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

William Shen[1*], Nishanth Kumar[1*], Sahit Chintalapudi[1], Ryan Lindeborg[1], Jie Wang[2], Christopher Watson[2], Edward S. Hu[2], Jing Cao[1], Dinesh Jayaraman[2], Leslie Pack Kaelbling[1], Tomás Lozano-Pérez[1]

[1]MIT CSAIL, [2]University of Pennsylvania

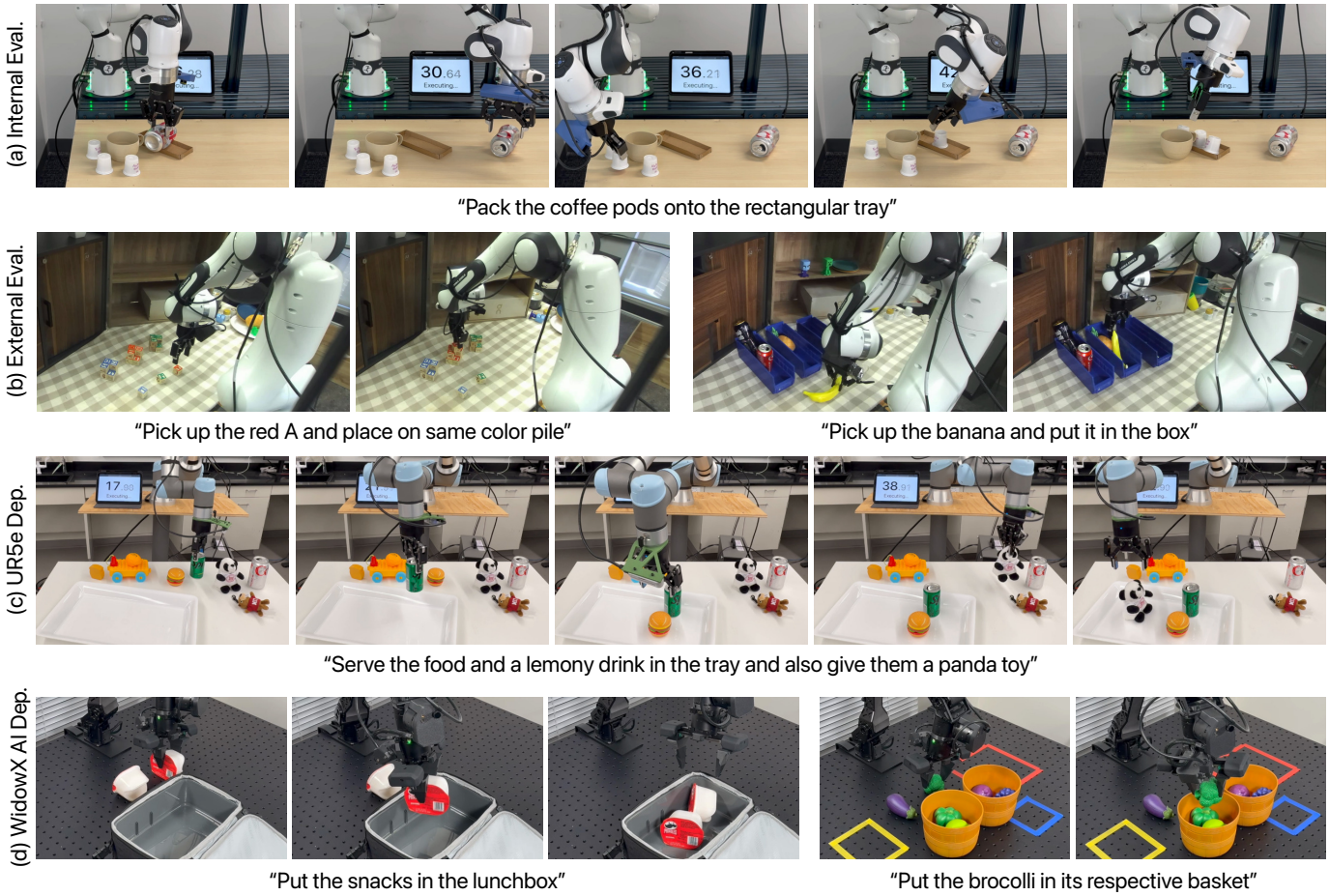


Fig. 1: **TiPToP operating over environments and embodiments.** (a) A long-horizon packing task on a DROID setup, where TiPToP first clears an obstructing Coke can. (b) Two semantic pick-and-place tasks on an external DROID setup. (c) TiPToP deployed on a UR5e and (d) on a Trossen WidowX AI.

***Abstract*—We present TiPToP, a modular manipulation system that integrates pretrained 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 serve as a reproducible baseline and to enable further research on modular manipulation systems. Project website and code: tiptop-robot.github.io**

*Equal contribution. Correspondence to {willshen,njk}@mit.edu

I. Introduction

A longstanding goal of robotics research is to build a general-purpose manipulation system that works out-of-the-box: one that can be deployed on arbitrary robots to perform language-specified tasks on arbitrary objects, with no object, environment, or embodiment-specific tuning. Recent progress in vision and language foundation models has expanded the

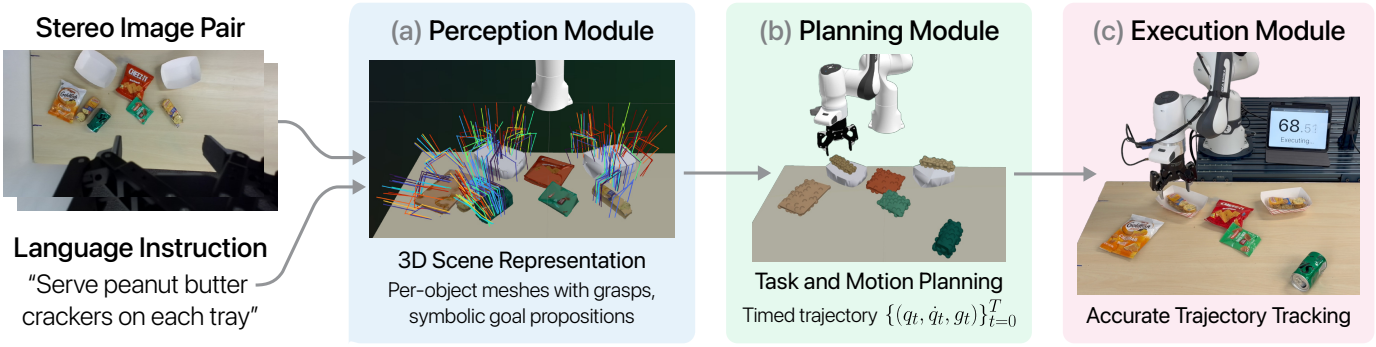


Fig. 2: **TiPToP System Overview.** From a stereo RGB pair and language instruction $\mathcal{L}$, TiPToP outputs joint trajectories with gripper commands. (a) Perception builds an object-centric 3D scene representation via depth estimation, grasp prediction, object detection, and segmentation. (b) Planning runs GPU-parallelized TAMP (cuTAMP) [57] to find feasible manipulation plans. (c) Execution tracks the plan with a joint impedance controller.

capabilities available to robotics, from open-vocabulary detection, grounding, and common-sense reasoning [29, 19, 7, 50] to depth estimation, grasp generation, and 3D shape completion [70, 74, 56, 75]. However, converting these capabilities into reliable robot manipulation behavior remains an open problem.

We introduce **TiPToP** (**T**iPToP **i**s a **P**lanner **T**hat just works **o**n **P**ixels), a complete manipulation system that pairs the perception and world knowledge of pretrained foundation models with test-time search via Task and Motion Planning (TAMP) [27]. TiPToP is built on three principles. First, it is *modular*: separate *perception*, *planning*, and *execution* modules can be improved or replaced in isolation and let us trace failures to a specific module. Second, it is *compositional*: a new skill requires only symbolic predicates, an operator, and a parameterized controller, which the planner composes with existing skills automatically. Third, it is *zero-shot*: built from off-the-shelf models, TiPToP transfers to new embodiments, scenes, and in-skill tasks with no training or robot data. TiPToP deploys on supported robots[1] in under an hour with only camera calibration needed.

We evaluate TiPToP against state-of-the-art Vision-Language-Action (VLA) models [52, 40, 22, 73] and World Action Models (WAMs) [72, 53, 2] given the same image and language input. On real hardware, an external team independently deployed and evaluated TiPToP on the DROID platform [35] against $\pi_{0.5}$-DROID [52], 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 [5], we evaluate all 9 pick and pick-and-place tasks (9,000 episodes). Despite using *zero* robot training data, TiPToP outperforms every VLA and WAM trained without in-distribution (ID) data on 5 of 9 tasks, and even beats ID-trained models on *Pick & Place-NextTo*.

Tracing failures to modules, we find most stem from unstable grasps under open-loop execution, pointing toward integrating TiPToP's planning with learned, closed-loop policies that scale with data. Its compositionality and embodiment-agnostic design make it easy to extend: we add a whiteboard-wiping skill on DROID, swap in better 3D reconstruction models, and deploy on a UR5e and a Trossen WidowX AI.

Overall, we make two contributions: (1) TiPToP, an open-source, extensible planning-based manipulation system integrating foundation models with GPU-accelerated TAMP that needs no robot training data and supports real-world deployment and simulation; and (2) an empirical study and failure analysis benchmarking TiPToP against existing approaches across real-world hardware, simulation, and an independent large-scale benchmark. We release TiPToP as a reproducible, easy-to-use 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 [64, 70, 30] predict dense depth maps from RGB image pairs. Foundation models for grasp generation [46, 62, 74] predict 6-DoF grasp poses from point clouds. SAM [37] and SAM-2 [54] provide promptable segmentation from bounding boxes or points, enabling precise object boundary delineation. Shape completion models such as SAM-3D [56] and RecGen [75] reconstruct full 3D object meshes and poses from partial RGB(-D) views. Vision-Language Models (VLMs) [29, 19, 7, 50] 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. These models each supply a perceptual capability but do not perform manipulation. SceneComplete [1] composes them into a 3D scene representation to enable manipulation. TiPToP takes this further, integrating grasp generation and a VLM with a full TAMP system to enable complex multi-step manipulation.

Vision-Language-Action Models. VLAs leverage VLM backbones trained on additional diverse robot data to enable language-conditioned control [12, 36, 9, 28] and demonstrate that transformer-based policies trained on large datasets (e.g., the Open X-Embodiment dataset [49]) can generalize across tasks and objects.

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

π_0 [10] introduced a flow-matching architecture, which was subsequently extended in $\pi_{0.5}$ [52] 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 [4] trains manipulation policies entirely in simulation using the MolmoSpaces ecosystem [5]. 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 embodiment-specific 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 allows failures to be traced to individual components. This modularity also allows individual components (perception, planning, execution) to be debugged and improved independently without modifying the entire system.

World Models. World models learn to predict future observations from internet-scale video, representing dynamics directly in pixel space. One family of approaches first generates a video plan, then recovers low-level robot actions with a separate inverse dynamics model [20, 55, 13]. World action models (WAMs) instead jointly predict video and actions [72, 53, 2]. We compare against these models, which generate an interpretable video rollout but still require fine-tuning on robot data.

Task and Motion Planning. TAMP algorithms jointly solve discrete task planning and continuous motion planning [33, 27], complementing the open-vocabulary perception and grounding VLMs provide. Common approaches use sampling [25, 26] or optimization [65, 66] to satisfy continuous constraints, but most require detailed object geometries given *a priori*. Closest to ours, Curtis et al. [16] integrate learned perception modules with PDDLStream [26] for long-horizon manipulation of unknown objects. TiPToP differs in three ways: (1) we use cuTAMP [57], a GPU-parallelized optimization-based planner that is far more efficient than sampling-based PDDLStream and search-then-sample bilevel planners [60, 15]; (2) we leverage larger foundation models trained on more data; and (3) we package TiPToP to install on any embodiment providing a camera, gripper, URDF, and trajectory-tracking controller.

Modular Robotic Planning Systems. Modular systems decompose manipulation into perception, high-level planning, and low-level control. Early symbolic planners such as STRIPS [23] on the Shakey robot [47] showed the power of symbolic plans but required hand-engineered world models. Recent neurosymbolic work uses LLMs to sequence pretrained skills [3, 31], generate robot programs [42, 59], construct 3D value maps [32], or pair VLMs with skills [44], and to drive TAMP by proposing task plans, subgoals, or constraints [68, 17, 71, 39]. Many of these systems sequence discrete skills, and none are packaged as a complete, embodiment-agnostic system deployable on real robots. TiPToP is a fully modular pipeline that directly consumes image and natural language input, jointly plans discrete task sequences and continuous collision-free trajectories, and is packaged to install easily across a variety of embodiments.

III. Problem Formulation and System Overview

We study language-conditioned manipulation: given a natural language instruction $\mathcal{L}$ and a robot with known kinematics, produce actions that accomplish it. At each timestep t, a policy π receives RGB observations $\mathbf{o}_t$ from one or more cameras and 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 interface lets us directly compare two paradigms that instantiate π in fundamentally different ways: end-to-end learned policies that map observations to actions through a single trained network, including VLAs and WAMs, and TiPToP, which composes frozen foundation models with test-time planning.

A. TiPToP

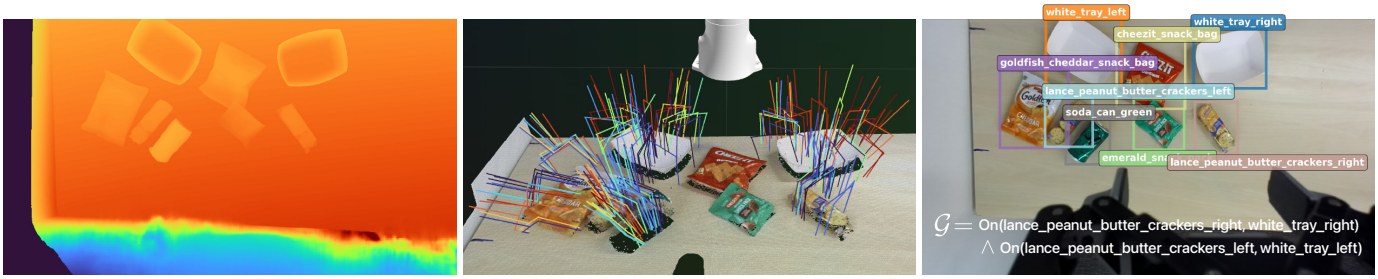
End-to-end policies instantiate π as a single trained network that runs closed-loop at high frequency. Our representative VLA, $\pi_{0.5}$-DROID [52], runs at 15 Hz: at each timestep t it observes $\mathbf{o}_t = (I_t^{\text{wrist}}, I_t^{\text{ext}})$, monocular RGB images from the wrist and external cameras, together with the current joint and gripper positions (q_t, g_t), and 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.

TiPToP instead instantiates π as a *planner*, differing on three axes: it senses *once* rather than continuously, plans a full trajectory rather than emitting short action chunks, and executes *open-loop* rather than reacting to new observations[2]. 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 2): (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 [57] 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 Robotiq 2F-85 gripper and 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 2. This task requires identifying peanut butter crackers among

[2]TiPToP can be made closed-loop and this is an important direction for future work (see §IX).



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

Fig. 3: **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 ($\texttt{On}(a, b)$ specifies that object a should be placed on object or surface b).

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.

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 [70], 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 3a).

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 3b).

Grasp Generation. We use M2T2 [74] 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). M2T2 reasons over the full scene and makes predictions 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 3b). 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.

We also tried GraspGen [46], but it requires segmented object point clouds and does not consider scene geometry for predicting grasps, requiring additional overhead for collision checking. We also considered AnyGrasp [21], but its license application process complicates out-of-the-box deployment.

B. Semantic Branch

Object Detection and Goal Grounding. We query Gemini Robotics-ER 1.5 [28], 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 between objects) over detected objects. The system was originally developed to support just the $\texttt{On}(a, b)$ 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, producing $\mathcal{G} = \texttt{On}(\text{crackers}_{\text{right}}, \text{tray}_{\text{right}}) \wedge \texttt{On}(\text{crackers}_{\text{left}}, \text{tray}_{\text{left}})$ (Figure 3c).

Object Segmentation. For each detected bounding box, we use SAM-2 [54] 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 seg-

Fig. 4: **Shape-Completion Mode.** From the observed scene (left), RecGen [75] faithfully recovers the full geometry of each object (right), including the elephant watering can.

mentation 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 [24] 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 $\mathbf{p}^{\text{world}}$, 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 [75], a foundation model that generates a full mesh and 6-DoF pose for each masked object (Figure 4).

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. Although it strongly over-approximates concave objects such as bananas, we find it sufficient for the majority of tasks we evaluate. The shape-completion mode provides higher-fidelity meshes for complex objects at substantially greater compute cost. RecGen requires $\approx 10\,\text{s}$ per object on an RTX 3090, though this trivially parallelizes over GPUs.

Grasp-to-Object Assignment. Each grasp predicted by M2T2 is assigned to the nearest object by querying its contact point against a KDTree [8] 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 [57], 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.

Plan Skeleton Enumeration. Given the symbolic goal $\mathcal{G}$, cuTAMP uses a PDDL-style symbolic planner [45] 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 2), 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 cracker 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 [61], 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 improve its reliability 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, 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) form a substantial part of our contribution of a working open-source system.

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.

Design Decision: 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 [52], 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 to answer Q1–Q3 by comparing TiPToP against existing end-to-end approaches 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 [48] (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 [67], 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).

	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.

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 performs better 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.

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 modules (included in 'Time').

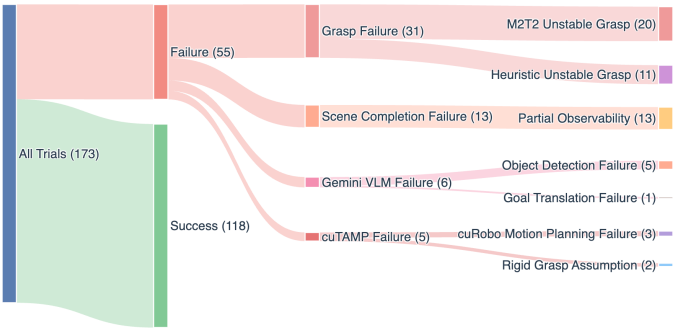


Fig. 5: **Failure Analysis.** Sankey diagram showing outcomes of 173 trials. The most common failure modes are grasping failures (missed or unstable grasps), followed by scene-completion errors, VLM detection errors, then cuTAMP failures.

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).

B. MolmoSpaces

MolmoSpaces [5] 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. As of the writing of this paper, TiPToP is the only non-ID method on the public leaderboard that uses zero robot or simulator demonstrations.

Table III reports the oracle[3] 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 [73]), ranking first among non-ID methods on 5 of 9 tasks. These five 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 5).

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.

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

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

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) [73]	**73.8**	**46.0**	**78.4**	47.9	47.2	30.1	13.8	15.9	18.3	41.3
Cosmos3-Nano (WAM) [2]	65.5	35.2	66.5	32.3	32.2	25.7	26.0	22.9	**34.3**	37.8
Psi-R2 (WAM) [53]	73.1	41.5	50.2	23.4	25.8	27.2	2.6	2.9	1.5	27.6
MolmoAct2-DROID (VLA) [22]	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) [52]	36.4	13.6	18.1	6.4	7.0	8.0	12.0	10.3	10.4	13.6
In-Distribution Methods: fine-tuned on MolmoSpaces and MolmoBot Simulator data										
MolmoBot Best (VLA) [4]	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.

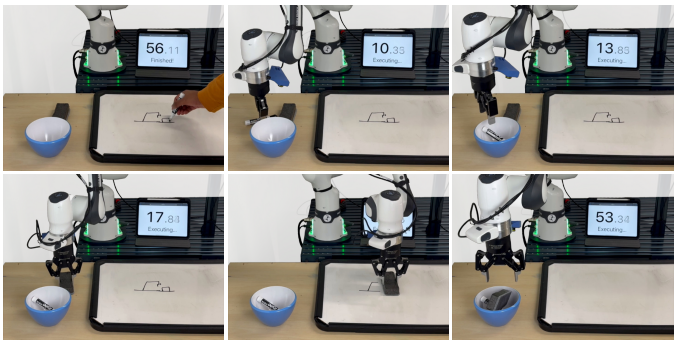


Fig. 6: **Wiping.** We demonstrate that TiPToP can be straightforwardly extended to perform wiping in addition to pick-and-place. Task instruction: "erase the whiteboard and put everything into the bowl".

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's modular design provides several advantages. We choose to demonstrate two: the system deploys to new robot embodiments without retraining, and it accepts new manipulation skills through small, localized additions.

Deploying to new embodiments. We deployed TiPToP on a UR5e arm with a wrist-mounted RealSense D435 camera (Figure 1c). 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 1d).

Adding new skills. We added a whiteboard-wiping primitive that erases writing from a surface with an eraser (Figure 6) through three localized changes, none of which touched the perception or execution infrastructure: (1) two new predicates (`IsEraser`, `IsCleaned`) and a goal-grounding prompt extension in the semantic branch; (2) a `Wipe` cuTAMP operator that the task planner automatically sequences after a pick; and (3) 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

Open-loop execution. This is the single most impactful limitation: our failure analysis (Figure 5) shows that grasping failures account for over half of all observed failures, many of which could be recovered from by re-attempting the grasp. The most direct implementation to close the loop is to re-run perception and planning after each pick-and-place step, enabling recovery from failed grasps or unexpected object movement [16, 41, 51, 11].

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 [69, 63] could further improve point cloud quality, and better grasp prediction models would also help address TiPToP's most common observed 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. One natural way to integrate these approaches is to use end-to-end policies as reactive skills within TiPToP. This would enable robust and reactive individual skills (e.g., opening and closing articulated objects, folding, cable manipulation, contact-rich tasks) as well as constraint-aware skill chaining and long-horizon behavior. Integrating such skills requires specifying their abstract preconditions and effects, which could be engineered or learned from data [58, 38, 43, 6], so the planner can reason about when to invoke them. Another way to integrate these approaches is to call them separately for different sub-steps of long-horizon tasks (e.g., a VLA opens a

drawer, then TiPToP performs multi-object pick-and-place to pack that drawer, before a VLA closes it).

Belief-space planning. Extending cuTAMP to operate in belief space would enable reasoning about uncertainty in object poses, grasp outcomes, and partially observable state [34, 18, 14]. This could also enable information-gathering actions (e.g., moving the camera to observe an occluded region before planning) and more robust action selection under perceptual uncertainty.

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 a dominant bottleneck for the current version of the system.

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. Additionally, 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 drives further research and progress toward broadly competent and generalizable manipulation systems.

Acknowledgments

We gratefully acknowledge support from NSF grant 2214177; from AFOSR grant FA9550-22-1-0249; from ONR MURI grants N00014-22-1-2740 and N00014-24-1-2603; from the MIT Quest for Intelligence; and from the Robotics and AI Institute. We thank Jesse Zhang for testing TiPToP at the University of Washington. We thank Wenlong Huang for help setting up FoundationStereo to improve point cloud accuracy, as well as several helpful discussions. We thank Omar Rayyan, Maximilian Argus, Wilbert Pumacay, and Mahi Shafiullah for their encouragement and invaluable debugging support, which enabled us to integrate TiPToP with MolmoSpaces, and for adding our results to the public leaderboard. We also thank Tom Silver, Chris Agia, Joey Hejna, Karl Pertsch, Danny Driess, and Fabio Ramos for helpful discussions and feedback on earlier drafts of this work.

Author Contributions

William Shen and **Nishanth Kumar** contributed equally to this work. William adapted and improved the core cuTAMP system to be suitable (simpler to use, faster) for our purposes. Nishanth implemented the perception interface to Gemini and SAM. Both William and Nishanth worked on integrating additional models (FoundationStereo, M2T2) into the system, packaging all components to be easily used, benchmarking system capabilities, and writing the paper. They also supported the MolmoSpaces integration and helped analyze and present results.

Sahit Chintalapudi implemented and packaged the control stack for the Franka Panda and FR3 robots. He also helped run quantitative experiments to investigate TiPToP's failure modes, and helped make figures and edit the paper.

Ryan Lindeborg led the TiPToP integration with MolmoSpaces and gathered and analyzed the results. He also deployed TiPToP on his Trossen WidowX AI robot and provided installation and debugging feedback.

Jie Wang led the evaluations conducted at the University of Pennsylvania (Penn), and assisted with analysis and experimental design.

Christopher Watson set up TiPToP at Penn and assisted with evaluations, experimental design and analysis.

Edward S. Hu assisted with TiPToP setup at Penn and contributed to experimental design and analysis.

Jing Cao set up the IsaacSim simulator and ran simulation experiments comparing $\pi_{0.5}$-DROID to TiPToP, and analyzed the results.

Dinesh Jayaraman advised the evaluations at the University of Pennsylvania and provided lab resources.

Leslie Pack Kaelbling and **Tomás Lozano-Pérez** provided several helpful system implementation and task suggestions, and strongly encouraged that the code should be easy to install. They helped edit the paper, and also provided several more suggestions for improvement, the bulk of which have been left for future work.

References

[1] Aditya Agarwal, Gaurav Singh, Bipasha Sen, Tomás Lozano-Pérez, and Leslie Pack Kaelbling. Scenecomplete: Open-world 3d scene completion in cluttered real world environments for robot manipulation. *IEEE Robotics and Automation Letters (RA-L)*, 2025. URL https://arxiv.org/abs/2410.23643.

[2] Niket Agarwal, Arslan Ali, Jon Allen, Martin Antolini, Adeline Aubame, Alisson Azzolini, Junjie Bai, Maciej Bala, Yogesh Balaji, Josh Bapst, et al. Cosmos 3: Omnimodal world models for physical ai. *arXiv preprint arXiv:2606.02800*, 2026. URL https://arxiv.org/abs/2606.02800.

[3] 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 *Conference on Robot Learning (CoRL)*, 2022. URL https://arxiv.org/abs/2204.01691.

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

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

[6] Ashay Athalye, Nishanth Kumar, Tom Silver, Yichao Liang, Jiuguang Wang, Tomás Lozano-Pérez, and Leslie Pack Kaelbling. From pixels to predicates: Learning symbolic world models via pretrained vision-language models. *Robotics and Automation Letters (RA-L)*, 2026. URL https://arxiv.org/abs/2501.00296.

[7] Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, et al. Qwen3-vl technical report. *arXiv preprint arXiv:2511.21631*, 2025. URL https://arxiv.org/abs/2511.21631.

[8] Jon Louis Bentley. Multidimensional binary search trees used for associative searching. *Communications of the ACM (CACM)*, 1975. URL https://dl.acm.org/doi/10.1145/361002.361007.

[9] Johan Bjorck, Fernando Castañeda, Nikita Cherniadev, Xingye Da, Runyu Ding, Linxi Fan, Yu Fang, Dieter Fox, Fengyuan Hu, Spencer Huang, Joel Jang, Zhenyu Jiang, Jan Kautz, Kaushil Kundalia, Lawrence Lao, Zhiqi Li, Zongyu Lin, Kevin Lin, Guilin Liu, Edith Llontop, Loic Magne, Ajay Mandlekar, Avnish Narayan, Soroush Nasiriany, Scott Reed, You Liang Tan, Guanzhi Wang, Zu Wang, Jing Wang, Qi Wang, Jiannan Xiang, Yuqi Xie, Yinzhen Xu, Zhenjia Xu, Seonghyeon Ye, Zhiding Yu, Ao Zhang, Hao Zhang, Yizhou Zhao, Ruijie Zheng, and Yuke Zhu. Gr00t n1: An open foundation model for generalist humanoid robots. *arXiv preprint arXiv:2503.14734*, 2025. URL https://arxiv.org/abs/2503.14734.

[10] Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, et al. π_0: A vision-language-action flow model for general robot control. In *Robotics: Science and Systems (RSS)*, 2025. URL https://arxiv.org/abs/2410.24164.

[11] Abdelbaki Bouguerra, Lars Karlsson, and Alessandro Saffiotti. Monitoring the execution of robot plans using semantic knowledge. *Robotics and Autonomous Systems (RAS)*, 2008. URL https://www.sciencedirect.com/science/article/abs/pii/S0921889008001152.

[12] Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Xi Chen, Krzysztof Choromanski, Tianli Ding, Danny Driess, Avinava Dubey, Chelsea Finn, et al. Rt-2: Vision-language-action models transfer web knowledge to robotic control. In *Conference on Robot Learning (CoRL)*, 2023. URL https://proceedings.mlr.press/v229/zitkovich23a.html.

[13] Boyuan Chen, Tianyuan Zhang, Haoran Geng, Kiwhan Song, William T. Freeman, Jitendra Malik, Russ Tedrake, Vincent Sitzmann, and Yilun Du. Large video planner enables generalizable robot control, 2025. URL http://arxiv.org/abs/2512.15840.

[14] Sahit Chintalapudi, Leslie Pack Kaelbling, and Tomás Lozano-Pérez. Bi-level belief space search for compliant part mating under uncertainty. *arXiv preprint arXiv:2409.15774*, 2024. URL https://arxiv.org/abs/2409.15774.

[15] Rohan Chitnis, Tom Silver, Joshua B. Tenenbaum, Tomás Lozano-Pérez, and Leslie Pack Kaelbling. Learning neuro-symbolic relational transition models for bilevel planning. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2022. URL https://arxiv.org/abs/2105.14074.

[16] Aidan Curtis, Xiaolin Fang, Leslie Pack Kaelbling, Tomás Lozano-Pérez, and Caelan Reed 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. URL https://arxiv.org/abs/2108.04145.

[17] Aidan Curtis, Nishanth Kumar, Jing Cao, Tomás Lozano-Pérez, and Leslie Pack Kaelbling. Trust the proc3s: Solving long-horizon robotics problems with llms and constraint satisfaction. In *Conference on Robot Learning (CoRL)*, 2024. URL https://arxiv.org/abs/2406.05572.

[18] Aidan Curtis, George Matheos, Nishad Gothoskar, Vikash Mansinghka, Joshua Tenenbaum, Tomás Lozano-Pérez, and Leslie Pack Kaelbling. Partially observable task and motion planning with uncertainty and risk awareness. In *Robotics: Science and Systems (RSS)*, 2024. URL https://www.roboticsproceedings.org/rss20/p118.pdf.

[19] Matt Deitke, Christopher Clark, Sangho Lee, Rohun Tripathi, Yue Yang, Jae Sung Park, Mohammadreza Salehi, Niklas Muennighoff, Kyle Lo, Luca 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. URL https://arxiv.org/abs/2409.17146.

[20] Yilun Du, Mengjiao Yang, Bo Dai, Hanjun Dai, Ofir Nachum, Joshua B. Tenenbaum, Dale Schuurmans, and Pieter Abbeel. Learning universal policies via text-guided video generation, 2023. URL https://arxiv.org/abs/2302.00111.

[21] Hao-Shu Fang, Chenxi Wang, Hongjie Fang, Minghao Gou, Jirong Liu, Hengxu Yan, Wenhai Liu, Yichen Xie, and Cewu Lu. Anygrasp: Robust and efficient grasp perception in spatial and temporal domains. *IEEE Transactions on Robotics (T-RO)*, 2023. URL https://ieeexplore.ieee.org/document/10167687.

[22] Haoquan Fang, Jiafei Duan, Donovan Clay, Sam Wang, Shuo Liu, Weikai Huang, Xiang Fan, Wei-Chuan Tsai, Shirui Chen, Yi Ru Wang, Shanli Xing, Jaemin Cho, Jae Sung Park, Ainaz Eftekhar, Peter Sushko, Karen Farley, Angad Wadhwa, Cole Harrison, Winson Han, Ying-Chun Lee, Eli VanderBilt, Rose Hendrix, Suveen Ellawela, Lucas Ngoo, Joyce Chai, Zhongzheng Ren, Ali Farhadi, Dieter Fox, and Ranjay Krishna. MolmoAct2: Action reasoning models for real-world deployment.

2026. URL https://arxiv.org/abs/2605.02881.

[23] Richard E. Fikes and Nils J. Nilsson. Strips: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence*, 1971. URL https://www.sciencedirect.com/science/article/abs/pii/0004370271900105.

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

[25] Caelan Reed Garrett, Tomás Lozano-Pérez, and Leslie Pack Kaelbling. FFRob: Leveraging symbolic planning for efficient task and motion planning. *International Journal of Robotics Research (IJRR)*, 2018. URL https://journals.sagepub.com/doi/abs/10.1177/0278364917739114.

[26] Caelan Reed Garrett, Tomás Lozano-Pérez, and Leslie Pack Kaelbling. PDDLStream: Integrating symbolic planners and blackbox samplers via optimistic adaptive planning. In *International Conference on Automated Planning and Scheduling (ICAPS)*, 2020. URL https://ojs.aaai.org/index.php/ICAPS/article/view/6739.

[27] Caelan Reed Garrett, Rohan Chitnis, Rachel Holladay, Beomjoon Kim, Tom Silver, Leslie Pack Kaelbling, and Tomás Lozano-Pérez. Integrated task and motion planning. *Annual Review of Control, Robotics, and Autonomous Systems*, 2021. URL https://www.annualreviews.org/doi/full/10.1146/annurev-control-091420-084139.

[28] Gemini Robotics Team, Saminda Abeyruwan, Joshua Ainslie, Jean-Baptiste Alayrac, Montserrat Gonzalez Arenas, Travis Armstrong, Ashwin Balakrishna, Robert Baruch, Maria Bauza, Michiel Blokzijl, et al. Gemini robotics: Bringing ai into the physical world. *arXiv preprint arXiv:2503.20020*, 2025. URL https://arxiv.org/abs/2503.20020.

[29] Gemini Team, Rohan Anil, Sebastian Borgeaud, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, Katie Millican, et al. Gemini: A family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*, 2023. URL https://arxiv.org/abs/2312.11805.

[30] Xianda Guo, Chenming Zhang, Youmin Zhang, Ruilin Wang, Dujun Nie, Wenzhao Zheng, Matteo Poggi, Hao Zhao, Mang Ye, Qin Zou, and Long Chen. Stereo anything: Unifying zero-shot stereo matching with large-scale mixed data. *arXiv preprint arXiv:2411.14053*, 2024. URL https://arxiv.org/abs/2411.14053.

[31] 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 *Conference on Robot Learning (CoRL)*, 2022. URL https://arxiv.org/abs/2207.05608.

[32] Wenlong Huang, Chen Wang, Ruohan Zhang, Yunzhu Li, Jiajun Wu, and Li Fei-Fei. Voxposer: Composable 3d value maps for robotic manipulation with language models. In *Conference on Robot Learning (CoRL)*, 2023. URL https://proceedings.mlr.press/v229/huang23b.html.

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

[34] Leslie Pack Kaelbling and Tomás Lozano-Pérez. Integrated task and motion planning in belief space. *International Journal of Robotics Research (IJRR)*, 2013. URL https://journals.sagepub.com/doi/10.1177/0278364913484072.

[35] Alexander Khazatsky, Karl Pertsch, Suraj Nair, Ashwin Balakrishna, Sudeep Dasari, Siddharth Karamcheti, Soroush Nasiriany, Mohan Kumar Srirama, Lauryn Luo, Kathy Vuong, et al. Droid: A large-scale in-the-wild robot manipulation dataset. In *Robotics: Science and Systems (RSS)*, 2024. URL https://arxiv.org/abs/2403.12945.

[36] Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Foster, Grace Lam, Pannag Sanketi, et al. Openvla: An open-source vision-language-action model. *arXiv preprint arXiv:2406.09246*, 2024. URL https://arxiv.org/abs/2406.09246.

[37] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C. Berg, Wan-Yen Lo, Piotr Dollár, and Ross Girshick. Segment anything. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023. URL https://arxiv.org/abs/2304.02643.

[38] Nishanth Kumar, Willie McClinton, Rohan Chitnis, Tom Silver, Tomás Lozano-Pérez, and Leslie Pack Kaelbling. Learning efficient abstract planning models that choose what to predict. In *Conference on Robot Learning (CoRL)*, 2023. URL https://proceedings.mlr.press/v229/kumar23a.html.

[39] Nishanth Kumar, William Shen, Fabio Ramos, Dieter Fox, Tomás Lozano-Pérez, Leslie Pack Kaelbling, and Caelan Reed Garrett. Open-world task and motion planning via vision-language model inferred constraints. *arXiv preprint arXiv:2411.08253*, 2024. URL https://arxiv.org/abs/2411.08253.

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

[41] Martin Levihn, Leslie Pack Kaelbling, Tomás Lozano-Pérez, and Mike Stilman. Foresight and reconsideration in hierarchical planning and execution. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2013. URL https://dspace.mit.edu/handle/1721.1/90271.

[42] Jacky Liang, Wenlong Huang, Fei Xia, Peng Xu, Karol Hausman, Brian Ichter, Pete Florence, and Andy Zeng. Code as policies: Language model programs for embodied control. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2023. URL https:

//arxiv.org/abs/2209.07753.

[43] Yichao Liang, Nishanth Kumar, Hao Tang, Adrian Weller, Joshua B. Tenenbaum, Tom Silver, João F. Henriques, and Kevin Ellis. Visualpredicator: Learning abstract world models with neuro-symbolic predicates for robot planning. In *International Conference on Learning Representations (ICLR)*, 2025. URL https://arxiv.org/abs/2410.23156.

[44] Peiqi Liu, Yaswanth Orru, Jay Vakil, Chris Paxton, Nur Muhammad Mahi Shafiullah, and Lerrel Pinto. Demonstrating ok-robot: What really matters in integrating open-knowledge models for robotics. In *Robotics: Science and Systems (RSS)*, 2024. URL https://arxiv.org/abs/2401.12202.

[45] Drew McDermott, Malik Ghallab, Adele E. Howe, Craig A. Knoblock, Ashwin Ram, Manuela M. Veloso, Daniel S. Weld, and David E. Wilkins. PDDL: The planning domain definition language, 1998. URL https://www.semanticscholar.org/paper/PDDL-the-planning-domain-definition-language-McDermott-Ghallab/d82c6b8081343b2eae63d45feefe630233ad60e1.

[46] Adithyavairavan Murali, Balakumar Sundaralingam, Yu-Wei Chao, Wentao Yuan, Jun Yamada, Mark Carlson, Fabio Ramos, Stan Birchfield, Dieter Fox, and Clemens Eppner. Graspgen: A diffusion-based framework for 6-dof grasping with on-generator training. *arXiv preprint arXiv:2507.13097*, 2025. URL https://arxiv.org/abs/2507.13097.

[47] Nils J. Nilsson. Shakey the robot. Technical report, SRI International, Artificial Intelligence Center, 1984. URL https://ai.stanford.edu/~nilsson/OnlinePubs-Nils/shakey-the-robot.pdf.

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

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

[50] OpenAI. Gpt-4o system card. *arXiv preprint arXiv:2410.21276*, 2024. URL https://arxiv.org/abs/2410.21276.

[51] Ola Pettersson. Execution monitoring in robotics: A survey. *Robotics and Autonomous Systems (RAS)*, 2005. URL https://www.sciencedirect.com/science/article/abs/pii/S092188900500134X.

[52] Physical Intelligence, Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Manuel Y. Galliker, Dibya Ghosh, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Devin LeBlanc, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Allen Z. Ren, Lucy Xiaoyang Shi, Laura Smith, Jost Tobias Springenberg, Kyle Stachowicz, James Tanner, Quan Vuong, Homer Walke, Anna Walling, Haohuan Wang, Lili Yu, and Ury Zhilinsky. $\pi_{0.5}$: a vision-language-action model with open-world generalization. *arXiv preprint arXiv:2504.16054*, 2025. URL https://arxiv.org/abs/2504.16054.

[53] PsiBot. From human skill to robotic mastery: Psi-R2 and Psi-W0. PsiBot technical report, 2026. URL https://www.psibot.ai/from-human-skill-to-robotic-mastery/. Accessed 2026-07-07.

[54] Nikhila Ravi, Valentin Gabeur, Yuan-Ting Hu, Ronghang Hu, Chaitanya Ryali, Tengyu Ma, Haitham Khedr, Roman Rädle, Chloe Rolland, Laura Gustafson, Eric Mintun, Junting Pan, Kalyan Vasudev Alwala, Nicolas Carion, Chao-Yuan Wu, Ross Girshick, Piotr Dollár, and Christoph Feichtenhofer. Sam 2: Segment anything in images and videos. *arXiv preprint arXiv:2408.00714*, 2024. URL https://arxiv.org/abs/2408.00714.

[55] Rhoda AI Team. Causal video models are data-efficient robot policy learners. *Rhoda AI Blog*, 2026. URL https://www.rhoda.ai/research/direct-video-action.

[56] SAM 3D Team, Xingyu Chen, Fu-Jen Chu, Pierre Gleize, Kevin J Liang, Alexander Sax, Hao Tang, Weiyao Wang, Michelle Guo, Thibaut Hardin, Xiang Li, Aohan Lin, Jiawei Liu, Ziqi Ma, Anushka Sagar, Bowen Song, Xiaodong Wang, Jianing Yang, Bowen Zhang, Piotr Dollár, Georgia Gkioxari, Matt Feiszli, and Jitendra Malik. SAM 3D: 3dfy anything in images. *arXiv preprint arXiv:2511.16624*, 2025. URL https://arxiv.org/abs/2511.16624.

[57] William Shen, Caelan Garrett, Nishanth Kumar, Ankit Goyal, Tucker Hermans, Leslie Pack Kaelbling, Tomás Lozano-Pérez, and Fabio Ramos. Differentiable gpu-parallelized task and motion planning. In *Robotics: Science and Systems (RSS)*, 2025. URL https://arxiv.org/abs/2411.11833.

[58] Tom Silver, Rohan Chitnis, Nishanth Kumar, Willie McClinton, Tomás Lozano-Pérez, Leslie Pack Kaelbling, and Joshua B. Tenenbaum. Predicate invention for bilevel planning. In *AAAI Conference on Artificial Intelligence (AAAI)*, 2023. URL https://ojs.aaai.org/index.php/AAAI/article/view/26429.

[59] Ishika Singh, Valts Blukis, Arsalan Mousavian, Ankit Goyal, Danfei Xu, Jonathan Tremblay, Dieter Fox, Jesse Thomason, and Animesh Garg. Progprompt: Generating situated robot task plans using large language models. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2023. URL https://arxiv.org/abs/2209.11302.

[60] Siddharth Srivastava, Eugene Fang, Lorenzo Riano, Rohan Chitnis, Stuart Russell, and Pieter Abbeel. Combined task and motion planning through an extensible planner-independent interface layer. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2014. URL https://people.eecs.berkeley.edu/~russell/papers/icra14-planrob.pdf.

[61] Balakumar Sundaralingam, Siva Kumar Sastry Hari, Adam Fishman, Caelan Reed Garrett, Karl Van Wyk, Valts Blukis, Alexander Millane, Helen Oleynikova, Ankur Handa, Fabio Ramos, Nathan D. Ratliff, and Dieter Fox. curobo: Parallelized collision-free robot motion generation. In *IEEE International Conference*

on Robotics and Automation (ICRA), 2023. URL https://ieeexplore.ieee.org/document/10160765/.

[62] Martin Sundermeyer, Arsalan Mousavian, Rudolph Triebel, and Dieter Fox. Contact-graspnet: Efficient 6-dof grasp generation in cluttered scenes. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2021. URL https://arxiv.org/abs/2103.14127.

[63] Bin Tan, Changjiang Sun, Xiage Qin, Hanat Adai, Zelin Fu, Tianxiang Zhou, Han Zhang, Yinghao Xu, Xing Zhu, Yujun Shen, and Nan Xue. Masked depth modeling for spatial perception. *arXiv preprint arXiv:2601.17895*, 2026. URL https://arxiv.org/abs/2601.17895.

[64] Fabio Tosi, Luca Bartolomei, and Matteo Poggi. A survey on deep stereo matching in the twenties. *International Journal of Computer Vision (IJCV)*, 2025. URL https://link.springer.com/article/10.1007/s11263-024-02331-0.

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

[66] Marc Toussaint, Kelsey Allen, Kevin Smith, and Joshua Tenenbaum. Differentiable physics and stable modes for tool-use and manipulation planning. In *Robotics: Science and Systems (RSS)*, 2018. URL https://www.roboticsproceedings.org/rss14/p44.html.

[67] Jie Wang, Matthew Leonard, Kostas Daniilidis, Dinesh Jayaraman, and Edward S. Hu. Evaluating π_0 in the wild: Strengths, problems, and the future of generalist robot policies, 2025. URL https://penn-pal-lab.github.io/Pi0-Experiment-in-the-Wild/.

[68] Shu Wang, Muzhi Han, Ziyuan Jiao, Zeyu Zhang, Ying Nian Wu, Song-Chun Zhu, and Hangxin 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. URL https://arxiv.org/abs/2403.11552.

[69] Bowen Wen, Shaurya Dewan, and Stan Birchfield. Fast-foundationstereo: Real-time zero-shot stereo matching. *arXiv preprint arXiv:2512.11130*, 2025. URL https://arxiv.org/abs/2512.11130.

[70] Bowen Wen, Matthew Trepte, Joseph Aribido, Jan Kautz, Orazio Gallo, and Stan Birchfield. Foundationstereo: Zero-shot stereo matching. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025. URL https://arxiv.org/abs/2501.09898.

[71] Zhutian Yang, Caelan Reed Garrett, Dieter Fox, Tomás Lozano-Pérez, and Leslie Pack Kaelbling. Guiding long-horizon task and motion planning with vision language models. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2025. URL https://arxiv.org/abs/2410.02193.

[72] Seonghyeon Ye, Yunhao Ge, et al. World action models are zero-shot policies. *arXiv preprint arXiv:2602.15922*, 2026. URL https://arxiv.org/abs/2602.15922.

[73] Ryan Yu, Pushi Zhang, Starrick Liu, Brae Liu, Miracle Kang, Shalfun Li, Lights Shi, Ellie Ma, Ping Yang, Chris Pan, Jerry Chen, Dongxiu Liu, Rain Sun, Miles Guo, Byron Zhang, Hugo Zhou, Zach Xu, Vincent Chen, Harrison Huang, James Wang, Dance Kuzi, Andy Zhai, Hang Su, Roy Gan, Lucy Liang, Hao Wang, and Qian Wang. Wall-oss-0.5 technical report. *arXiv preprint arXiv:2605.30877*, 2026. URL https://arxiv.org/abs/2605.30877.

[74] Wentao Yuan, Adithyavairavan Murali, Arsalan Mousavian, and Dieter Fox. M2t2: Multi-task masked transformer for object-centric pick and place. In *Conference on Robot Learning (CoRL)*, 2023. URL https://proceedings.mlr.press/v229/yuan23a.html.

[75] Andrii Zadaianchuk, Leonardo Barcellona, Lennard Schuenemann, Christian Gumbsch, Zehao Wang, Muhammad Zubair Irshad, Fabien Despinoy, Rahaf Aljundi, Stratis Gavves, and Sergey Zakharov. Reconstruction by generation: 3d multi-object scene reconstruction from sparse observations. *arXiv preprint arXiv:2604.27106*, 2026. URL https://arxiv.org/abs/2604.27106.

APPENDIX

A. cuTAMP Extensions

We made several extensions to cuTAMP [57] 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 5 mm 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. 7: **Object Segmentation.** SAM-2 generates eight pixel-level segmentation masks from the bounding boxes in Fig. 3c.

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 1), which is automatically stopped upon robot execution for TiPToP, and manually stopped for $\pi_{0.5}$-DROID.

Failure Analysis. Our systematic failure analysis from §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 1). 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 125 Hz 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)`).

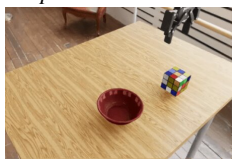
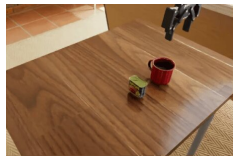
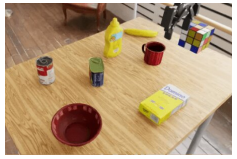
Planning. We define a new `Wipe` cuTAMP operator with preconditions that the robot is holding an eraser and the target is a surface, and an effect that marks the surface as cleaned. The task planner automatically sequences pick then wipe to satisfy an `IsCleaned` goal. During motion solving, `Wipe` hands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

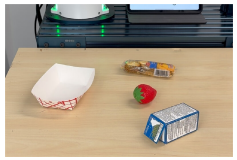
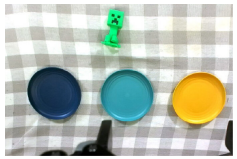
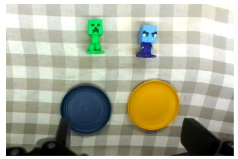
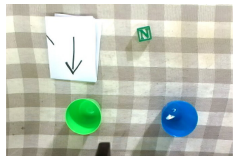
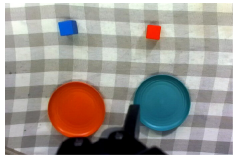
F. MolmoSpaces Integration

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, the 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 §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.