Yevgen Chebotar, Pierre Sermanet, Noah Brown, Tomas Jackson, Linda Luu, Sergey Levine, Karol Hausman, and Brian Ichter. Inner monologue: Embodied reasoning through planning with language models. In *6th Conference on Robot Learning (CoRL)*, volume 205 of *Proceedings of Machine Learning Research*, pp. 1769–1782. PMLR, 2022.

Zhenyu Jiang, Yuqi Xie, Kevin Lin, Zhenjia Xu, Weikang Wan, Ajay Mandlekar, Linxi Jim Fan, and Yuke Zhu. DexMimicGen: Automated data generation for bimanual dexterous manipulation via imitation learning. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 16923–16930. IEEE, 2025.

Yiyang Jin, Yu Zheng, Xiao He, and Hesheng Wang. DynaForge: Planning-guided residual learning for dynamic manipulation demonstration generation. *arXiv preprint arXiv:2609.25631*, 2026.

Tobias Johannink, Shikhar Bahl, Ashvin Nair, Jianlan Luo, Avinash Kumar, Matthias Loskyll, Juan Aparicio Ojea, Eugen Solowjow, and Sergey Levine. Residual reinforcement learning for robot control. In *2019 International Conference on Robotics and Automation (ICRA)*, pp. 6023–6029. IEEE, 2019.

Leslie Pack Kaelbling and Tomás Lozano-Pérez. Hierarchical task and motion planning in the now. In *2011 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1470–1477, 2011.

Shyam Sundar Kannan, Vishnunandan L. N. Venkatesh, and Byung-Cheol Min. SMART-LLM: Smart multi-agent robot task planning using large language models. *arXiv preprint arXiv:2309.10062*, 2023.

Zechu Li, Yufeng Jin, Puze Liu, Jan Peters, and Georgia Chalvatzaki. Scalable multi-task data generation via reinforcement learning for language-conditioned bimanual dexterous manipulation. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2026.

Bo Liu, Yuqian Jiang, Xiaohan Zhang, Qiang Liu, Shiqi Zhang, Joydeep Biswas, and Peter Stone. LLM+P: Empowering large language models with optimal planning proficiency. *arXiv preprint arXiv:2304.11477*, 2023.

Ajay Mandlekar, Danfei Xu, Josiah Wong, Soroush Nasiriany, Chen Wang, Rohun Kulkarni, Li Fei-Fei, Silvio Savarese, Yuke Zhu, and Roberto Martín-Martín. What matters in learning from offline human demonstrations for robot manipulation. In *Conference on Robot Learning (CoRL)*, 2021.

Ajay Mandlekar, Soroush Nasiriany, Bowen Wen, Iretiayo Akinola, Yashraj Narang, Linxi Fan, Yuke Zhu, and Dieter Fox. MimicGen: A data generation system for scalable robot learning using human demonstrations. In *7th Conference on Robot Learning (CoRL)*, volume 229 of *Proceedings of Machine Learning Research*, pp. 1820–1864. PMLR, 2023.

Andrea Micheli, Arthur Bit-Monnot, Gabriele Röger, Enrico Scala, Alessandro Valentini, Luca Framba, Alberto Rovetta, Alessandro Trapasso, Luigi Bonassi, Alfonso Emilio Gerevini, Luca Iocchi, Felix Ingrand, Uwe Köckemann, Fabio Patrizi, Alessandro Saetti, Ivan Serina, and Sebastian Stock. Unified Planning: Modeling, manipulating and solving AI planning problems in Python. *SoftwareX*, 29:102012, 2025.

Yao Mu, Tianxing Chen, Zanxin Chen, Shijia Peng, Zhiqian Lan, Zeyu Gao, Zhixuan Liang, Qiaojun Yu, Yude Zou, Mingkun Xu, et al. RoboTwin: Dual-arm robot benchmark with generative digital twins. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 27649–27660. IEEE, 2025.

Kazuma Obata, Tatsuya Aoki, Takato Horii, Tadahiro Taniguchi, and Takayuki Nagai. LiP-LLM: Integrating linear programming and dependency graph with large language models for multi-robot task planning. *IEEE Robotics and Automation Letters*, 10(2):1122–1129, 2025.

Yiran Qin, Li Kang, Xiufeng Song, Zhenfei Yin, Xiaohong Liu, Xihui Liu, Ruimao Zhang, and Lei Bai. RoboFactory: Exploring embodied agent collaboration with compositional constraints. In *2025 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 10075–10085. IEEE, 2025.

Andrei A Rusu, Sergio Gomez Colmenarejo, Caglar Gulcehre, Guillaume Desjardins, James Kirkpatrick, Razvan Pascanu, Volodymyr Mnih, Koray Kavukcuoglu, and Raia Hadsell. Policy distillation. *arXiv preprint arXiv:1511.06295*, 2015.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Tom Silver, Kelsey Allen, Josh Tenenbaum, and Leslie Kaelbling. Residual policy learning. *arXiv preprint arXiv:1812.06298*, 2018.

Xiangyu Su, Juzhan Xu, Oliver van Kaick, Kai Xu, and Ruizhen Hu. IMR-LLM: Industrial multi-robot task planning and program generation using large language models. *arXiv preprint arXiv:2603.02669*, 2026.

Stone Tao, Fanbo Xiang, Arth Shukla, Yuzhe Qin, Xander Hinrichsen, Xiaodi Yuan, Chen Bao, Xinsong Lin, Yulin Liu, Tse-Kai Chan, et al. ManiSkill3: GPU parallelized robot simulation and rendering for generalizable embodied AI. *arXiv preprint arXiv:2410.00425*, 2024.

Wenli Xiao, Haotian Lin, Andy Peng, Haoru Xue, Tairan He, Yuqi Xie, Fengyuan Hu, Jimmy Wu, Zhengyi Luo, Linxi Fan, Guanya Shi, and Yuke Zhu. Self-improving vision-language-action models with data generation via residual RL. In *International Conference on Learning Representations (ICLR)*, 2026.

Charles Xu, Qiyang Li, Jianlan Luo, and Sergey Levine. RLDG: Robotic generalist policy distillation via reinforcement learning. *arXiv preprint arXiv:2412.09858*, 2024.

Zhengrong Xue, Shuying Deng, Zhenyang Chen, Yixuan Wang, Zhecheng Yuan, and Huazhe Xu. DemoGen: Synthetic demonstration generation for data-efficient visuomotor policy learning. In *Proceedings of Robotics: Science and Systems (RSS)*, 2025.

Xiaopan Zhang, Hao Qin, Fuquan Wang, Yue Dong, and Jiachen Li. LaMMA-P: Generalizable multi-agent long-horizon task allocation and planning with LM-driven PDDL planner. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 10221–10221. IEEE, 2025.

Tony Z. Zhao, Vikash Kumar, Sergey Levine, and Chelsea Finn. Learning fine-grained bimanual manipulation with low-cost hardware. In *Proceedings of Robotics: Science and Systems (RSS)*, 2023.

Zihan Zhou, Animesh Garg, Dieter Fox, Caelan Garrett, and Ajay Mandlekar. SPIRE: Synergistic planning, imitation, and reinforcement learning for long-horizon manipulation. In *Conference on Robot Learning (CoRL)*, 2024.

Kefei Zhu, Fengshuo Bai, YuanHao Xiang, Yishuai Cai, Xinglin Chen, Ruochong Li, Xingtao Wang, Hao Dong, Yaodong Yang, Xiaopeng Fan, and Yuanpei Chen. Dexflywheel: A scalable and self-improving data generation framework for dexterous manipulation. *Advances in Neural Information Processing Systems*, 38, 2025.

A HYPERPARAMETERS

A.1 TASK-GRAPH-GUIDED RESIDUAL REINFORCEMENT LEARNING

For base policy and residual policy training, we use the hyperparameters in Table 3.

Table 3: Hyperparameters for base-policy and residual RL.

Hyperparameter	Base policy	Residual policy
Algorithm	PPO	IPPO
Parallel environments	1,024	1,024
Environment-step budget	10^8	10^8
Minibatches per epoch	32	4
PPO epochs per update	5	5
Discount factor γ	0.99	0.99
GAE lambda λ	0.95	0.95
Clip param ϵ	0.2	0.2
Entropy coefficient	0	0
Optimizer	Adam	Adam
Actor learning rate	3×10^{-4}	3×10^{-4}
critic learning rate	3×10^{-4}	3×10^{-4}
MLP hidden layers	3	3
Hidden width	256	256
Activation	SiLU	SiLU
Initial log standard deviation	-0.5	-1.5
Residual coefficient β	—	1.0

A.2 VISUOMOTOR DIFFUSION POLICIES

For diffusion policy training, we use the hypaerparameters in Table 4, which is the same for all demonstration sources.

Table 4: Diffusion-policy hyperparameters.

Hyperparameter	Value
Gradient steps	12,000
Batch size	64
Observation history	3
Action prediction horizon	8
Executed action steps	6
Action dimension	8
Optimizer	Adam
Learning rate	10^{-4}
Maximum gradient norm	10.0
Learning-rate schedule	Cosine
Visual encoder	ResNet-18
Denoising network	Conditional U-Net
U-Net channel widths	$(512, 1024, 2048)$
Convolution kernel size	5
Diffusion-timestep embedding dimension	128
Diffusion scheduler	DDPM
Training diffusion timesteps	100
Inference denoising steps	100
Noise schedule	Cosine

B TASK GRAPH VISUALIZATION

The following figures present the task graphs for all 11 tasks from the RoboFactory benchmark, where subtask sequences, inter-agent dependencies, and joint actions are shown.

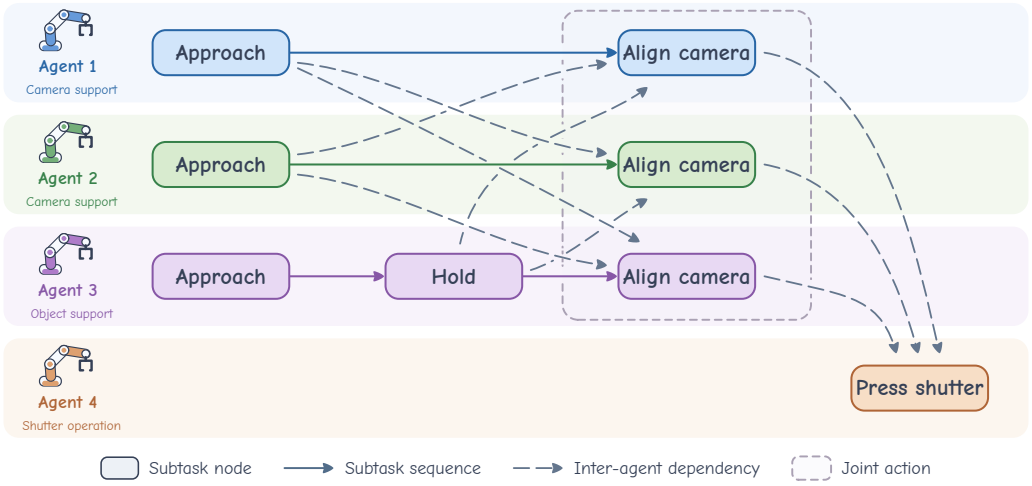


Figure 5: Task graph for Take Photo task.

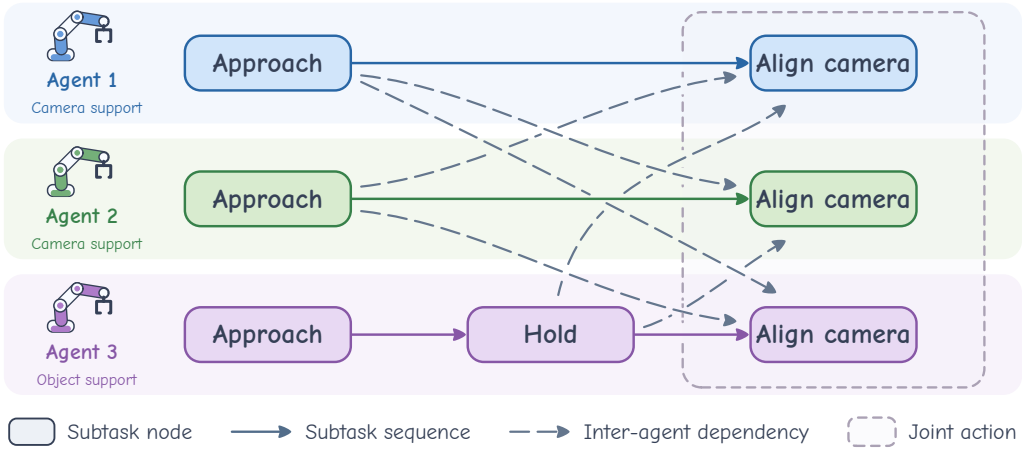


Figure 6: Task graph for Camera Alignment task.

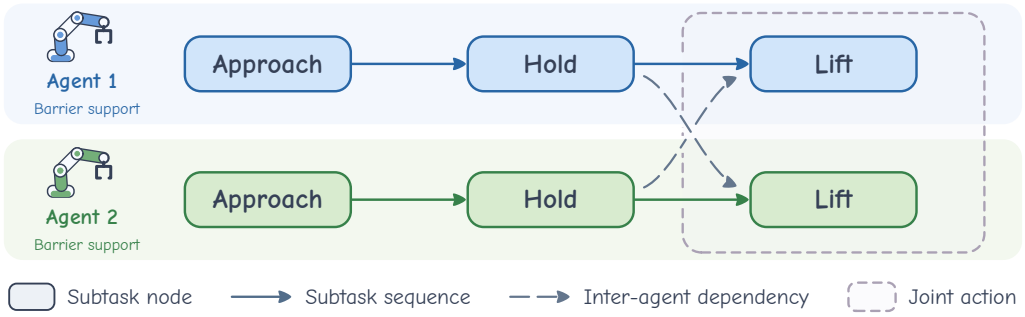


Figure 7: Task graph for Lift Barrier task.

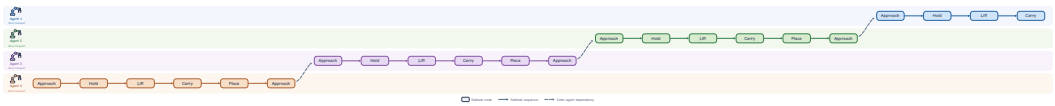


Figure 8: Task graph for Long Pipeline Delivery task.

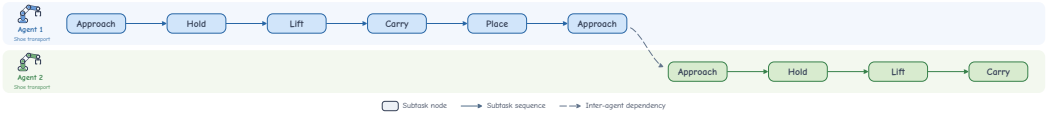


Figure 9: Task graph for Pass Shoe task.

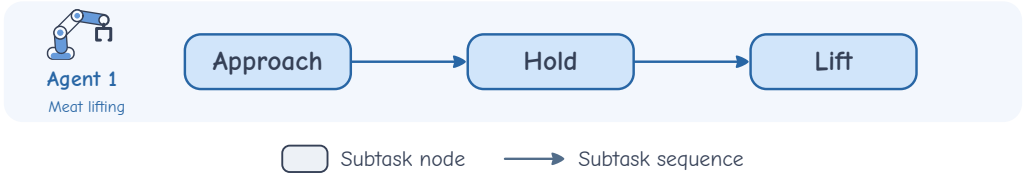


Figure 10: Task graph for Pick Meat task.

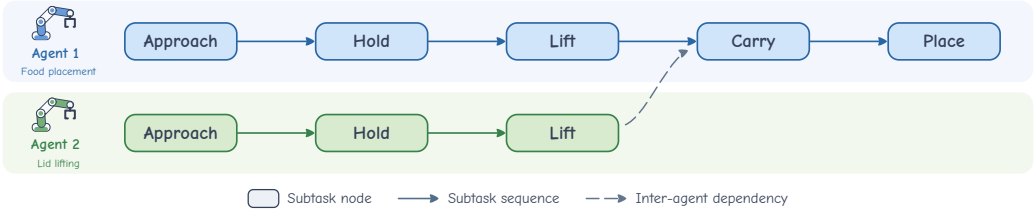


Figure 11: Task graph for Place Food task.

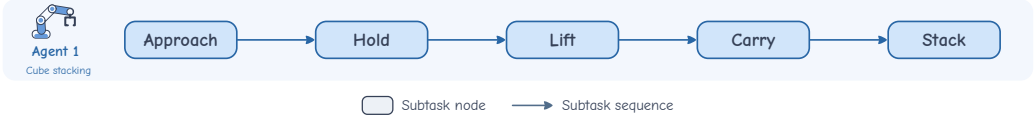


Figure 12: Task graph for Stack Cube task.

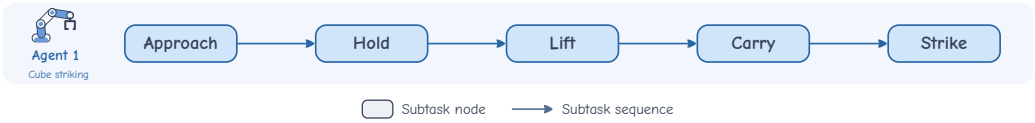


Figure 13: Task graph for Strike Cube task.

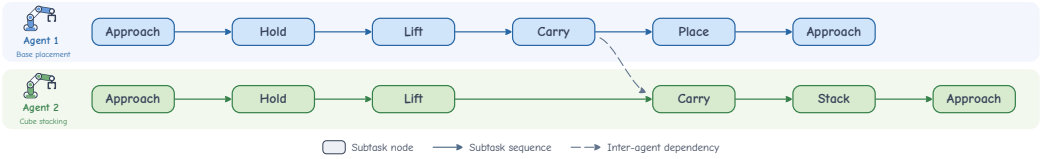


Figure 14: Task graph for Two Robots Stack Cube task.

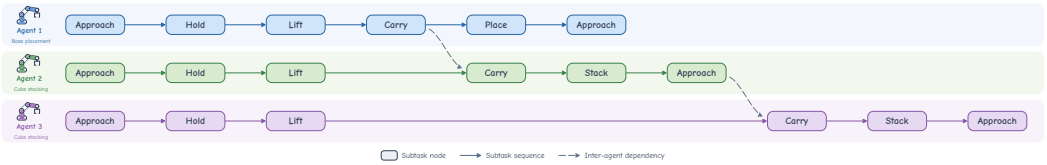


Figure 15: Task graph for Three Robots Stack Cube task.