TiPToP: A Modular Open-Vocabulary Robot Manipulation System That Plans

Author Names Omitted for Anonymous Review.

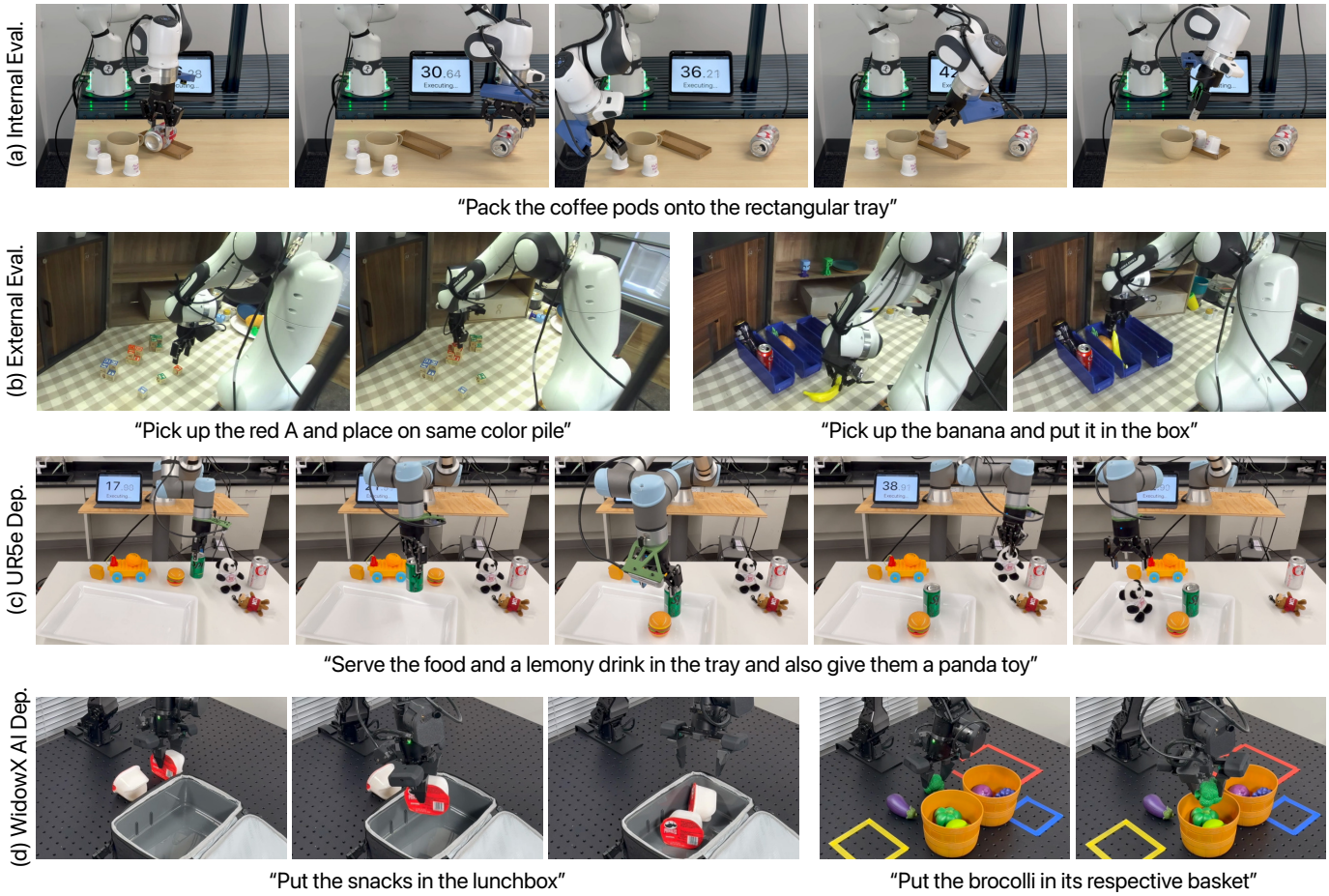


Fig. 3: **TiPToP operating over various environments and embodiments.** (a) TiPToP moves an obstructing Coke can out of the way to complete a long-horizon packing task in a DROID setup. (b) TiPToP solves two pick-and-place tasks requiring semantic understanding on an external DROID setup. (c) TiPToP deployed on a UR5e and (d) on a Trossen WidowX AI.

***Abstract*—We present TiPToP, a modular manipulation system that integrates pre-trained foundation models with a GPU-accelerated Task and Motion Planner to solve tasks directly from RGB images and natural language. TiPToP composes perception, planning, and execution modules and requires no robot training data. It can be deployed on a standard DROID setup in under an hour and adapted to new embodiments with minimal effort. We evaluate TiPToP against $\pi_{0.5}$-DROID, a state-of-the-art VLA fine-tuned on 350 hours of demonstrations, across two real-world DROID setups (one operated by an external team) and simulation, where TiPToP attains a higher average success rate and faster average completion time. We also evaluate on the MolmoSpaces benchmark, where TiPToP ranks first overall on pick and pick-and-place tasks among methods not trained on in-distribution data. We further show that TiPToP's modularity enables us to trace failures to specific components, revealing where to target improvements. We release TiPToP open-source to enable further research on modular manipulation and as a reproducible baseline.**

I. Introduction

A longstanding goal of robotics research is to build a general-purpose manipulation system: one that can be deployed on arbitrary robots and perform tasks specified in natural language on arbitrary objects, without requiring object, environment, or embodiment-specific tuning. Recent progress in vision and language foundation models has expanded the capabilities available to robotics, from open-vocabulary detection, grounding, and common-sense reasoning [2, 3, 4, 5] to depth estimation, grasp generation, and 3D shape completion [6, 7, 8, 9]. However, converting these capabilities into reliable manipulation behavior on real-world hardware remains an open problem.

We pursue this goal by combining the broad world knowledge and perception capabilities of pretrained foundation

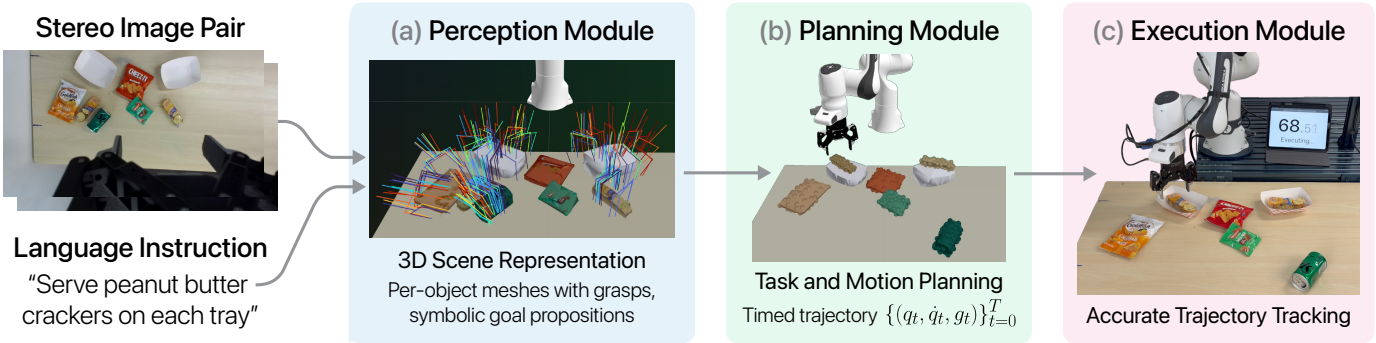


Fig. 4: **TiPToP System Overview.** TiPToP takes a stereo RGB image pair and a natural language instruction $\mathcal{L}$ as input and outputs robot joint trajectories with gripper commands. (a) The perception module constructs an object-centric 3D scene representation using learned depth estimation, grasp prediction, object detection, and segmentation. (b) The planning module uses GPU-parallelized TAMP (cuTAMP) [1] to find feasible manipulation plans. (c) The execution module tracks the planned trajectory using a joint impedance controller.

models with search-based planning at test-time via Task and Motion Planning (TAMP) [10]. In particular, foundation models supply perception and language grounding, while GPU-accelerated TAMP [1] supplies the low-level geometric and kinematic planning needed to produce executable robot trajectories. Neither component requires task or embodiment-specific training data.

We introduce **TiPToP** (**T**iPToP **i**s a **P**lanner **T**hat just works **o**n **P**ixels), a complete manipulation system built on three principles. First, it is *modular*: it has separate *perception*, *planning*, and *execution* modules that can be improved or replaced in isolation, and that let us trace failures to a specific module. Second, it is *compositional*: new skills require only symbolic predicates, an operator, and a parameterized controller, which the planner composes with existing skills automatically, without retraining. Finally, it is *zero-shot*: it is built from off-the-shelf models and transfers to new embodiments, scenes, and tasks within its skill set without any training or robot data. TiPToP can be deployed on supported robots[1] in under an hour with only camera calibration.

We evaluate TiPToP against state-of-the-art Vision-Language-Action (VLA) models [11, 12, 13, 14] and World Action Models (WAMs) [15, 16, 17] given the same image and language input. On real hardware, an external team independently deployed TiPToP on the DROID platform [18] and compared it against $\pi_{0.5}$-DROID [11], a VLA fine-tuned on 350 hours of demonstrations. TiPToP matches or exceeds its success rate at comparable or shorter completion times. On the independent MolmoSpaces benchmark [19], we evaluate on all 9 of its pick and pick-and-place tasks (9,000 total episodes). Despite using *zero* robot training data, on 6 of these 9 tasks TiPToP outperforms every VLA and WAM using no in-distribution (ID) training data, and even outperforms ID-trained models on *Pick & Place-NextTo*.

We trace TiPToP's failures to specific modules and find that the majority stem from unstable grasps under open-loop execution. These point toward integrating TiPToP's planning with learned, closed-loop policies that scale with data. TiPToP's compositionality and embodiment-agnostic design allows us to easily extend the system: we add a whiteboard-wiping skill on the DROID embodiment and demonstrate deployment on a UR5e and a Trossen WidowX AI.

To summarize, we make two contributions: (1) TiPToP, an open-source, extensible planning-based manipulation system that integrates foundation models with GPU-accelerated TAMP, requires no robot training data, and supports both real-world deployment and development in simulation [20], and (2) an empirical study and per-module failure analysis characterizing the complementary strengths and failure modes of planning-based and data-driven manipulation across real-world hardware, simulation, and an independent leaderboard. We release TiPToP as a reproducible baseline for future systems to build on.

II. Related Work

Foundation Models for Perception. Recent advances in vision foundation models have enabled robots to perceive diverse objects and scenes without task-specific training data. Stereo depth estimation models [21, 6, 22] predict dense depth maps from RGB image pairs. Foundation models for grasp generation [23, 24, 7] predict 6-DoF grasp poses from point clouds. SAM [25] and SAM-2 [26] provide promptable segmentation from bounding boxes or points, enabling precise object boundary delineation. Shape completion models such as SAM-3D [8] and RecGen [9] reconstruct full 3D object meshes and poses from partial RGB(-D) views. VLMs [2, 3, 4, 5] combine vision and language understanding to perform open-vocabulary object detection, visual reasoning, and language grounding, providing semantic scene understanding and enabling robots to interpret natural language instructions. SceneComplete [27] composes perceptual foundation models to build 3D scene representations from 2D observations. TiPToP leverages this idea and integrates grasp generation and a VLM with a TAMP system to enable full manipulation planning.

Vision-Language-Action Models. Building on vision-language foundation models, recent work has scaled end-to-end learning for robotic manipulation by training large models on diverse robot data. Many VLAs leverage VLM backbones

[1]The embodiment must possess a camera, gripper, URDF, and trajectory tracking controller to be supported.

to enable language-conditioned control [28, 29, 30, 31] and demonstrate that transformer-based policies trained on large datasets (e.g., the Open X-Embodiment dataset [32]) can generalize across tasks and objects.

π_0 [33] introduced a flow-matching architecture, which was subsequently extended in $\pi_{0.5}$ [11] via co-training on heterogeneous data sources to improve generalization. We compare against $\pi_{0.5}$-DROID, a variant fine-tuned on 350 hours of DROID demonstrations. A parallel line of work scales *simulator*-trained VLAs: MolmoBot [34] trains manipulation policies entirely in simulation using the MolmoSpaces ecosystem [19]. While these approaches can be applied across embodiments, show impressive generalization over scenes and objects, and can solve challenging tasks directly from pixels, they require substantial training data and are trained end-to-end, making it difficult to diagnose failures. By contrast, TiPToP composes pretrained models with no embodiment-specific training, and its modular structure lets failures be traced to individual components.

Task and Motion Planning. TAMP algorithms jointly solve discrete task planning and continuous motion planning problems [35, 10], providing a principled framework for combinatorial geometric reasoning that complements the open-vocabulary perception and grounding VLMs provide. Common approaches use sampling-based methods [36, 37] or optimization [38, 39] to identify the continuous values that satisfy the constraints. However, most TAMP systems require detailed object geometries, limiting their application primarily to environments in which the geometries are given *a priori*. The work most closely related to ours is by Curtis et al. [40], who integrated learned perception modules for property and affordance estimation with PDDLStream [37] to enable long-horizon manipulation of unknown objects in the real world. Our system differs in several critical ways: first, we use cuTAMP [1], a GPU-parallelized optimization-based TAMP algorithm, which provides significantly improved computational efficiency compared to sampling-based approaches like PDDLStream and search-then-sample bilevel planners [41, 42]; second, we leverage much larger foundation models trained on significantly more data than the perception models used in previous work.

Modular Robotic Planning Systems. Modular approaches to robot planning decompose manipulation into distinct components such as perception, high-level planning, and low-level control. Early symbolic systems like STRIPS [43], applied to the Shakey robot [44], demonstrated the power of symbolic plans but required detailed world models and controlled environments. More recently, neurosymbolic approaches use LLMs to sequence pre-trained skills [45, 46], generate robot programs [47, 48], construct 3D value maps [49], or combine VLMs with manipulation skills [50]. Other work integrates learning with TAMP: LLM^3 [51] uses LLMs to propose task plans with motion failure reasoning, PRoC3S [52] combines LLMs with continuous constraint satisfaction, VLM-TAMP [53] feeds VLM-generated subgoals to a task and motion planner, and OWL-TAMP [54] uses a VLM to infer constraints passed to a TAMP system. Compared to LLM-based systems that sequence discrete skills, TiPToP jointly solves discrete task plans and continuous collision-free trajectories, which is important when constraints on continuous parameters impact the feasibility of skill sequences.

III. Problem Formulation and System Overview

We consider language-conditioned manipulation: given a natural language instruction $\mathcal{L}$ and a robot with known kinematics, produce actions that accomplish the task. At each timestep t, the policy π receives RGB observations $\mathbf{o}_t$ from one or more cameras, the current joint configuration q_t, and outputs an action a_t. That is, $a_t = \pi(\mathbf{o}_t,\, q_t \mid \mathcal{L})$. This shared input-output specification enables direct empirical comparison across paradigms. The two systems we primarily compare instantiate it very differently.

A. $\pi_{0.5}$-DROID

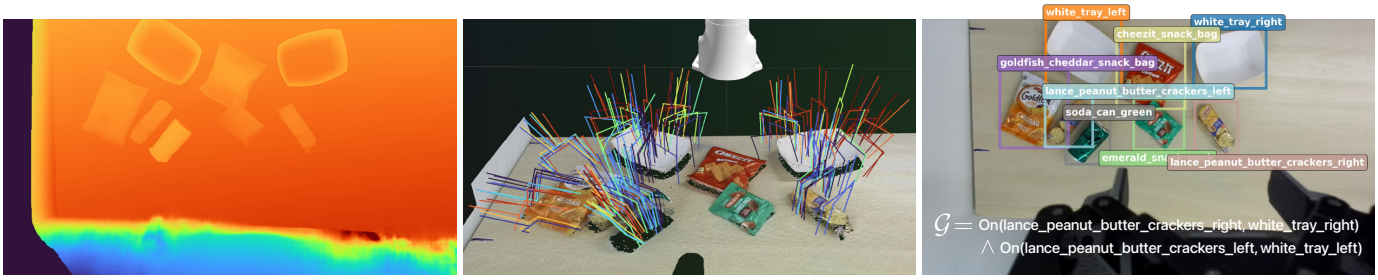
$\pi_{0.5}$-DROID [11] runs at 15 Hz. At each timestep it observes $\mathbf{o}_t = (I_t^{\text{wrist}}, I_t^{\text{ext}})$, monocular RGB images from the wrist and external cameras, along with the current joint and gripper positions (q_t, g_t). It emits chunks of 15 actions $(\dot{q}_{t:t+15}, g_{t:t+15})$, where $\dot{q}$ is a joint velocity command and $g \in \{0, 1\}$ is the binary gripper action.

B. TiPToP

TiPToP implements the policy using a *planner*: it observes the scene *once* at $t{=}0$ from a *calibrated* wrist camera at a capture pose, which we assume provides a good view of the workspace. The observation $\mathbf{o}_0 = (I_0^{\text{left}}, I_0^{\text{right}})$ is a stereo RGB image pair with known intrinsics K, camera-to-end-effector extrinsics $T_{\text{cam}}^{\text{ee}}$, and stereo baseline b. From this single observation, TiPToP produces a complete timed trajectory $a_0 = \{(q_t, \dot{q}_t, g_t)\}_{t=0}^{T}$, where q_t is a joint configuration, $\dot{q}_t$ a joint velocity, and $g_t \in \{0, 1\}$ a binary gripper action. This plan is then executed open-loop with no further visual observations.

Modular Architecture. TiPToP is composed of three modules (Figure 4): (1) the *perception module* (§IV) takes $\mathbf{o}_0$ and $\mathcal{L}$ and constructs an object-centric 3D scene representation with per-object meshes, candidate grasps, and a symbolic goal $\mathcal{G}$; (2) the *planning module* (§V) uses cuTAMP [1] to search over plan skeletons and optimize continuous parameters (grasp poses, placement poses, collision-free trajectories) to find a feasible plan; and (3) the *execution module* (§VI) tracks the planned trajectory open-loop using a joint impedance controller.

Illustrative Example. We illustrate TiPToP in the DROID setup (Franka FR3 with a ZED Mini stereo camera mounted on the wrist) in the following scenario: the robot is given the instruction *"serve peanut butter crackers on each tray"* and the scene in Figure 4. This task requires identifying peanut butter crackers among visually similar snacks (Goldfish, Cheez-Its), requiring cultural understanding and visual knowledge to distinguish them. Additionally, a Sprite can obstructs all grasps on the left peanut butter cracker package, complicating depth estimation due to its reflective surface and requiring the robot to move the can out of the way before grasping the crackers.



(a) Predicted Depth (b) Point Cloud with 6 DoF Grasps (c) Object Detection & Goal Grounding

Fig. 5: **Perception Module.** (a) Depth map predicted by FoundationStereo with sharp object boundaries. (b) Grasps predicted by M2T2 on the scene point cloud (colors correspond to grasp confidences). (c) Labeled object bounding boxes and symbolic goal $\mathcal{G}$ predicted by Gemini ($\mathrm{On}(a, b)$ specifies that object a should be placed on object or surface b).

IV. Perception Module

The perception module takes the initial observation $\mathbf{o}_0$, joint configuration q_0, and the language instruction $\mathcal{L}$ as input to produce an object-centric 3D scene representation consisting of per-object meshes with candidate grasps, along with symbolic goal propositions that ground $\mathcal{L}$ into the desired relations between objects. Two branches run in parallel: the *3D Vision Branch* (§IV-A) extracts scene geometry and grasps, while the *Semantic Branch* (§IV-B) identifies objects and grounds the task goal. Their outputs are then merged (§IV-C).

A. 3D Vision Branch

Depth Estimation. We use FoundationStereo [6], a foundation model for stereo depth estimation, to predict a dense depth map D from the stereo RGB pair $\mathbf{o}_0 = (I_0^{\text{left}}, I_0^{\text{right}})$ from the wrist camera, the camera intrinsics K, and stereo baseline b. D is aligned to the left image I_0^{left}. We found that FoundationStereo produces cleaner depth maps than the ZED camera's proprietary stereo matching, particularly on transparent, specular, and textureless surfaces (Figure 5a).

Unprojecting depth to 3D. We unproject the depth map D into a 3D point cloud using the camera intrinsics K, then transform the points to the world frame by composing the camera-to-end-effector extrinsics $T_{\text{cam}}^{\text{ee}}$ with the forward kinematics (FK) at the capture joint configuration q_0:

$$\mathbf{p}^{\text{world}} = T_{\text{ee}}^{\text{world}}\, T_{\text{cam}}^{\text{ee}}\, \mathbf{p}^{\text{cam}} \quad \text{where } T_{\text{ee}}^{\text{world}} = \text{FK}(q_0).$$

This produces a dense point cloud of the scene in the world frame $\mathbf{p}^{\text{world}}$ (Figure 5b).

Grasp Generation. We use M2T2 [7] to predict ranked 6-DoF grasp poses from the full scene point cloud. Object-to-grasp association is performed in §IV-C using segmentation masks from the *Semantic Branch* (§IV-B). Because M2T2 reasons over the full scene, its predictions are informed by surrounding geometry, though they are not guaranteed to be collision-free.

In our illustrative example, M2T2 generates candidate grasps on the trays, one cracker package, and the soda can (Figure 5b). Note that some objects may not have predicted grasps; in such cases, we fall back to a heuristic 4-DoF grasp sampler in the *planning module* (§V). Having a large set of scene-level candidate grasps at this stage allows the planner to later select appropriate grasps based on task requirements and collision constraints.

B. Semantic Branch

Object Detection and Goal Grounding. We query Gemini Robotics-ER 1.5 [31], a VLM, once to jointly extract: (1) labels and 2D bounding boxes for objects in the scene, and (2) a symbolic goal $\mathcal{G}$ expressed as a conjunction of *predicates* (i.e., logical relations betweeen objects) over detected objects. The system was originally developed to support just the $\mathtt{On}(a, b)$ predicate, though we demonstrate defining additional predicates for new skills in §VIII. The VLM leverages its common-sense reasoning and cultural knowledge to ground references in the instruction to specific objects and assign task-relevant labels.

In our example with $\mathcal{L}$ = "serve peanut butter crackers on each tray", the VLM correctly identifies that "peanut butter crackers" refers to the two Lance cracker packages among other snacks (Goldfish crackers, Cheez-It crackers, nuts), and reasons that "each tray" requires placing one package on each to produce $\mathcal{G}$ (Figure 5c).

Object Segmentation. For each detected bounding box, we use SAM-2 [26] to generate a pixel-level segmentation mask from I_0^{left}. These masks are combined with the scene point cloud in §IV-C to extract per-object geometry and assign grasps to specific objects.

C. Combining Outputs

We combine scene-level geometry and candidate grasps from the *3D Vision Branch* with object identities and segmentation masks from the Semantic Branch into an object-centric 3D scene representation, producing per-object meshes with assigned grasps for the planning module.

Table Detection. We apply RANSAC [55] to the scene point cloud $\mathbf{p}^{\text{world}}$ to fit the dominant planar surface, which we assume to be the table. This assumption could be relaxed by detecting multiple support surfaces (e.g., tables, floors, cabinets) via semantic segmentation or multi-plane fitting.

Per-Object Mesh Reconstruction. We support two modes for reconstructing a watertight mesh for each object. The default *convex-hull* mode uses the segmentation mask of each detected object to extract the corresponding points from $\mathbf{p}^{\text{world}}$,

Fig. 6: **Shape-Completion Mode.** From the observed scene (left), RecGen [9] faithfully recovers the full geometry of each object (right), including the elephant watering can.

projects them downward along the z-axis to the object's lowest observed point, and computes the convex hull. We project to each object's own lowest point rather than to the table, as objects may rest on each other. The *shape-completion* mode instead feeds the RGB image I_0^{left}, predicted depth D, intrinsics K, and per-object segmentation masks from SAM-2 to RecGen [9], a foundation model that generates a full mesh and 6-DoF pose for each masked object (Figure 6).

Design Decision: Mesh Reconstruction. We use the convex hull as the default because it is extremely cheap to compute and typically over-approximates the object, providing conservative geometry for collision checking. The shape-completion mode provides higher-fidelity meshes for complex objects at substantially greater compute cost.

Grasp-to-Object Assignment. Each grasp predicted by M2T2 is assigned to the nearest object by querying its contact point against a KDTree [56] built from all object point clouds. Grasps whose nearest object point exceeds a distance threshold are discarded, as these typically arise from point cloud noise or partial observability.

V. Planning Module

TiPToP uses cuTAMP [1], a GPU-parallelized Task and Motion Planning algorithm, to search over discrete *plan skeletons* and optimize continuous parameters (grasp poses, placement poses, trajectories) to produce a full manipulation plan. cuTAMP operates primarily over pick-and-place primitives, though it can be extended to support additional primitives such as wiping (§VIII). We chose cuTAMP for its fast solution times on a single GPU and its ease of installation, and made several extensions to improve its real-world deployability (Appendix A).

Plan Skeleton Enumeration. Given the symbolic goal $\mathcal{G}$, cuTAMP uses a PDDL-style symbolic planner [57] to enumerate candidate plan skeletons — sequences of symbolic actions without committed continuous parameters. For example:

$$[\texttt{MoveFree}(q_0, ?q_1, ?\tau_1),\ \texttt{Pick}(\texttt{cracker}, ?g, p_0, ?q_1),$$
$$\texttt{MoveHolding}(\texttt{cracker}, ?g, ?q_1, ?q_2, ?\tau_2),$$
$$\texttt{Place}(\texttt{cracker}, ?g, ?p_1, \texttt{tray}, ?q_2)]$$

where $?g$, $?p_1$, $?q_i$, and $?\tau_i$ are unbound continuous parameters (grasp pose, placement pose, robot configurations, and trajectories, respectively).

The planner generates multiple skeletons that differ in action ordering and, crucially, may include auxiliary actions to move obstructing objects. In our example (Figure 4), shorter skeletons pick and place the two cracker packages directly onto the trays, while longer skeletons additionally move the soda can out of the way before grasping the obstructed crackers.

Particle Initialization. For each skeleton, cuTAMP initializes a large batch of candidate solutions, called *particles*, by sampling the continuous parameters left unbound by the skeleton: grasp poses (from M2T2 predictions or a heuristic top-down grasp sampler), placement poses on target surfaces, and robot configurations via inverse kinematics. For multi-step problems, these initial samples are generally infeasible as they may violate collision, stability, or kinematic constraints.

Particle Optimization. cuTAMP then ranks skeletons by a heuristic over the feasibility of their initialized particles. For each skeleton, cuTAMP performs differentiable optimization over all particles simultaneously, refining placement poses and robot configurations to jointly satisfy collision avoidance, stable placement, and kinematic feasibility constraints. The optimization terminates once sufficient particles satisfy all constraints, moving to the next skeleton otherwise. In our example, skeletons that attempt to pick the left crackers package directly fail optimization because the soda can obstructs all feasible grasps. cuTAMP finds satisfying particles on a longer skeleton that first moves the soda can elsewhere on the table.

Motion Planning. For each satisfying particle, cuTAMP invokes cuRobo [58], a GPU-accelerated motion planner, to solve for the remaining trajectory parameters ($?\tau_i$) as collision-free, time-parameterized trajectories. The final output is a manipulation plan $\{(q_t, \dot{q}_t, g_t)\}_{t=0}^{T}$: joint positions, joint velocities, and gripper commands.

Design Decision: cuTAMP Extensions. Deploying cuTAMP in the real world required several modifications to make it more reliable on imperfect scene reconstructions from the perception module and to extend its capabilities. Because convex-hull reconstruction over-approximates geometry, objects can appear in collision in the initial state. We exclude these false collisions from the collision cost functions so they do not block planning, and additionally increase the number of motion planning attempts and iteratively relax collision checking thresholds. To support arbitrarily oriented surfaces, we extend cuTAMP's placement-surface cost functions to oriented bounding boxes. These and other extensions (Appendix A) are a substantial part of our engineering contribution.

VI. Execution Module

The execution module tracks a planned trajectory $\{(q_t, \dot{q}_t, g_t)\}_{t=0}^{T}$ on the robot. Accurately tracking trajectories is crucial, since the planner assumes consistency between the robot's joint-space execution and the resulting scene configuration. Even sub-centimeter tracking errors can cause grasps or placements to fail.

Design Decision: Custom Joint Impedance Controller. Existing open-source controllers, including DROID's default Polymetis controller, could not track our timed trajectories precisely enough. We implemented our own joint-space impedance controller with tuned gains that keep the resulting end-effector tracking error within 5 mm at high speeds, which is tight enough for our evaluation tasks. The full control law is in Appendix B.

Open-loop execution. Our current system executes plans open-loop, without replanning from execution-time observations. We keep this first version deliberately simple to see how far a plan-once approach can go. This suffices when the scene is static and trajectories track accurately, but fails when objects move or grasps slip. Closing the loop is a natural and straightforward extension of TiPToP, and among our most impactful planned improvements (§IX).

VII. Experiments

Our experiments are designed to answer the following questions:

- **Q1.** How well does TiPToP perform across diverse open-ended tabletop manipulation tasks, with no embodiment-specific training data?
- **Q2.** How efficient is TiPToP in terms of total time (planning plus execution)?
- **Q3.** What are the primary failure modes of TiPToP, and how are they split across the system's modules?

We evaluate TiPToP in two settings: a controlled comparison against $\pi_{0.5}$-DROID [11], a state-of-the-art VLA fine-tuned on 350 hours of DROID demonstrations, and on MolmoSpaces, a large-scale independent benchmark. We present each study in turn and analyze the results against Q1–Q3 in §VII-D.

A. Controlled Comparison with $\pi_{0.5}$-DROID

We evaluated both systems on 28 tabletop scenes across three settings: an IsaacSim simulation [20] (5 tasks), the DROID setup used by TiPToP's developers (8 tasks), and a separate DROID setup operated by an external evaluation team (15 tasks). The scenes span four categories of increasing difficulty: *simple* single-step pick-and-place with no distractors; *distractor* tasks that require manipulating only the relevant object amid clutter; *semantic* tasks with referring expressions that demand common sense or physical reasoning about the scene (e.g., "pick up the largest toy"); and *multi-step* tasks that require sequencing several actions with physical reasoning (e.g., constrained packing, or moving an obstacle out of the way). Following an "in-the-wild" protocol [59], both systems received the same instruction and starting configuration. We report binary success rate (SR) and a finer-grained task progress (TP) defined via per-task subgoals (see Appendix C).

Table I reports per-scene results. Over 165 trials, TiPToP achieves a higher overall success rate (98/165 vs. 55/165) and task progress (74.6% vs. 52.4%) than $\pi_{0.5}$-DROID. On simple pick-and-place the two are comparable, with $\pi_{0.5}$-DROID slightly ahead on success rate (27/40 vs. 22/40) and TiPToP ahead on task progress. The gap widens with task complexity: TiPToP leads on distractor, semantic, and multi-step scenes. TiPToP is also faster in total completion time (Table II), beating $\pi_{0.5}$-DROID on five of six scenes and matching it on the sixth, often completing single-step tasks in roughly half the time.

Even when TiPToP does not fully succeed, it often completes most subgoals (e.g., 72.5% task progress at 5/10 on cube→bowl), indicating that failures tend to be isolated to a single step. $\pi_{0.5}$-DROID, by contrast, fails completely (0/5) on four of the semantic scenes. However, $\pi_{0.5}$-DROID does better in cases where TiPToP's convex-hull meshes misrepresent concave objects (both banana scenes), and where a slipped grasp must be retried, which TiPToP's open-loop execution cannot do (e.g., red cubes→plate: 1/5 at 50% task progress).

	TiPToP		$\pi_{0.5}$-DROID	
Scene	SR	TP	SR	TP
Simple				
Cube → bowl (sim)	5/10	72.5%	**8/10**	**90%**
Can → mug (sim)	**9/10**	**97.5%**	2/10	50%
Banana → bin (sim)	0/10	70%	**9/10**	**97.5%**
Marker → tray	3/5	80%	**5/5**	**100%**
Crackers → tray†	**5/5**	**100%**	3/5	60%
	22/40	**84.0%**	**27/40**	79.5%
Distractor				
Meat can → sugar box (sim)	**5/10**	**72.5%**	0/10	5%
Coffee capsules → plate	**4/5**	**90%**	2/5	58%
Turkish figs → plate	**3/5**	**64%**	2/5	52%
Cashews → plate	0/5	**16%**	0/5	12%
Red cubes → plate	1/5	50%	**5/5**	**92%**
Fish → box	**4/5**	**80%**	0/5	10%
Crackers → tray (medium)†	**5/5**	**100%**	3/5	80%
PB crackers → tray (hard)†	**5/5**	**100%**	0/5	20%
	27/45	**71.6%**	12/45	41.1%
Semantic				
Toy → matching plate	**4/5**	**90%**	1/5	62%
Creeper → plate	**3/5**	**70%**	0/5	0%
Largest toy → plate	**3/5**	**70%**	0/5	20%
Red A → color pile	**5/5**	**100%**	3/5	80%
Banana → box	**2/5**	**40%**	0/5	30%
N block → indicated cup	**3/5**	**80%**	2/5	60%
Sort blocks by color	**5/5**	**100%**	0/5	32%
Banana → matching plate	1/5	20%	**4/5**	**90%**
	26/40	**71.3%**	10/40	46.8%
Multi-step				
Color cubes → bowl (sim)	**9/10**	**94.6%**	0/10	24.2%
AirPods → cup	1/5	55%	**3/5**	**75%**
Pack pods → tray†	**4/5**	**80%**	1/5	65.7%
Pack pods → tray (obs.)†	**1/5**	**67%**	0/5	64%
Aleve bottle → tray (obs.)†	**4/5**	**80%**	2/5	70%
Three marbles → cup†	**2/5**	**80%**	0/5	6.7%
Marbles + cable†	**2/5**	**70%**	0/5	60%
	23/40	**75.2%**	6/40	52.2%
Overall	**98/165**	**74.6%**	55/165	52.4%

TABLE I: **Per-scene performance comparison over 28 scenes.** SR = Success Rate, TP = Task Progress. Best results are **bolded**. Per category and overall, SR is summed and TP averaged. †Evaluated by system designers; unmarked scenes evaluated by the external evaluation team.

B. MolmoSpaces

MolmoSpaces [19] is a large-scale benchmark of diverse household tasks. We run its 9 pick and pick-and-place tasks on the DROID setup (1,000 episodes per task) and omit the open and close tasks, which TiPToP does not currently support. The leaderboard separates *in-distribution (ID)* methods, trained on MolmoSpaces or MolmoBot simulator data, from *non-ID* methods. TiPToP is the only non-ID method that uses zero robot or simulator demonstrations.

Scene	$\pi_{0.5}$-DROID Time (s)	TiPToP Time (s)	TiPToP Plan (s)
Simulation			
Cube → bowl	27.4	**17.9**	9.7
Can → mug	41.0	**18.6**	9.2
Real-World			
Crackers → tray (simple)	32.2	**14.9**	7.0
Crackers → tray (medium)	45.2	**14.9**	7.3
Pack pods → tray	53.4	**47.0**	20.5
Aleve bottle → tray (obs.)†	31.2	31.2	16.4

TABLE II: **Completion time comparison**. 'Time' reports average time-to-success over successful trials only. 'Plan' reports the time required for TiPToP to run the perception and planning module (included in 'Time').

Table III reports the oracle[2] success rate against all leaderboard entrants at the time of writing. TiPToP is the strongest non-ID method overall (46.1% SR vs. 41.3% for the next-best, WALL-OSS-0.5 [14]), ranking first among non-ID methods on 6 of 9 tasks. These six are the tasks with complex referring instructions and constraints that end-to-end VLAs and WAMs struggle with. On Pick & Place-NextTo, TiPToP tops the entire leaderboard, the only task where a non-ID method beats every ID-trained model.

C. Failure Analysis

A key advantage of TiPToP's modular architecture is that we can trace each failure to the responsible module, something difficult to do for an end-to-end policy. Separately from the controlled comparison, we ran 173 rearrangement trials on our own DROID setup and traced the root cause of every failure to a specific module (Figure 7).

Grasping failures (31/55) are the most common mode, occurring when M2T2 produces high-scoring grasps that fail in execution or when the heuristic fallback sampler is used for objects without M2T2 predictions. **Scene-completion errors** (13/55) arise from convex-hull mesh approximations that over-approximate concave objects (e.g., a banana) or under-approximate under partial observability, causing collisions during execution. **VLM errors** (6/55) occur when Gemini misdetects objects or produces incorrect bounding boxes, and **cuTAMP failures** (5/55) occur when the planner cannot find a feasible plan within its time budget.

D. Discussion

Q1: TiPToP is competitive across diverse tasks with no robot training data. On the MolmoSpaces benchmark, TiPToP is the strongest non-ID entrant, and in the controlled comparison it substantially outperforms $\pi_{0.5}$-DROID as task complexity grows. These gaps are largest exactly where TiPToP's structure helps: distractor and semantic tasks benefit from explicit VLM goal grounding, which isolates the task-relevant objects amid clutter and resolves referring expressions ("largest toy," "matching plate," "sort by color") that the baseline VLAs and WAMs have no mechanism for. Multi-step tasks benefit from cuTAMP's decomposition into feasible, collision-free action sequences. We further show this generality extends to new robots and new skills in §VIII.

[2]Oracle success rate is defined as "success anywhere in policy execution for a time horizon, not only at the end of the episode".

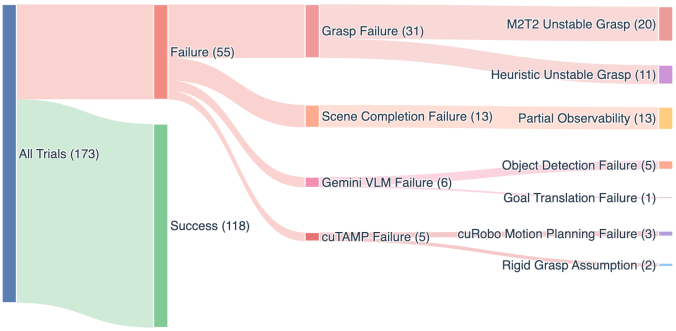


Fig. 7: **Failure Analysis.** Sankey diagram showing outcomes of 173 trials. The most common failure modes are grasping failures (missed or unstable grasps), followed by scene completion errors, VLM detection errors, then cuTAMP failures.

Q2: TiPToP is efficient. Although TiPToP spends time up front building the scene representation and planning, it is faster on nearly every timed scene in Table II: it commits to a single time-optimal trajectory and executes it directly, whereas $\pi_{0.5}$-DROID reacts step by step, often hovering and re-attempting grasps instead of making progress. The time advantage narrows on multi-step tasks, where execution rather than planning dominates total time, although TiPToP achieves a higher success rate.

Q3: failures concentrate in a few modules, each addressable independently. Our failure analysis points to two main culprits: grasping failures (over half of all failures) and convex-hull mesh approximation. Grasping failures persist because open-loop execution cannot retry a slipped or missed grasp, while the convex-hull approximation poorly represents concave shapes and is coarse for collision checking. Because TiPToP is modular, each maps to a specific component, and we suggest directions for addressing them in §IX.

VIII. Extending to New Embodiments and Skills

TiPToP is embodiment-agnostic: perception runs on RGB pixels and the planner is parameterized by a robot model. We deployed it on a UR5e arm with a wrist-mounted RealSense D435 camera (Figure 3c). Adapting to the new embodiment required providing the robot URDF, generating collision spheres, writing a cuRobo configuration file, and implementing camera and controller interfaces for the new hardware. The full adaptation was completed within a few hours (Appendix D). We also deployed it on a Trossen WidowX AI arm with a wrist-mounted RealSense D405 camera (Figure 3d).

New skills beyond pick-and-place. TiPToP's modularity makes it straightforward to extend beyond pick-and-place. We added a whiteboard-wiping primitive that erases writing from a surface with an eraser (Figure 8) through three localized changes, none of which touched the perception or execution infrastructure: two new predicates (`IsEraser`, `IsCleaned`) and a goal-grounding prompt extension in the semantic branch; a `Wipe` cuTAMP operator that the task planner automatically

Method	MolmoSpaces		MolmoBot Pick variants				MolmoBot Pick & Place variants			Overall
	Pick	PnP	MSProc	Classic	Filam.	RandCam	PnP	PnP-NextTo	PnP-Color	
Non-In-Distribution Methods										
TiPToP (ours)	68.7	33.2	67.5	**50.0**	**48.5**	**47.8**	**29.4**	**38.0***	**31.5**	**46.1**
WALL-OSS-0.5 (VLA) [14]	**73.8**	**46.0**	**78.4**	47.9	47.2	30.1	13.8	15.9	18.3	41.3
Cosmos3-Nano (WAM) [17]	67.5	36.9	67.7	32.8	30.7	31.2	25.7	18.4	28.6	37.7
Psi-R2 (WAM) [16]	73.1	41.5	50.2	23.4	25.8	27.2	2.6	2.9	1.5	27.6
MolmoAct2-DROID (VLA) [13]	48.3	23.3	43.4	20.5	21.9	11.1	15.6	18.5	19.3	24.7
$\pi_{0.5}$-DROID (VLA) [11]	36.4	13.6	18.1	6.4	7.0	8.0	12.0	10.2	10.4	13.6
In-Distribution Methods: fine-tuned on MolmoSpaces and MolmoBot Simulator data										
MolmoBot Best (VLA) [34]	92.8	–	93.5	66.8	64.0	63.7	66.5	28.7	67.8	60.1

TABLE III: **MolmoSpaces benchmark results** (oracle success rate %). Best non-ID method per task in **bold**. * ranks first on the full leaderboard. Integration details in Appendix F.

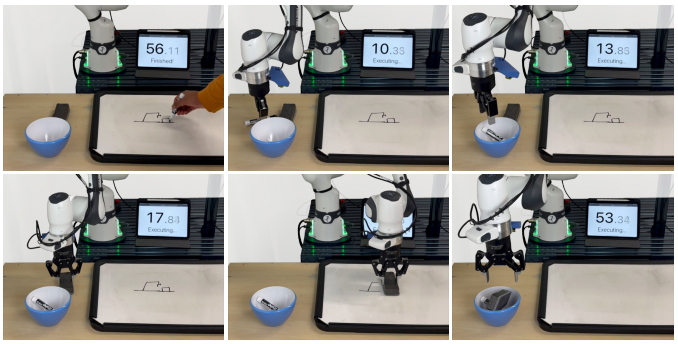


Fig. 8: **Wiping.** We demonstrate that TiPToP can be straightforwardly extended to perform wiping in addition to pick-and-place. Task instruction: "erase the whiteboard and put everything into the bowl".

sequences after a pick; and a wiping controller that localizes the region to clean and executes back-and-forth strokes over it. The entire extension took under a day (Appendix E).

IX. Limitations and Future Directions

TiPToP's modular architecture means each of its current limitations maps to a specific component that can be improved independently. Below, we discuss these limitations alongside concrete directions for addressing them.

Open-loop execution. This is the single most impactful limitation: our failure analysis (Figure 7) shows that grasping failures account for over half of all failures, many of which could be recovered from by re-attempting the grasp. The most direct improvement is to re-run perception and planning after each pick-and-place step, enabling recovery from failed grasps or unexpected object movement [40, 60, 61, 62].

Single-viewpoint perception. All task-relevant objects must be at least partially visible from a single wrist-camera pose. This also limits mesh quality: with only one viewpoint, both convex hull and learned shape completion can over or under-approximate object geometry, leading to unnecessary collisions or missed collisions during execution. Multi-view perception, via active camera movement before planning or additional static cameras, would reduce occlusions and improve shape estimates. Advances in depth estimation [63, 64] could further improve point cloud quality, and better grasp prediction models would address TiPToP's most common failure mode.

Integrating learned policies. Our experiments show that TiPToP and $\pi_{0.5}$-DROID exhibit complementary failure modes: TiPToP excels at geometric reasoning, long-horizon sequencing, and semantic grounding via its VLM goal-grounding step, but fails when grasps slip or meshes are poorly approximated; $\pi_{0.5}$-DROID benefits from closed-loop reactivity but struggles with multi-step structure, tight constraints, and distractor-rich scenes. We view this complementarity as constructive: end-to-end policies are a natural reactive layer for TiPToP's skills, both improving robustness for existing skills (e.g., re-grasping a slipped object) and enabling new ones that are difficult to hand-engineer (e.g., folding, cable manipulation, contact-rich tasks). Integrating such skills requires specifying their abstract preconditions and effects, which could be engineered or learned from data [65, 66, 67, 68], so the planner can reason about when to invoke them.

X. Conclusion

We presented TiPToP, a modular planning-based manipulation system that composes pretrained vision foundation models with GPU-accelerated TAMP to solve multi-step manipulation tasks from RGB images and natural language, without any robot training data. Over 165 trials in 28 evaluation scenes in simulation and on real hardware, TiPToP matches or outperforms $\pi_{0.5}$-DROID, particularly on tasks requiring semantic grounding, distractor rejection, and multi-step sequencing. On the MolmoSpaces benchmark, TiPToP ranks first overall on pick and pick-and-place tasks among methods not trained on in-distribution data. Our system's modular architecture enables component-level failure analysis: we traced failures over 173 trials to specific modules, identifying grasping as the dominant bottleneck.

A central finding of this work is that a modular system built from off-the-shelf foundation models and planning algorithms can serve as a strong manipulation system. Each component can be independently upgraded as better depth estimators, grasp predictors, VLMs, and TAMP or motion planners become available. The complementary failure profiles of TiPToP and end-to-end policies suggest that integrating learned reactive skills within TiPToP's framework could yield systems that combine the structured reasoning of planning with the robustness of closed-loop visuomotor control. We hope our open-source system supports further research toward broadly competent and generalizable manipulation systems.

References

[1] W. Shen, C. Garrett, N. Kumar, A. Goyal, T. Hermans, L. P. Kaelbling, T. Lozano-Pérez, and F. Ramos, "Differentiable gpu-parallelized task and motion planning," in *Robotics: Science and Systems (RSS)*, 2025. [Online]. Available: https://arxiv.org/abs/2411.11833

[2] Gemini Team, R. Anil, S. Borgeaud, J.-B. Alayrac, J. Yu, R. Soricut, J. Schalkwyk, A. M. Dai, A. Hauth, K. Millican *et al.*, "Gemini: A family of highly capable multimodal models," *arXiv preprint arXiv:2312.11805*, 2023. [Online]. Available: https://arxiv.org/abs/2312.11805

[3] M. Deitke, C. Clark, S. Lee, R. Tripathi, Y. Yang, J. S. Park, M. Salehi, N. Muennighoff, K. Lo, L. Soldaini *et al.*, "Molmo and pixmo: Open weights and open data for state-of-the-art vision-language models," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025. [Online]. Available: https://arxiv.org/abs/2409.17146

[4] S. Bai, Y. Cai, R. Chen, K. Chen, X. Chen, Z. Cheng, L. Deng, W. Ding, C. Gao, C. Ge *et al.*, "Qwen3-vl technical report," *arXiv preprint arXiv:2511.21631*, 2025. [Online]. Available: https://arxiv.org/abs/2511.21631

[5] OpenAI, "Gpt-4o system card," *arXiv preprint arXiv:2410.21276*, 2024. [Online]. Available: https://arxiv.org/abs/2410.21276

[6] B. Wen, M. Trepte, J. Aribido, J. Kautz, O. Gallo, and S. Birchfield, "Foundationstereo: Zero-shot stereo matching," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025. [Online]. Available: https://arxiv.org/abs/2501.09898

[7] W. Yuan, A. Murali, A. Mousavian, and D. Fox, "M2t2: Multi-task masked transformer for object-centric pick and place," in *Conference on Robot Learning (CoRL)*, 2023. [Online]. Available: https://proceedings.mlr.press/v229/yuan23a.html

[8] SAM 3D Team, X. Chen, F.-J. Chu, P. Gleize, K. J. Liang, A. Sax, H. Tang, W. Wang, M. Guo, T. Hardin, X. Li, A. Lin, J. Liu, Z. Ma, A. Sagar, B. Song, X. Wang, J. Yang, B. Zhang, P. Dollár, G. Gkioxari, M. Feiszli, and J. Malik, "SAM 3D: 3dfy anything in images," *arXiv preprint arXiv:2511.16624*, 2025. [Online]. Available: https://arxiv.org/abs/2511.16624

[9] A. Zadaianchuk, L. Barcellona, L. Schuenemann, C. Gumbsch, Z. Wang, M. Z. Irshad, F. Despinoy, R. Aljundi, S. Gavves, and S. Zakharov, "Recgen: Reconstructive generation of 3d scenes from rgb-d observations," *arXiv preprint arXiv:2604.27106*, 2026. [Online]. Available: https://arxiv.org/abs/2604.27106

[10] C. R. Garrett, R. Chitnis, R. Holladay, B. Kim, T. Silver, L. P. Kaelbling, and T. Lozano-Pérez, "Integrated task and motion planning," *Annual Review of Control, Robotics, and Autonomous Systems*, 2021. [Online]. Available: https://www.annualreviews.org/doi/full/10.1146/annurev-control-091420-084139

[11] Physical Intelligence, K. Black, N. Brown, J. Darpinian, K. Dhabalia, D. Driess, A. Esmail, M. Equi, C. Finn, N. Fusai, M. Y. Galliker, D. Ghosh, L. Groom, K. Hausman, B. Ichter, S. Jakubczak, T. Jones, L. Ke, D. LeBlanc, S. Levine, A. Li-Bell, M. Mothukuri, S. Nair, K. Pertsch, A. Z. Ren, L. X. Shi, L. Smith, J. T. Springenberg, K. Stachowicz, J. Tanner, Q. Vuong, H. Walke, A. Walling, H. Wang, L. Yu, and U. Zhilinsky, "$\pi_{0.5}$: a vision-language-action model with open-world generalization," *arXiv preprint arXiv:2504.16054*, 2025. [Online]. Available: https://arxiv.org/abs/2504.16054

[12] LAP Team, "LAP: Language-action pre-training enables zero-shot cross-embodiment transfer," *arXiv preprint arXiv:2602.10556*, 2026. [Online]. Available: https://arxiv.org/abs/2602.10556

[13] H. Fang, J. Duan, D. Clay, S. Wang, S. Liu, W. Huang, X. Fan, W.-C. Tsai, S. Chen, Y. R. Wang, S. Xing, J. Cho, J. S. Park, A. Eftekhar, P. Sushko, K. Farley, A. Wadhwa, C. Harrison, W. Han, Y.-C. Lee, E. VanderBilt, R. Hendrix, S. Ellawela, L. Ngoo, J. Chai, Z. Ren, A. Farhadi, D. Fox, and R. Krishna, "MolmoAct2: Action reasoning models for real-world deployment," 2026. [Online]. Available: https://arxiv.org/abs/2605.02881

[14] R. Yu, P. Zhang, S. Liu, B. Liu, M. Kang, S. Li, L. Shi, E. Ma, P. Yang, C. Pan, J. Chen, D. Liu, R. Sun, M. Guo, B. Zhang, H. Zhou, Z. Xu, V. Chen, H. Huang, J. Wang, D. Kuzi, A. Zhai, H. Su, R. Gan, L. Liang, H. Wang, and Q. Wang, "Wall-oss-0.5 technical report," *arXiv preprint arXiv:2605.30877*, 2026. [Online]. Available: https://arxiv.org/abs/2605.30877

[15] S. Ye, Y. Ge *et al.*, "World action models are zero-shot policies," *arXiv preprint arXiv:2602.15922*, 2026. [Online]. Available: https://arxiv.org/abs/2602.15922

[16] World Labs, "PSI-R2: A predictive spatial intelligence robot policy," MolmoSpaces-Bench leaderboard entry, 2026. [Online]. Available: https://molmospaces.allen.ai/leaderboard

[17] N. Agarwal, A. Ali, J. Allen, M. Antolini, A. Aubame, A. Azzolini, J. Bai, M. Bala, Y. Balaji, J. Bapst *et al.*, "Cosmos 3: Omnimodal world models for physical ai," *arXiv preprint arXiv:2606.02800*, 2026. [Online]. Available: https://arxiv.org/abs/2606.02800

[18] A. Khazatsky, K. Pertsch, S. Nair, A. Balakrishna, S. Dasari, S. Karamcheti, S. Nasiriany, M. K. Srirama, L. Luo, K. Vuong *et al.*, "Droid: A large-scale in-the-wild robot manipulation dataset," in *Robotics: Science and Systems (RSS)*, 2024. [Online]. Available: https://arxiv.org/abs/2403.12945

[19] Allen Institute for AI, "Molmospaces: A large-scale open ecosystem for robot navigation and manipulation," *arXiv preprint arXiv:2602.11337*, 2026. [Online]. Available: https://arxiv.org/abs/2602.11337

[20] NVIDIA, "Isaac Sim," https://developer.nvidia.com/isaac/sim, 2024.

[21] F. Tosi, L. Bartolomei, and M. Poggi, "A survey on deep stereo matching in the twenties," *International Journal of Computer Vision (IJCV)*, 2025. [Online]. Available: https://link.springer.com/article/10.1007/s11263-024-02331-0

[22] X. Guo, C. Zhang, Y. Zhang, R. Wang, D. Nie, W. Zheng, M. Poggi, H. Zhao, M. Ye, Q. Zou, and L. Chen, "Stereo anything: Unifying zero-shot stereo matching with large-scale mixed data," *arXiv preprint arXiv:2411.14053*, 2024. [Online]. Available: https://arxiv.org/abs/2411.14053

[23] A. Murali, B. Sundaralingam, Y.-W. Chao, W. Yuan, J. Yamada, M. Carlson, F. Ramos, S. Birchfield, D. Fox, and C. Eppner, "Graspgen: A diffusion-based framework for 6-dof grasping with on-generator training," *arXiv preprint arXiv:2507.13097*, 2025. [Online]. Available: https://arxiv.org/abs/2507.13097

[24] M. Sundermeyer, A. Mousavian, R. Triebel, and D. Fox, "Contact-graspnet: Efficient 6-dof grasp generation in cluttered scenes," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2021. [Online]. Available: https://arxiv.org/abs/2103.14127

[25] A. Kirillov, E. Mintun, N. Ravi, H. Mao, C. Rolland, L. Gustafson, T. Xiao, S. Whitehead, A. C. Berg, W.-Y. Lo, P. Dollár, and R. Girshick, "Segment anything," in *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023. [Online]. Available: https://arxiv.org/abs/2304.02643

[26] N. Ravi, V. Gabeur, Y.-T. Hu, R. Hu, C. Ryali, T. Ma, H. Khedr, R. Rädle, C. Rolland, L. Gustafson, E. Mintun, J. Pan, K. V. Alwala, N. Carion, C.-Y. Wu, R. Girshick, P. Dollár, and C. Feichtenhofer, "Sam 2: Segment anything in images and videos," *arXiv preprint arXiv:2408.00714*, 2024. [Online]. Available: https://arxiv.org/abs/2408.00714

[27] A. Agarwal, G. Singh, B. Sen, T. Lozano-Pérez, and L. P. Kaelbling, "Scenecomplete: Open-world 3d scene completion in cluttered real world environments for robot manipulation," *IEEE Robotics and Automation Letters (RA-L)*, 2025. [Online]. Available: https://arxiv.org/abs/2410.23643

[28] A. Brohan, N. Brown, J. Carbajal, Y. Chebotar, X. Chen, K. Choromanski, T. Ding, D. Driess, A. Dubey, C. Finn *et al.*, "Rt-2: Vision-language-action models transfer web knowledge to robotic control," in *Conference on Robot Learning (CoRL)*, 2023. [Online]. Available: https://proceedings.mlr.press/v229/zitkovich23a.html

[29] M. J. Kim, K. Pertsch, S. Karamcheti, T. Xiao, A. Balakrishna, S. Nair, R. Rafailov, E. Foster, G. Lam, P. Sanketi *et al.*, "Openvla: An open-source vision-language-action model," *arXiv preprint arXiv:2406.09246*, 2024. [Online]. Available: https://arxiv.org/abs/2406.09246

[30] J. Bjorck, F. Castañeda, N. Cherniadev, X. Da, R. Ding, L. Fan, Y. Fang, D. Fox, F. Hu, S. Huang, J. Jang, Z. Jiang, J. Kautz, K. Kundalia, L. Lao, Z. Li, Z. Lin, K. Lin, G. Liu, E. Llontop, L. Magne, A. Mandlekar, A. Narayan, S. Nasiriany, S. Reed, Y. L. Tan, G. Wang, Z. Wang, J. Wang, Q. Wang, J. Xiang, Y. Xie, Y. Xu, Z. Xu, S. Ye, Z. Yu, A. Zhang, H. Zhang, Y. Zhao, R. Zheng, and Y. Zhu, "Gr00t n1: An open foundation model for generalist humanoid robots," *arXiv preprint arXiv:2503.14734*, 2025. [Online]. Available: https://arxiv.org/abs/2503.14734

[31] Gemini Robotics Team, S. Abeyruwan, J. Ainslie, J.-B. Alayrac, M. G. Arenas, T. Armstrong, A. Balakrishna, R. Baruch, M. Bauza, M. Blokzijl *et al.*, "Gemini robotics: Bringing ai into the physical world," *arXiv preprint arXiv:2503.20020*, 2025. [Online]. Available: https://arxiv.org/abs/2503.20020

[32] Open X-Embodiment Collaboration, "Open x-embodiment: Robotic learning datasets and rt-x models," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2024. [Online]. Available: https://arxiv.org/abs/2310.08864

[33] K. Black, N. Brown, D. Driess, A. Esmail, M. Equi, C. Finn, N. Fusai, L. Groom, K. Hausman, B. Ichter *et al.*, "π_0: A vision-language-action flow model for general robot control," in *Robotics: Science and Systems (RSS)*, 2025. [Online]. Available: https://arxiv.org/abs/2410.24164

[34] Allen Institute for AI, "Molmobot: Large-scale simulation enables zero-shot manipulation," *arXiv preprint arXiv:2603.16861*, 2026. [Online]. Available: https://arxiv.org/abs/2603.16861

[35] L. P. Kaelbling and T. Lozano-Pérez, "Hierarchical task and motion planning in the now," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2011.

[36] C. R. Garrett, T. Lozano-Pérez, and L. P. Kaelbling, "FFRob: Leveraging symbolic planning for efficient task and motion planning," *International Journal of Robotics Research (IJRR)*, 2018. [Online]. Available: https://journals.sagepub.com/doi/abs/10.1177/0278364917739114

[37] ——, "PDDLStream: Integrating symbolic planners and blackbox samplers via optimistic adaptive planning," in *International Conference on Automated Planning and Scheduling (ICAPS)*, 2020. [Online]. Available: https://ojs.aaai.org/index.php/ICAPS/article/view/6739

[38] M. Toussaint, "Logic-geometric programming: An optimization-based approach to combined task and motion planning," in *International Joint Conference on Artificial Intelligence (IJCAI)*, 2015. [Online]. Available: https://www.ijcai.org/Proceedings/15/Papers/274.pdf

[39] M. Toussaint, K. Allen, K. Smith, and J. Tenenbaum, "Differentiable physics and stable modes for tool-use and manipulation planning," in *Robotics: Science and Systems (RSS)*, 2018. [Online]. Available: https://www.roboticsproceedings.org/rss14/p44.html

[40] A. Curtis, X. Fang, L. P. Kaelbling, T. Lozano-Pérez, and C. R. Garrett, "Long-horizon manipulation of unknown objects via task and motion planning with estimated affordances," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2022. [Online]. Available: https://arxiv.org/abs/2108.04145

[41] S. Srivastava, E. Fang, L. Riano, R. Chitnis, S. Russell, and P. Abbeel, "Combined task and motion planning through an extensible planner-independent interface layer," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2014. [Online]. Available: https://people.eecs.berkeley.

edu/~russell/papers/icra14-planrob.pdf

[42] R. Chitnis, T. Silver, J. B. Tenenbaum, T. Lozano-Pérez, and L. P. Kaelbling, "Learning neuro-symbolic relational transition models for bilevel planning," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2022. [Online]. Available: https://arxiv.org/abs/2105.14074

[43] R. E. Fikes and N. J. Nilsson, "Strips: A new approach to the application of theorem proving to problem solving," *Artificial Intelligence*, 1971. [Online]. Available: https://www.sciencedirect.com/science/article/abs/pii/0004370271900105

[44] N. J. Nilsson, "Shakey the robot," SRI International, Artificial Intelligence Center, Tech. Rep., 1984. [Online]. Available: https://ai.stanford.edu/~nilsson/OnlinePubs-Nils/shakey-the-robot.pdf

[45] M. Ahn, A. Brohan, N. Brown, Y. Chebotar, O. Cortes, B. David, C. Finn, C. Fu, K. Gopalakrishnan, K. Hausman *et al.*, "Do as i can, not as i say: Grounding language in robotic affordances," in *Conference on Robot Learning (CoRL)*, 2022. [Online]. Available: https://arxiv.org/abs/2204.01691

[46] W. Huang, F. Xia, T. Xiao, H. Chan, J. Liang, P. Florence, A. Zeng, J. Tompson, I. Mordatch, Y. Chebotar, P. Sermanet, N. Brown, T. Jackson, L. Luu, S. Levine, K. Hausman, and B. Ichter, "Inner monologue: Embodied reasoning through planning with language models," in *Conference on Robot Learning (CoRL)*, 2022. [Online]. Available: https://arxiv.org/abs/2207.05608

[47] J. Liang, W. Huang, F. Xia, P. Xu, K. Hausman, B. Ichter, P. Florence, and A. Zeng, "Code as policies: Language model programs for embodied control," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2023. [Online]. Available: https://arxiv.org/abs/2209.07753

[48] I. Singh, V. Blukis, A. Mousavian, A. Goyal, D. Xu, J. Tremblay, D. Fox, J. Thomason, and A. Garg, "Progprompt: Generating situated robot task plans using large language models," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2023. [Online]. Available: https://arxiv.org/abs/2209.11302

[49] W. Huang, C. Wang, R. Zhang, Y. Li, J. Wu, and L. Fei-Fei, "Voxposer: Composable 3d value maps for robotic manipulation with language models," in *Conference on Robot Learning (CoRL)*, 2023. [Online]. Available: https://proceedings.mlr.press/v229/huang23b.html

[50] P. Liu, Y. Orru, J. Vakil, C. Paxton, N. M. M. Shafiullah, and L. Pinto, "Demonstrating ok-robot: What really matters in integrating open-knowledge models for robotics," in *Robotics: Science and Systems (RSS)*, 2024. [Online]. Available: https://arxiv.org/abs/2401.12202

[51] S. Wang, M. Han, Z. Jiao, Z. Zhang, Y. N. Wu, S.-C. Zhu, and H. Liu, "Llm3: Large language model-based task and motion planning with motion failure reasoning," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2024. [Online]. Available: https://arxiv.org/abs/2403.11552

[52] A. Curtis, N. Kumar, J. Cao, T. Lozano-Pérez, and L. P. Kaelbling, "Trust the proc3s: Solving long-horizon robotics problems with llms and constraint satisfaction," in *Conference on Robot Learning (CoRL)*, 2024. [Online]. Available: https://arxiv.org/abs/2406.05572

[53] Z. Yang, C. R. Garrett, D. Fox, T. Lozano-Pérez, and L. P. Kaelbling, "Guiding long-horizon task and motion planning with vision language models," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2025. [Online]. Available: https://arxiv.org/abs/2410.02193

[54] N. Kumar, W. Shen, F. Ramos, D. Fox, T. Lozano-Pérez, L. P. Kaelbling, and C. R. Garrett, "Open-world task and motion planning via vision-language model inferred constraints," *arXiv preprint arXiv:2411.08253*, 2024. [Online]. Available: https://arxiv.org/abs/2411.08253

[55] M. A. Fischler and R. C. Bolles, "Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM (CACM)*, 1981. [Online]. Available: https://dl.acm.org/doi/10.1145/358669.358692

[56] J. L. Bentley, "Multidimensional binary search trees used for associative searching," *Communications of the ACM (CACM)*, 1975. [Online]. Available: https://dl.acm.org/doi/10.1145/361002.361007

[57] D. McDermott, M. Ghallab, A. E. Howe, C. A. Knoblock, A. Ram, M. M. Veloso, D. S. Weld, and D. E. Wilkins, "PDDL: The planning domain definition language," 1998. [Online]. Available: https://www.semanticscholar.org/paper/PDDL-the-planning-domain-definition-language-McDermott-Ghallab/d82c6b8081343b2eae63d45feefe630233ad60e1

[58] B. Sundaralingam, S. K. S. Hari, A. Fishman, C. R. Garrett, K. Van Wyk, V. Blukis, A. Millane, H. Oleynikova, A. Handa, F. Ramos, N. D. Ratliff, and D. Fox, "curobo: Parallelized collision-free robot motion generation," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2023. [Online]. Available: https://ieeexplore.ieee.org/document/10160765/

[59] J. Wang, M. Leonard, K. Daniilidis, D. Jayaraman, and E. S. Hu, "Evaluating π_0 in the wild: Strengths, problems, and the future of generalist robot policies," 2025. [Online]. Available: https://penn-pal-lab.github.io/Pi0-Experiment-in-the-Wild/

[60] M. Levihn, L. P. Kaelbling, T. Lozano-Pérez, and M. Stilman, "Foresight and reconsideration in hierarchical planning and execution," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2013. [Online]. Available: https://dspace.mit.edu/handle/1721.1/90271

[61] O. Pettersson, "Execution monitoring in robotics: A survey," *Robotics and Autonomous Systems (RAS)*, 2005. [Online]. Available: https://www.sciencedirect.com/science/article/abs/pii/S092188900500134X

[62] A. Bouguerra, L. Karlsson, and A. Saffiotti, "Monitoring the execution of robot plans using semantic knowledge," *Robotics and Autonomous Systems (RAS)*, 2008. [Online]. Available: https://www.sciencedirect.com/science/article/abs/pii/S0921889008001152

[63] B. Wen, S. Dewan, and S. Birchfield, "Fast-foundationstereo: Real-time zero-shot stereo matching," *arXiv preprint arXiv:2512.11130*, 2025. [Online]. Available: https://arxiv.org/abs/2512.11130

[64] B. Tan, C. Sun, X. Qin, H. Adai, Z. Fu, T. Zhou, H. Zhang, Y. Xu, X. Zhu, Y. Shen, and N. Xue, "Masked depth modeling for spatial perception," *arXiv preprint arXiv:2601.17895*, 2026. [Online]. Available: https://arxiv.org/abs/2601.17895

[65] T. Silver, R. Chitnis, N. Kumar, W. McClinton, T. Lozano-Pérez, L. P. Kaelbling, and J. B. Tenenbaum, "Predicate invention for bilevel planning," in *AAAI Conference on Artificial Intelligence (AAAI)*, 2023. [Online]. Available: https://ojs.aaai.org/index.php/AAAI/article/view/26429

[66] N. Kumar, W. McClinton, R. Chitnis, T. Silver, T. Lozano-Pérez, and L. P. Kaelbling, "Learning efficient abstract planning models that choose what to predict," in *Conference on Robot Learning (CoRL)*, 2023. [Online]. Available: https://proceedings.mlr.press/v229/kumar23a.html

[67] Y. Liang, N. Kumar, H. Tang, A. Weller, J. B. Tenenbaum, T. Silver, J. F. Henriques, and K. Ellis, "Visualpredicator: Learning abstract world models with neuro-symbolic predicates for robot planning," in *International Conference on Learning Representations (ICLR)*, 2025. [Online]. Available: https://arxiv.org/abs/2410.23156

[68] A. Athalye, N. Kumar, T. Silver, Y. Liang, J. Wang, T. Lozano-Pérez, and L. P. Kaelbling, "From pixels to predicates: Learning symbolic world models via pretrained vision-language models," *Robotics and Automation Letters (RA-L)*, 2026. [Online]. Available: https://arxiv.org/abs/2501.00296

APPENDIX

A. cuTAMP Extensions

We made several extensions to cuTAMP [1] to improve real-world deployability:

M2T2 Grasp Integration. We support initializing grasp particles from M2T2 6-DoF grasp predictions, with collision filtering to reject grasps where the gripper would collide with the target object.

Oriented Bounding Box Surfaces. We added support for oriented bounding boxes (OBBs) as placement surfaces, with cost functions that penalize object placements near surface edges, and placement samplers that account for object extents during particle initialization.

Motion Planning Robustness. We increased motion planning attempts over constraint-satisfying particles since cuTAMP's collision representation (spheres) differs from cuRobo's low-level collision checks (oriented bounding boxes). When motion planning fails for path segments, we optionally relax collision checking thresholds as a fallback.

Efficient Movable Object Collision Handling. We optimized collision checks between movable objects by only evaluating costs after an object's action is activated. This allows us to handle objects that are initially in collision (e.g., due to clutter or convex hull overapproximation) by excluding them from collision penalties until moved.

Hardware Support. We added robot models for the Franka FR3 with Robotiq gripper and ZED Mini camera mount, including collision sphere approximations.

Task Planning Caching. Task planning becomes a bottleneck with many objects in the scene. We cache intermediate results over the task planner's tree search to reduce redundant computation.

B. Controller Implementation Details

For the DROID setup with the Franka FR3 arm, we implemented a joint impedance controller to track the planned trajectory waypoints (see §V) by computing joint torques at each control timestep:

$$\tau = K_p \odot (q_d - q) + K_d \odot (\dot{q}_d - \dot{q}) + \tau_{\text{coriolis}} + \tau_g + M\ddot{q}_d$$

where K_p and K_d are per-joint position and velocity gains, τ_{coriolis} compensates for Coriolis forces, τ_g compensates for gravity, M is the mass matrix, and $\ddot{q}_d$ is the desired acceleration estimated via filtered numerical differentiation of $\dot{q}_d$. The term $M\ddot{q}_d$ compensates for the robot's inertia.

The gains K_p and K_d were tuned to improve trajectory tracking, though the controller still exhibits small deviations during execution at high speeds (typically up to 5mm of resulting end-effector position error). We will open-source our controller implementation for the Franka FR3 and Panda robots upon acceptance.

For the UR5e, we instead use the `servoJ` primitive via Universal Robots' RTDE interface, with a high-gain motion phase followed by a settling phase to mitigate mechanical oscillation.

Fig. 9: **Object Segmentation.** SAM-2 generates eight pixel-level segmentation masks from the bounding boxes in Fig. 5c.

C. Additional Experiment Details

Table IV shows each evaluation scene with its language instruction and task progress metric.

Evaluation protocol. $\pi_{0.5}$-DROID is a reactive policy that runs continuously until manually terminated. TiPToP, by contrast, plans once and either produces a full trajectory or explicitly fails if no valid plan is found. We use a 30–60 second planning timeout for TiPToP.

In simulation, we terminated $\pi_{0.5}$-DROID trials after 60s or upon success and reset object configurations identically across all trials for each scene.

For real-world experiments run by the external evaluators (unmarked scenes in Table IV), $\pi_{0.5}$-DROID trials were terminated after 800 steps or upon success. They independently chose a step-based limit, which decouples evaluation from inference speed. Objects were reset to similar positions within the wrist camera's field of view using the same robot starting configuration.

For real-world experiments run by the system designers (scenes marked with † in Table IV), $\pi_{0.5}$-DROID trials were terminated after 120s or upon success. The longer timeout accommodates multi-step tasks. Since exact scene resets are not possible in the real world, we reset scenes by visually comparing against reference images, producing generally consistent configurations.

All termination limits are generous relative to typical task completion times, ensuring timeouts do not artificially limit $\pi_{0.5}$-DROID's performance. We ran all systems on an NVIDIA L4 (simulation), RTX 3080 Laptop (external evaluators), or RTX 4090 (system designers) GPU.

Completion time. In Table II, we report average completion time over successful trials only. For TiPToP, we set the `time_dilation_factor` in cuRobo to 0.6 in both simulation and real-world experiments. In the real-world experiments, we measure execution time using a remote iPad timer (Figure 3), which is automatically stopped upon robot execution for TiPToP, and manually stopped for $\pi_{0.5}$-DROID.

Failure Analysis. Our systematic failure analysis from Section VII-C was performed by collecting 173 trials of TiPToP execution over a range of different tasks (different from the evaluation tasks). For each trial, we selected a random

set of objects in a random initial pose on the tabletop, and provided an appropriate natural language goal given the objects and initial configuration. For each trial, we judged success manually and traced failures via logging and visualization.

Of the 173 trials, 52 were simple single-object pick-and-place tasks with no distractor objects (instruction: "put the object into the container"). The remaining 121 trials all included distractor objects on the table: 50 were single-object tasks with significant clutter (instruction: "put the smallest object into or onto the container"), 20 were single-object tasks with varied natural language goals (e.g. "put the soft yellow object into the box", "put the red thing into the box", "put the orange item in the receptacle"), 21 were two-object pick-and-place (e.g. "put the fruits in the orange bowl"), 20 were three-object pick-and-place (e.g. "serve all the non-fruit food on the tray"), 5 were four-object pick-and-place (instruction: "put all the cups with handles on the bin"), and 5 were five-object pick-and-place (instruction: "put the caffeinated beverages and coffee pods on the box").

D. Deployment on UR5e

We deployed TiPToP on a UR5e arm with a RealSense D435 wrist camera (bottom row of Figure 3). Adapting TiPToP to this new embodiment required:

- The robot URDF.
- Collision spheres for the robot geometry, automatically generated using tools like Ballpark or Foam.
- A cuRobo configuration file, following this guide from the cuRobo developers.
- Code changes in cuTAMP to load the new configuration files.
- Code changes in TiPToP to interface with the RealSense camera (via `pyrealsense2`) and the robot controller (via Universal Robots' Real-Time Data Exchange (RTDE) interface).

TiPToP's codebase provides abstractions that make adding new camera types or robot controllers straightforward. Given an existing robot controller, we completed all changes in approximately 2–3 hours.

FoundationStereo with a RealSense. For stereo input to FoundationStereo, we used the RealSense's left and right infrared (IR) sensors. This qualitatively resulted in noisier depth estimates than the DROID setup, which uses RGB stereo pairs from the ZED Mini, particularly on transparent, specular, and reflective objects. This is expected: active IR stereo struggles with such surfaces because the projected pattern does not reflect reliably.

Controller Implementation. We implement a joint-space trajectory tracking controller using the Universal Robots `servoJ` primitive via the RTDE interface. The controller interpolates sparse waypoints to a 125Hz command stream. We use a high proportional gain (400) during motion to minimize tracking error, then reduce the gain (300) during a settling phase with dwell waypoints at the end of the trajectory to mitigate mechanical oscillation.

E. Whiteboard Wiping Skill

Adding the whiteboard-wiping primitive (§VIII) required three localized changes, none of which modified the perception or execution infrastructure.

Semantic branch. We add two new predicates, `IsEraser` and `IsCleaned`, and extend the VLM goal-grounding prompt to translate instructions involving cleaning into conjunctions over these predicates (e.g., `IsCleaned(whiteboard)`).

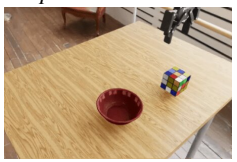
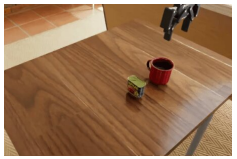
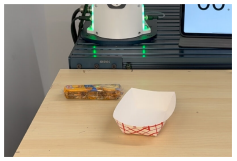
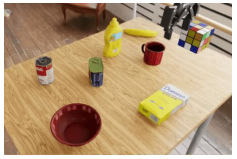
Planning. We define a new `Wipe` cuTAMP operator with preconditions that the robot is holding an eraser and the target is a surface, and an effect that marks the surface as cleaned. The task planner automatically sequences pick then wipe to satisfy an `IsCleaned` goal. During motion solving, `Wipe` hands off to a low-level wiping skill.

Execution. The wiping controller calls the VLM a second time to localize the region of interest (e.g., written text) on the surface via a bounding-box query. It reprojects the bounding-box corners into world coordinates using the existing point cloud, then executes a sequence of back-and-forth strokes covering the detected region using IK-based Cartesian control.

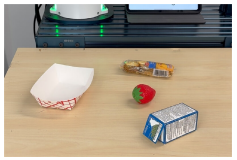
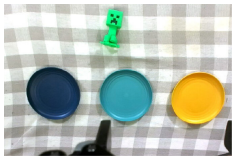
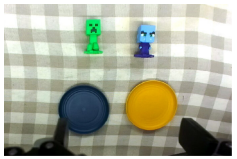
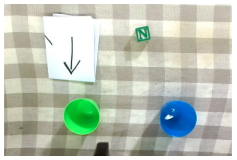
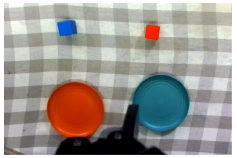
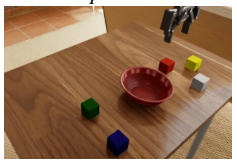
F. MolmoSpaces Integration

For the MolmoSpaces Pick and Pick & Place-NextTo task sets, TiPToP was extended to support the predicates `Holding(?movable)` and `Near(?movable, ?reference)`. Our method can be easily extended to support new predicates for novel task types. We leave for future work adding support for the Open/Close task set variants. Other policies on the leaderboard use multiple camera views. In contrast, TiPToP only uses the wrist camera and does not need external camera views. However, TiPToP assumes that objects of interest are in the starting view of the wrist camera. Thus, we begin every trajectory by first executing a motion to a constant start pose that positions the scene within the view of the wrist camera. TiPToP relies on depth data and camera pose as inputs. Whereas in real-world experiments, this depth data is derived from stereo RGB or infrared (IR) inputs, MolmoSpaces API allows querying for ground-truth simulation depth data directly.

TABLE IV: **Evaluation scene details.** Each scene shows an image of the task, its identifier (as referenced in Table I), language prompt, and task progress metric. Scenes are grouped by category: Simple, Distractor, Semantic, and Multi-step. † indicates tasks evaluated by the system designers. Unmarked scenes are evaluated by external evaluators not involved in the development of TiPToP. (sim) denotes tasks evaluated in simulation. Task progress metric numbers are reported in %; a + or − sign indicates that the particular denoted amount is added or subtracted from the overall score, and no sign indicates that the number is the absolute score for achieving that particular condition. Progress metrics may vary by the evaluator and the task. Some metrics penalize manipulating distractors while others do not.

Scene	Identifier / Language Prompt	Progress Metric	Scene	Identifier / Language Prompt	Progress Metric
Simple					
	Cube → bowl (sim) "put the cube in the bowl"	25% approach cube, 50% grasp, 75% approach bowl with cube, 100% place		Can → mug (sim) "put the can in the mug"	25% approach can, 50% grasp, 75% approach mug with can, 100% place
	Banana → bin (sim) "put banana in the bin"	25% approach banana, 50% grasp, 75% approach bin with banana, 100% place		Marker → tray "put the marker in the tray"	+25% touch marker, +25% grasp, +25% touch tray, +25% place
	Crackers → tray† "place the crackers onto the tray"	50% grasp crackers, 100% place			
Distractor					
	Meat can → sugar box (sim) "put the meat can on the sugar box"	25% approach meat can, 50% grasp, 75% approach box with meat can, 100% place		Coffee capsules → plate "put all of the coffee capsules onto the white plate"	+50% per capsule placed, −20% per distractor
	Turkish figs → plate "put the turkish figs onto the white plate"	+50% per fig placed, −20% per cashew		Cashews → plate "put the roasted cashews onto the white plate"	+50% per cashew placed, −20% per fig
	Red cubes → plate "put the red cubes onto the white plate"	+50% per cube placed, −20% if distractor placed		Fish → box "place the fish into the white box"	+50% pick fish, +50% place into white box

Continued on next page

Scene	Identifier / Language Prompt	Progress Metric	Scene	Identifier / Language Prompt	Progress Metric
	Crackers → tray (med.)[†] "place the crackers onto the tray"	+50% pick crackers, +50% place on the tray (no penalty for distractor)		PB crackers → tray (hard)[†] "place the peanut butter crackers onto the tray"	+50% pick crackers, +50% place on the tray (no penalty for distractor)
Semantic					
	Toy → matching plate "pick up the toy and place on the plate with similar color"	+50% pick toy, +50% place on teal or +30% place on blue		Creeper → plate "pick up the creeper and place onto the purple plate"	+50% pick creeper toy, +50% place onto purple plate
	Largest toy → plate "pick up the largest toy and place onto the purple plate"	+50% pick creeper, +50% place onto purple plate, −20% if attempt to place on distractor		Red A → color pile "pick up the red A and place on same color pile"	+50% pick red A block, +50% place onto red pile, −20% knock pile over
	Banana → box "pick up the banana and put it in the box"	+50% place banana into any box, +50% place into box with fruit (aims to test common sense of human selection)		N block → indicated cup "put the N block into the cup pointed to by the arrow"	+50% grasp N block, +50% place into cup pointed at
	Sort blocks by color "sort the blocks into opposite color plates"	+10% per block touched, +40% per correct place		Banana → matching plate "place banana into plate has similar color"	+50% pick banana, +50% place into orange plate
Multi-step					
	Color cubes → bowl (sim) "put 3 cubes into the bowl"	For up to 3 cubes (normalized to 100%): +5% approach cube, +10% grasp, +10% approach bowl with cube, +15% place		AirPods → cup "place airpods into the yellow cup"	+25% per AirPods picked, +25% per place, −20% distractor
	Pack pods → tray[†] "pack the coffee pods onto the rectangular tray"	For each of the 3 pods: +3.33% approach, +15% grasp, +0% place not in tray, +15% place touching tray		Pack pods → tray (obs.)[†] "pack the coffee pods onto the rectangular tray"	+12.5% pick can, +12.5% place s.t. it doesn't obstruct tray (or +25% for clearing can obstruction without pick-/place), for each of 3 pods: +5% for approaching pod, +10% for correct pick, +10% for correct place into tray

Continued on next page

Scene	Identifier / Language Prompt	Progress Metric	Scene	Identifier / Language Prompt	Progress Metric
	Aleve bottle → tray (obs.)[†] "put the small white aleve bottle into the cardboard tray"	+10% pick an obstacle object, +10% place obstacle s.t. unobstructs aleve, +30% pick aleve bottle (+50% if picked without clearing obstacles), +50% place bottle in tray		Three marbles → cup[†] "put only the marbles in the cup"	+16.67% for each pick of a marble, +16.67% for each place of a marble into the cup
	Marbles + cable[†] "put the small plastic bag of marbles into the black mesh bag, and the cable on top of the empty large plastic bag"	wire: +5% approach, +20% stable pick, +25% stable place atop plastic; marbles pouch: +5% approach, +20% pick, +25% place into mesh bag			

G. VLM Prompting Details.

As part of the perception module in Section IV, we use the following prompt for object detection and goal grounding:

```
Perform two tasks on this image based on the task instruction: "{task_instruction}".

TASK 1 - OBJECT DETECTION:
Detect and return bounding boxes for objects in the image.
- DO NOT include the robot, robot gripper, or table surface
- DO NOT include objects or surfaces irrelevant to the task or too far away to matter (e.g. walls, things on the wall that are
    far away, things on the floor below the table, people who might be in the scene, etc.)
- Limit to 25 objects
- If an object appears multiple times, name them by unique characteristics (color, size, position, etc.). If they seem the same,
     then just use numbers (e.g. 'soda_can1' and 'soda_can2', ... for identical-looking soda cans)
- An object **cannot have the same name** as another under any circumstance.
- Format: normalized coordinates 0-1000 as integers

Be very careful to identify objects and name them in a way that's relevant to the task. If the task involves picking up a red
    apple, make sure that 'red' appears in the name of the apple.


TASK 2 - TASK TRANSLATION:
Translate this natural language instruction into some conjunction of formal predicates:

AVAILABLE PREDICATES:
- on(movable, surface): Object A is placed on top of object B

It is very important that the goal is exact: use your visual recognition, common-sense and reasoning abilities to make sure the
    goal expression is perfectly accurate.

For instance - for the task "throw away the trash in the bin" when there is a bin, an open empty chips packet, an empty soda can
    , a closed and full soda bottle, and several full candy bars on the table, the goal should be:

"predicates": [
    {{"name": "on", "args": ["chips_packet", "bin"]}},
    {{"name": "on", "args": ["soda_can", "bin"]}},
]
```

This is because only the chips and soda are empty and clearly trash. Everything else is still usable!

Return a single JSON object with this structure (no code fencing):

```
{{
    "bboxes": [
        {{"box_2d": [ymin, xmin, ymax, xmax], "label": "object name"}},
        ...
    ],
    "predicates": [
        {{"name": "predicate_name", "args": ["object1", "object2"]}},
        ...
    ]
}}
```

Use the object labels you detect in Task 1 when creating predicates in Task 2.
Only reference objects that you actually detected in the image.