TiPToP: A Modular Open-Vocabulary Planning System for Robotic Manipulation

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

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

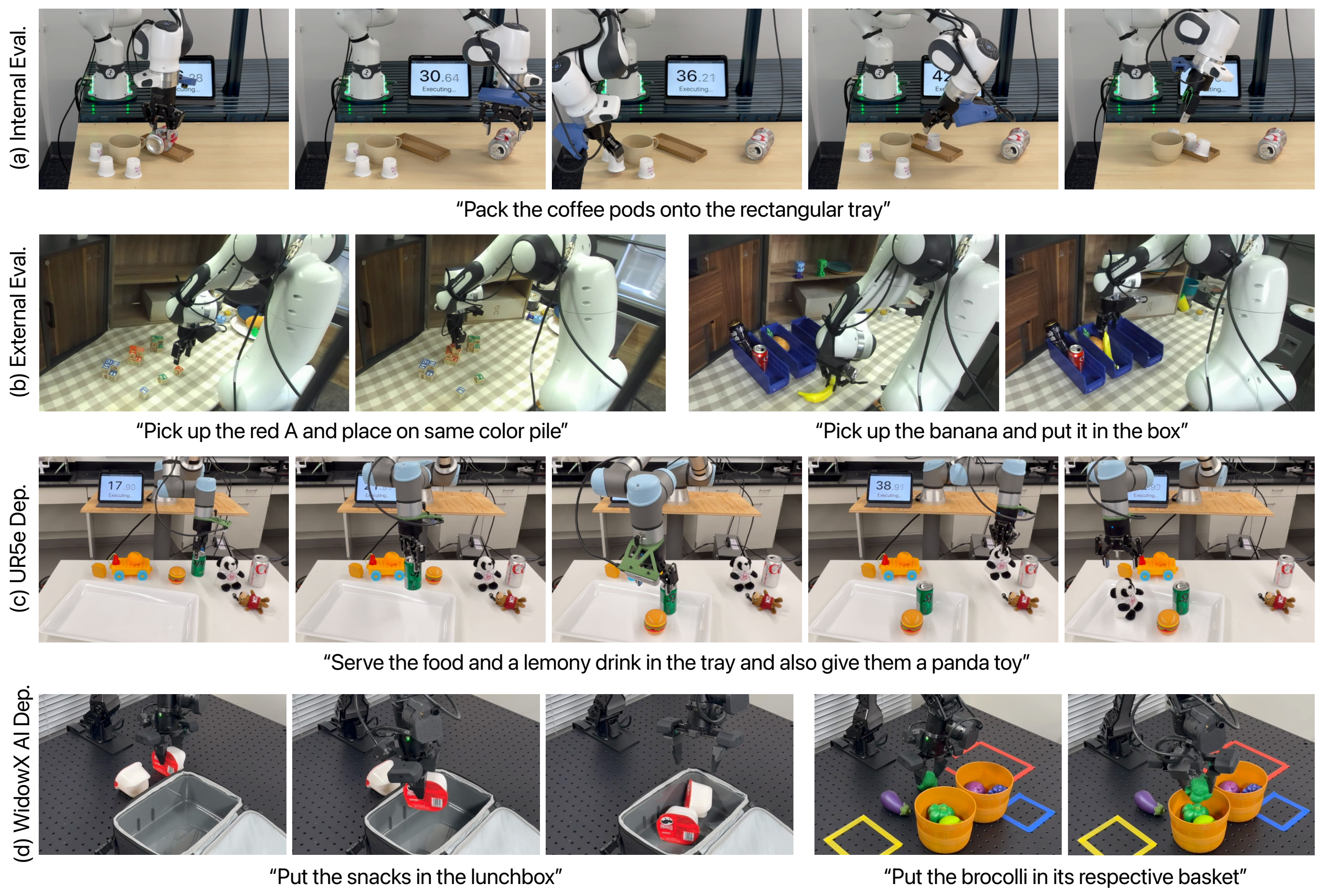

(a) Internal Eval.
"Pack the coffee pods onto the rectangular tray"
(b) External Eval.
"Pick up the red A and place on same color pile"
"Pick up the banana and put it in the box"
(c) UR5e Dep.
"Serve the food and a lemony drink in the tray and also give them a panda toy"
(d) WidowX AI Dep.
"Put the snacks in the lunchbox"
"Put the brocolli in its respective basket"

Fig. 1: **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, an extensible modular system that combines pretrained vision foundation models with an existing Task and Motion Planner (TAMP) to solve multi-step manipulation tasks directly from input RGB images and natural-language instructions. Our system aims to be simple and easy-to-use: it can be installed and run on a standard DROID setup in under one hour and adapted to new embodiments with minimal effort. We evaluate TiPToP — which requires zero robot data — over 28 tabletop manipulation tasks in simulation and the real world and find it matches or outperforms $\pi_{0.5}$-DROID, a vision-language-action (VLA) model fine-tuned on 350 hours of embodiment-specific demonstrations. TiPToP's modular architecture enables us to analyze the system's failure modes at the component level. We analyze results from an evaluation of 173 trials and identify directions for improvement. We release TiPToP open-source to further research on modular manipulation systems and tighter integration between learning and planning. Project website and code: tiptop-robot.github.io**

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

I. Introduction

A longstanding goal of robotics research has been to build a manipulation system that "just works" out-of-the-box: 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.

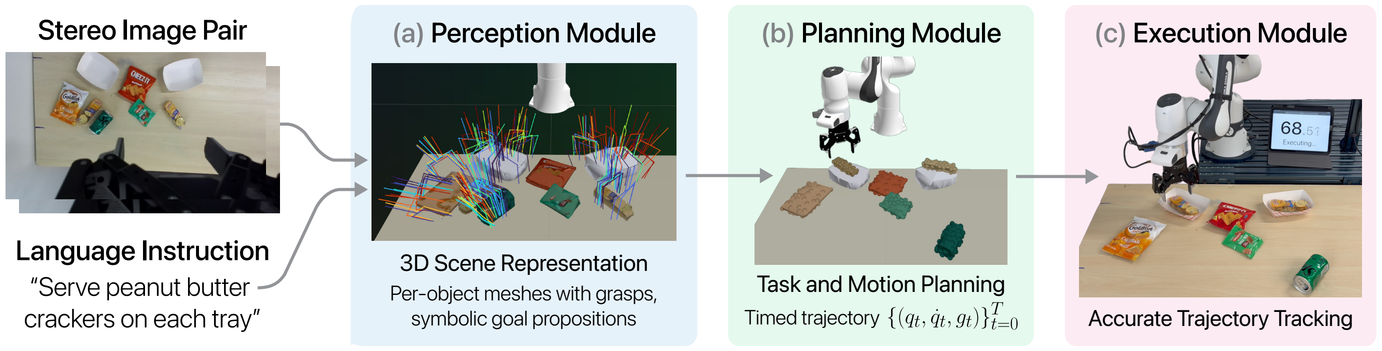

Stereo Image Pair
Language Instruction
"Serve peanut butter crackers on each tray"
(a) Perception Module
3D Scene Representation
Per-object meshes with grasps, symbolic goal propositions
(b) Planning Module
Task and Motion Planning
Timed trajectory $\{(q_t, \dot{q}_t, g_t)\}_{t=0}^{T}$
(c) Execution Module
Accurate Trajectory Tracking

Fig. 2: **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) to find feasible manipulation plans. (c) The execution module tracks the planned trajectory using a joint impedance controller.

Vision-Language-Action (VLA) models such as $\pi_{0.5}$ [45] and OpenVLA [30] offer an appealing input-output specification — natural language and camera images in, robot actions out — but require substantial training data and lack reliable cross-embodiment generality. On the other hand, Task and Motion Planning (TAMP) [27, 21, 12] offers a structured framework for multi-step manipulation, jointly reasoning over discrete action sequences and continuous geometric constraints. There has been substantial research demonstrating TAMP in the real world [51, 20, 12, 48], but these systems lack generality and have generally relied on implementations that are tightly coupled to specific hardware, perception, and control stacks, making them difficult to access and build upon.

We introduce **TiPToP** (TiPToP is a Planner That just works on Pixels), a modular planning-based manipulation system built on recent advances in vision and language foundation models, learned grasp prediction, and GPU-accelerated TAMP. TiPToP accepts the same inputs as VLAs — natural language and camera images — but requires no data collection or training, instead leveraging pretrained foundation models for perception and GPU-accelerated TAMP for planning at inference time.

Our primary contribution is a complete manipulation system that can be installed and deployed on supported[1] robot embodiments in under one hour with only camera calibration. Its modular architecture allows individual components to be improved or replaced independently as better foundation models become available, and when failures occur, they can be traced to specific modules for targeted debugging.

To validate TiPToP's accessibility and generalizability, we sent our code to an external evaluation team not involved in its development. This team independently deployed TiPToP on the DROID hardware platform [29] and conducted a systematic comparison against $\pi_{0.5}$-DROID [7], a state-of-the-art VLA fine-tuned on 350 hours of embodiment-specific demonstrations. We additionally performed similar comparisons on our own hardware and in simulation. Results show that TiPToP achieves comparable or better success rates across diverse tasks, and we find that the two systems fail in complementary ways. We provide a detailed failure analysis and use it to identify concrete directions for improving individual modules. We further demonstrate deployment on UR5e and Trossen WidowX AI robot arms to validate ease of deployment on new embodiments.

We release TiPToP as open-source software, with support for both real-world deployments and development in simulation [41], to provide an accessible foundation for future research into planning-based manipulation and to facilitate direct comparison and integration with end-to-end learned approaches.

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 [55, 61, 24] predict dense depth maps from RGB image pairs. Foundation models for grasp generation [39, 53, 63] predict 6-DoF grasp poses from point clouds. SAM [31] and SAM-2 [46] provide promptable segmentation from bounding boxes or points, enabling precise object boundary delineation. VLMs [23, 15, 4, 43] 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 [1] similarly 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

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

on diverse robot data. Many VLAs leverage VLM backbones to enable language-conditioned control [9, 30, 6, 22] and demonstrate that transformer-based policies trained on large datasets (e.g., the Open X-Embodiment dataset [42]) can generalize across tasks and objects.

π_0 [7] introduced a flow-matching architecture, which was subsequently extended in $\pi_{0.5}$ [45] 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. 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's modular architecture requires no embodiment-specific training and explicitly reasons about geometry and task structure.

Task and Motion Planning. TAMP algorithms jointly solve discrete task planning and continuous motion planning problems [27, 21], enabling the satisfaction of constraints involving both high-level action sequencing and low-level geometric feasibility. Common approaches use sampling-based methods [19, 20] or optimization [56, 57] 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. [12], who integrated learned perception modules for property and affordance estimation with PDDLStream [20] to enable long-horizon manipulation of unknown objects in the real world. Our system differs in several critical ways: first, we use cuTAMP [48], 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 [51, 11]; 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 [17], applied to the Shakey robot [40], 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 [2, 25], generate robot programs [35, 50], construct 3D value maps [26], or combine VLMs with manipulation skills [37]. Other work integrates learning with TAMP: LLM^3 [59] uses LLMs to propose task plans with motion failure reasoning, PRoC3S [13] combines LLMs with continuous constraint satisfaction, VLM-TAMP [62] feeds VLM-generated subgoals to a task and motion planner, and OWL-TAMP [33] uses a VLM to infer constraints passed to a TAMP system. Compared to LLM-based systems that sequence discrete skills, TiPToP jointly optimizes discrete task plans and continuous collision-free trajectories, which is important when constraints on continuous parameters impact the feasibility of skill sequences.

III. Problem Setting

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, $a_t = \pi(\mathbf{o}_t, q_t \mid \mathcal{L})$. The two systems we compare instantiate this specification very differently:

A. $\pi_{0.5}$-DROID

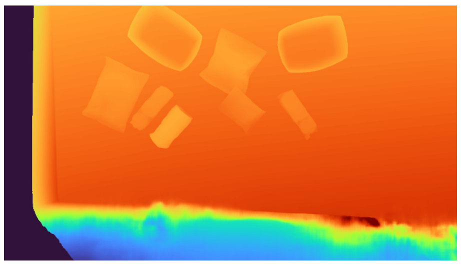
$\pi_{0.5}$-DROID runs at 15 Hz. At every timestep it observes $\mathbf{o}_t = (I_t^{\text{wrist}}, I_t^{\text{ext}})$, monocular RGB images from the wrist camera and external camera, respectively, along with the current joint and gripper positions (q_t, g_t). It outputs chunks of 15 actions $\{a_t\}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

B. TiPToP

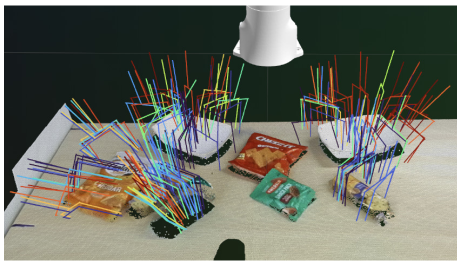
TiPToP implements the policy using a *planner*: it observes the scene *once* at $t{=}0$ from a *calibrated* wrist camera at a capture pose, which we assume provides a good view of the workspace. The observation $\mathbf{o}_0 = (I_0^{\text{left}}, I_0^{\text{right}})$ is a stereo RGB image pair with known intrinsics K, camera-to-end-effector extrinsics $T_{\text{cam}}^{\text{ee}}$, and stereo baseline b. From this single observation, it 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 [48] 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.

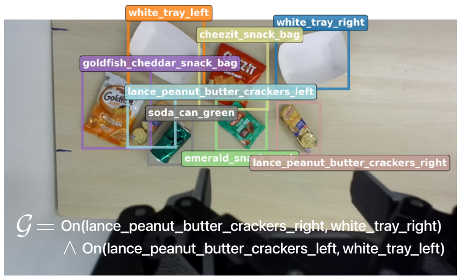
Illustrative Example. We illustrate TiPToP in the DROID setup (Franka FR3 with a ZED Mini stereo camera mounted on the wrist) in the following scenario: the robot is given the instruction *"serve peanut butter crackers on each tray"* and the scene in Figure 2. This task requires identifying peanut butter 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


white_tray_left
white_tray_right
cheezit_snack_bag
goldfish_cheddar_snack_bag
lance_peanut_butter_crackers_left
soda_can_green
emerald_sna
lance_peanut_butter_crackers_right
$\mathcal{G} =$ On(lance_peanut_butter_crackers_right, white_tray_right)
$\wedge$ On(lance_peanut_butter_crackers_left, white_tray_left)

(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 (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 [61], 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 [63] 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 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 [39], 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 [16], 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 [22], 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. We currently support one predicate, $\texttt{On}(a, b)$, though we demonstrate defining additional predicates for new skills in §VII-D. 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 3c).

Object Segmentation. For each detected bounding box, we use SAM-2 [46] 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 [18] 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 use the segmentation mask of each detected object to extract the corresponding points from $\mathbf{p}^{\text{world}}$, project them downward along the z-axis to the object's lowest observed point, and compute the convex hull to form a watertight mesh. We project to each object's own lowest point rather than to the table, as objects may rest on each other. Since we observe from a single viewpoint, the convex hull typically over-approximates object geometry, which is preferable for collision checking.

Grasp-to-Object Assignment. Each grasp predicted by M2T2 is assigned to the nearest object by querying its contact point against a KDTree [5] 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 [48], 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 (§VII-D). 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 [38] 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 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 [52], 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.

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. We implemented our own joint-space impedance controller for Franka arms (Appendix B), because existing open-source controllers, including DROID's default Polymetis controller, were unable to track timed trajectories sufficiently precisely.

TiPToP does not monitor execution or replan based on execution-time observations (i.e., it is open-loop with respect to visual observations). This succeeds when the world is static and trajectories are tracked accurately, but fails when objects move unexpectedly or grasps slip.

VII. Experiments

Our experiments are designed to answer the following questions:

- **Q1.** To what extent is TiPToP capable of satisfying manipulation tasks specified with open-ended language expressions and involving everyday objects, especially when compared to a state-of-the-art VLA?
- **Q2.** How does TiPToP's time taken for task success compare to a state-of-the-art VLA?
- **Q3.** What are the primary failure modes of TiPToP?

A. Experimental Setup

We tested TiPToP in 3 different settings: (i) in a simulation environment built using the IsaacSim [41] simulator, (ii) on a real-world DROID hardware setup, (iii) on a separate DROID hardware setup operated by an external evaluation team.

Evaluation Protocol. Following an "in-the-wild" evaluation protocol inspired by Wang et al. [58], we chose natural language goal expressions and objects that corresponded to tasks that both TiPToP and $\pi_{0.5}$-DROID seemed capable of. Both systems received the same natural-language instruction and started from the same robot configuration. We ran 5 tasks

Scene	TiPToP SR	TiPToP TP	$\pi_{0.5}$-DROID SR	$\pi_{0.5}$-DROID 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 evaluation scenes.** SR = Success Rate, TP = Task Progress. Best results are **bolded**. The row at the bottom of each category shows the aggregate SR (sum) and TP (mean); **Overall** aggregates across all scenes. Sim scenes use 10 trials; all others are real-world evaluations with 5 trials. [†]Evaluated by system designers; unmarked scenes evaluated by the external evaluation team.

in simulation, 8 tasks on the DROID setup used by TiPToP's developers (top row Figure 1), and 15 tasks on the external DROID setup (middle row of Figure 1). We ran 5 trials per task in the real world, and 10 trials per task in simulation. Our key measure was binary success rate averaged over trials, though we also report task progress, which is a more continuous measure that was defined using subgoals on a per-task basis (see Appendix C for details).

B. Results and Analysis

Table I shows the results of these experiments (Q1). We grouped the tasks from the evaluation into 4 separate categories for the purposes of interpretation and analysis. The *simple* tasks are one-step pick-and-place tasks with no distractor objects. The *distractor* tasks are rearrangement tasks that feature a number of distractors and require identifying and manipulating only the relevant object(s). The *semantic* tasks feature complex natural language goal expressions that require semantic reasoning about the scene to identify the correct object(s) (e.g. "Pick up the largest toy and place onto the purple plate"). Finally, the *multi-step* tasks require executing many actions in sequence in settings where some physical reasoning is required to accomplish the task (e.g. constrained packing, moving an obstacle out of the way to reach a particular object, or moving an obstacle to make room for a placement).

TiPToP matches $\pi_{0.5}$-DROID on simple tasks and outperforms on the others. On the five simple pick-and-place scenes, TiPToP and $\pi_{0.5}$-DROID perform comparably: each system achieves a higher success rate on two scenes, with one tied. As task complexity increases, however, a consistent performance gap emerges. On distractor tasks, TiPToP achieves a higher success rate on six of eight scenes, including three scenes where it reaches 100% while $\pi_{0.5}$-DROID scores 30% or below. On semantic tasks, this gap is more pronounced: TiPToP achieves a higher success rate on seven of eight scenes, and $\pi_{0.5}$-DROID scores 0/5 on four of them. We attribute this performance to TiPToP's use of a large VLM to translate visual observations and natural language instructions into a symbolic goal $\mathcal{G}$. This explicit grounding step enables TiPToP to correctly identify task-relevant objects amid distractors and to interpret complex referring expressions (e.g., "largest toy," "matching plate," "sort by color") that $\pi_{0.5}$-DROID has no mechanism to reason about.

On multi-step tasks, TiPToP achieves a higher success rate on six of seven scenes, with the largest difference in the simulated color cubes scene (8/10 vs. 0/10). Here, the advantage stems from TAMP: cuTAMP decomposes multi-step goals into a sequence of feasible pick-and-place actions with collision-free motion plans, whereas $\pi_{0.5}$-DROID must implicitly discover multi-step structure from the language command.

Task progress scores tell a complementary story: even in scenes where TiPToP does not fully succeed, it often completes most subgoals (e.g., 70% task progress at 4/10 success rate on cube → bowl), indicating that failures tend to be isolated to a single step rather than wholesale inability to make progress. $\pi_{0.5}$-DROID also achieves non-trivial task progress on many scenes where it ultimately fails, but TiPToP consistently achieves higher task progress in both semantic and multi-step categories.

Failure cases. The scenes where TiPToP performs relatively poorly, and where $\pi_{0.5}$-DROID outperforms TiPToP reveal

	$\pi_{0.5}$-**DROID**	**TiPToP**	
Scene	Time (s)	Time (s)	Plan (s)
Cube $\rightarrow$ bowl (sim)	27.4	**17.9**	9.7
Can $\rightarrow$ mug (sim)	41.0	**18.6**	9.2
Crackers $\rightarrow$ tray (simple)†	32.2	**14.9**	7.0
Crackers $\rightarrow$ tray (medium)†	45.2	**14.9**	7.3
Pack pods $\rightarrow$ tray†	53.4	**47.0**	20.5
Aleve bottle $\rightarrow$ 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'). † indicates tasks evaluated in the real world by the system designers.

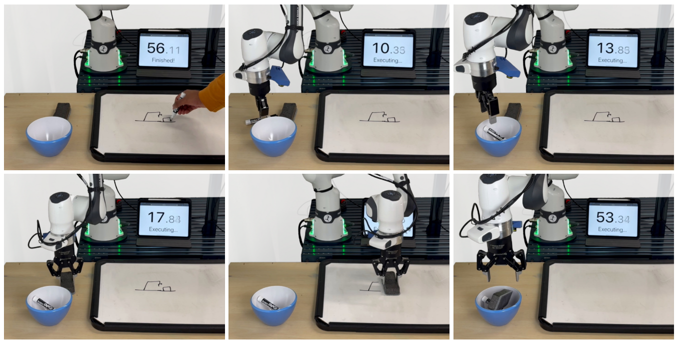


Fig. 4: **Wiping.** We demonstrated that TiPToP can be straightforwardly extended to perform wiping in addition to pick-and-place. Task instruction: "erase the whiteboard and put everything into the bowl".

systematic limitations of TiPToP's current implementation. First, TiPToP's perception module approximates object meshes using convex hulls, which fails for objects with concave geometry. Bananas are the clearest example: in both banana $\rightarrow$ bin (sim) and banana $\rightarrow$ matching plate, the convex hull poorly represents the true shape, leading to grasp and placement failures. This accounts for two of the five scenes where $\pi_{0.5}$-DROID achieves a higher success rate. Second, small objects such as cashews and AirPods are difficult to grasp reliably in a single attempt. $\pi_{0.5}$-DROID benefits here from its closed-loop, reactive policy: when a grasp fails, it can re-attempt on the next control step, whereas TiPToP commits to a single open-loop trajectory and has no mechanism to retry. Third, objects can slip after an initially successful grasp. In the red cubes scene, TiPToP frequently grasped the correct cube but lost it during transport, resulting in only 1/5 successes despite 50% task progress. $\pi_{0.5}$-DROID's continuous visual feedback allows it to adjust its grip or re-grasp, achieving 5/5 on this task. These failure modes point to a common theme: TiPToP's lack of execution-time reactivity is a significant limitation.

Execution time. Table II compares average time-to-success for both systems (Q2). TiPToP is faster than $\pi_{0.5}$-DROID in six of seven scenes, often substantially so. On single-step real-world tasks, TiPToP completes execution in under 15 seconds, roughly half the time of $\pi_{0.5}$-DROID. The advantage is even larger on scenes where $\pi_{0.5}$-DROID requires multiple grasp attempts (e.g., can $\rightarrow$ mug: 19.1s vs. 56.5s). This speed advantage stems from TiPToP's architecture: it plans a single, time-optimal trajectory upfront and executes it open-loop, whereas $\pi_{0.5}$-DROID runs a reactive control loop that may need several cycles of approaching, grasping, and recovering from failed attempts. In fact, we observed qualitatively that $\pi_{0.5}$-DROID spends a significant amount of time idling and seemingly not making any task progress. On multi-step tasks, TiPToP remains faster (46.8s vs. 53.4s on pack pods; 31.2s vs. 35.5s on Aleve bottle), though the margin narrows as more of the total time is spent on execution rather than planning. The one scene where $\pi_{0.5}$-DROID is faster (banana $\rightarrow$ bin, 16.4s vs. 20.5s) is also one where TiPToP's convex hull approximation causes it to require additional planning time to find collision-free grasps and placements, leading to slower trajectories even when they succeed.

C. Cross-Embodiment Generalization

To validate TiPToP's modularity with respect to embodiments, we deployed the system on a UR5e arm with a wrist-mounted Intel Realsense D435 camera (Figure 1c). Adapting to the new embodiment required providing the robot URDF, generating collision spheres for the robot, 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). TiPToP was also deployed on a Trossen WidowX AI arm with a wrist-mounted RealSense D405 camera in collaboration with an external researcher (Figure 1d).

D. Extending Beyond Pick-and-Place

TiPToP's modularity makes it straightforward to extend beyond pick-and-place. To demonstrate this, we added a *whiteboard wiping* primitive that enables the robot to wipe writing off a surface given an eraser (as depicted in Figure 4). This 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` TAMP operator in cuTAMP 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.

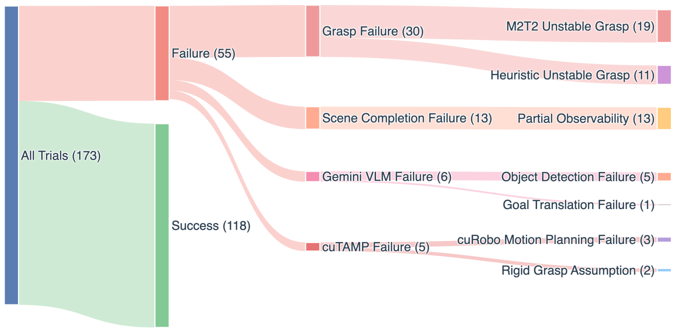


Fig. 5: **Failure Analysis.** Sankey diagram showing outcomes of 173 trials. The most common failure modes are grasping failures (missed or unstable grasps), followed by scene completion errors, VLM detection errors, then cuTAMP failures.

The entire extension was implemented in under a day without modifying any existing perception, planning, or execution code outside of the additions described above.

E. Failure Analysis

A key advantage of TiPToP's modular architecture is the ability to 'debug' the system by tracing the root cause of failures to particular parts of the system. To understand the relative frequency of failures in different parts of the system (Q3), we ran an additional 173 trials of a variety of rearrangement tasks on the real-world DROID setup in the lab of the TiPToP developers. For each failure, we traced the root cause to a particular part of the system. The results are displayed in Figure 5.

Grasping failures (30/55 failures) are the most common failure mode. These occur 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 failures) result from incorrect mesh approximations that cause collisions during execution. These typically occur when convex hull completion overapproximates concave objects (e.g. overapproximating the mesh of a banana into a large oval) or when partial observability leads to underapproximation.

VLM errors (6/55 failures) occur when Gemini fails to detect objects or produces incorrect bounding boxes.

cuTAMP failures (5/55 failures) occur when the planner cannot find feasible plans within the time budget, typically due to motion planning failures in cluttered scenes.

VIII. Discussion

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 5) 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 [12, 34, 44, 8].

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, convex hull 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. Learned shape completion methods such as SAM-3D [47] could replace convex hull approximation with more accurate meshes. Advances in depth estimation [60, 54] 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, 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. This suggests that learned policies such as VLAs could serve as reactive skill primitives within TiPToP, both improving robustness for existing skills and enabling new ones that are difficult to hand-engineer (e.g., folding, cable manipulation). Integrating such skills requires specifying their abstract preconditions and effects so the planner can reason about when to invoke them. These abstract models could be manually engineered or learned from interaction data [49, 32, 36, 3].

Belief-space planning. Extending cuTAMP to operate in belief space would enable reasoning about uncertainty in object poses, grasp outcomes, and partially observable state [28, 14, 10]. 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.

IX. 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. 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. Importantly, each component of TiPToP can be independently upgraded as better depth estimators, grasp predictors, VLMs, and TAMP or motion planners become available. At the same time, the complementary failure profiles of TiPToP and $\pi_{0.5}$-DROID suggest that integrating end-to-end VLA models with our framework could yield systems that combine the structured reasoning of planning with the robustness and flexibility of closed-loop visuomotor control. We hope that our open-source system, as well as our empirical findings, support future research towards 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 Ryan Lindeborg for deploying TiPToP on his Trossen WidowX AI and for providing installation and debugging feedback. 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 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.

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.

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 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] 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.

[3] 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.

[4] 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.

[5] 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.

[6] 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.

[7] 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.

[8] 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.

[9] 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.

[10] 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.

[11] 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.

[12] 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.

[13] 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.

[14] 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.

[15] 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.

[16] 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.

[17] 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.

[18] 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.

[19] 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.

[20] 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.

[21] 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.

[22] 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.

[23] 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.

[24] 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.

[25] 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.

[26] 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.

[27] 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. URL https://ieeexplore.ieee.org/document/

5980391.

[28] 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.

[29] 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.

[30] 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.

[31] 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.

[32] 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.

[33] 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.

[34] 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.

[35] 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.

[36] 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.

[37] 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.

[38] 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.

[39] 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.

[40] 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.

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

[42] 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.

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

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

[45] 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.

[46] 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.

[47] 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.

[48] 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.

[49] 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.

[50] 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.

[51] 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.

[52] 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/.

[53] 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.

[54] 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.

[55] 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.

[56] 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.

[57] 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.

[58] 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/.

[59] 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.

[60] 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.

[61] 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.

[62] 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.

[63] 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.

Appendix

A. *cuTAMP Extensions*

We made several extensions to cuTAMP [48] 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 position error). We will open-source our controller implementation for the Franka FR3 and Panda robots upon acceptance.

C. *Additional Experiment Details*

Table III 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.

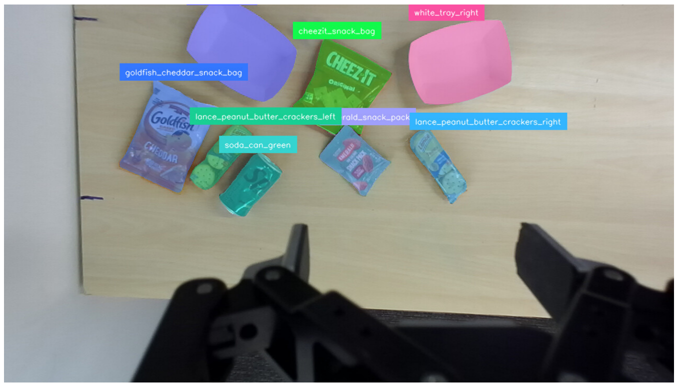

white_tray_right
cheezit_snack_bag
goldfish_cheddar_snack_bag
lance_peanut_butter_crackers_left
rald_snack_pack
lance_peanut_butter_crackers_right
soda_can_green

Fig. 6: **Object Segmentation.** SAM-2 generates eight pixel-level segmentation masks from the bounding boxes in Fig. 3c.

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 III), $\pi_{0.5}$-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 III), $\pi_{0.5}$-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 Section VII-E 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:

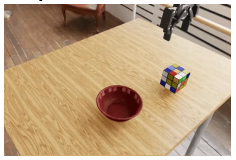
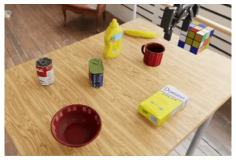
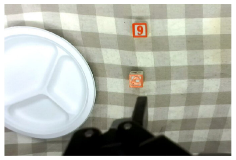
- The robot URDF.
- Collision spheres for the robot geometry, automatically generated using tools like Ballpark or Foam.
- A cuRobo configuration file, following this guide from the cuRobo developers.
- Code changes in cuTAMP to load the new configuration files.
- Code changes in TiPToP to interface 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.

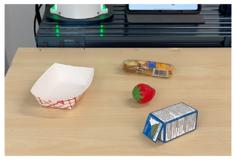
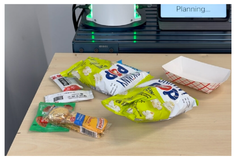
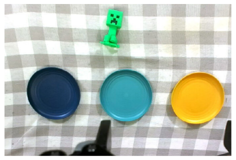
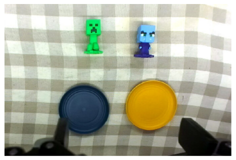
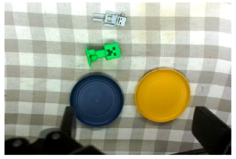
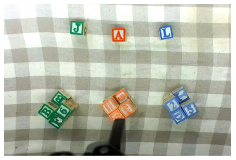
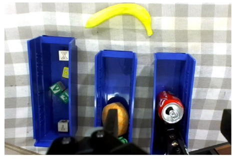
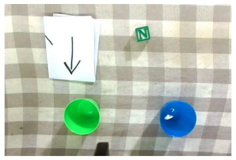
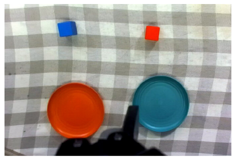
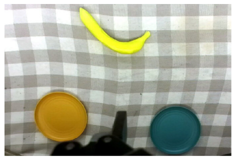
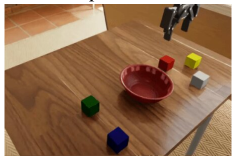
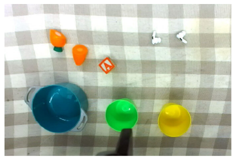
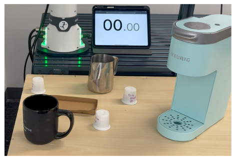
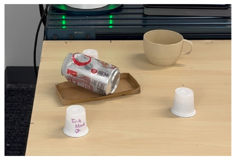
FoundationStereo with a RealSense. For stereo input to FoundationStereo, we used the RealSense’s left and right infrared (IR) sensors. This qualitatively resulted in noisier depth estimates than the DROID setup, which uses RGB stereo pairs from the ZED Mini, particularly on transparent, specular, and 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 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.

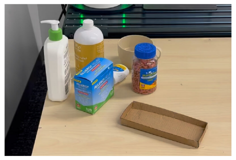
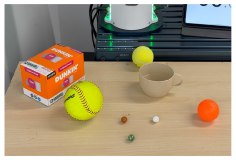
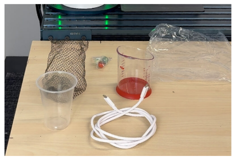
TABLE III: **Evaluation scene details.** Each scene shows an image of the task, its identifier (as referenced in Table I), language prompt, and task progress metric. Scenes are grouped by category: Simple, Distractor, Semantic, and Multi-step. † indicates tasks evaluated by the system designers. Unmarked 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 $\rightarrow$ bowl (sim) “put the cube in the bowl”	25% approach cube, 50% grasp, 75% approach bowl with cube, 100% place		Can $\rightarrow$ mug (sim) “put the can in the mug”	25% approach can, 50% grasp, 75% approach mug with can, 100% place
	Banana $\rightarrow$ bin (sim) “put banana in the bin”	25% approach banana, 50% grasp, 75% approach bin with banana, 100% place		Marker $\rightarrow$ tray “put the marker in the tray”	+25% touch marker, +25% grasp, +25% touch tray, +25% place
	Crackers $\rightarrow$ tray[†] “place the crackers onto the tray”	50% grasp crackers, 100% place			
Distractor					
	Meat can $\rightarrow$ 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 $\rightarrow$ plate “put all of the coffee capsules onto the white plate”	+50% per capsule placed, −20% per distractor
	Turkish figs $\rightarrow$ plate “put the turkish figs onto the white plate”	+50% per fig placed, −20% per cashew		Cashews $\rightarrow$ plate “put the roasted cashews onto the white plate”	+50% per cashew placed, −20% per fig
	Red cubes $\rightarrow$ plate “put the red cubes onto the white plate”	+50% per cube placed, −20% if distractor placed		Fish $\rightarrow$ 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 $\rightarrow$ tray (med.)† "place the crackers onto the tray"	+50% pick crackers, +50% place on the tray (no penalty for distractor)		PB crackers $\rightarrow$ tray (hard)† "place the peanut butter crackers onto the tray"	+50% pick crackers, +50% place on the tray (no penalty for distractor)
Semantic					
	Toy $\rightarrow$ 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 $\rightarrow$ plate "pick up the creeper and place onto the purple plate"	+50% pick creeper toy, +50% place onto purple plate
	Largest toy $\rightarrow$ 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 $\rightarrow$ 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 $\rightarrow$ 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 $\rightarrow$ 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 $\rightarrow$ matching plate "place banana into plate has similar color"	+50% pick banana, +50% place into orange plate
Multi-step					
	Color cubes $\rightarrow$ 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 $\rightarrow$ cup "place airpods into the yellow cup"	+25% per AirPods picked, +25% per place, −20% distractor
	Pack pods $\rightarrow$ 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 $\rightarrow$ 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			

E. 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.
```