2026 IEEE International Conference on Robotics and Automation (ICRA 2026)
June 1-5, 2026. Vienna, Austria

Task and Skill Planning: Hierarchical Robot Planning with Black-Box Skills

Benned Hedegaard[1†*], Yichen Wei[1†], Ziyi Yang[1], Ahmed Jaafar[1], Stefanie Tellex[1], George Konidaris[1], and Naman Shah[1,2§]

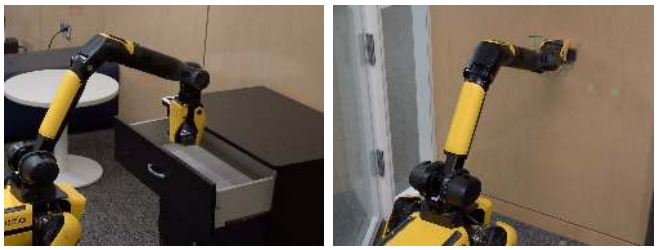
***Abstract*— Task and motion planning (TAMP) is a well-established approach for solving long-horizon robot planning problems. Although TAMP methods have historically assumed that each task-level robot action, or skill, can be reduced to kinematic motion planning, recent work has explored integrating closed-loop controllers and learned skills into TAMP-style systems. Our approach integrates pre-existing, heterogeneous robot skills—including learned, force-controlled, and black-box policies—into a hierarchical planner while preserving the object-centric failure reasoning of typical TAMP solvers. We leverage Composable Interaction Primitives (CIPs) to synthesize head and tail motion plans bridging consecutive skills, facilitating both planning-time refinement and execution-time adjustment. We validate our *Task and Skill Planning (TASP)* approach through real-world experiments on a bimanual manipulator and a mobile manipulator, demonstrating that CIPs enable diverse robots to combine heterogeneous skills to solve complex, long-horizon tasks, including multi-room mobile manipulation problems with non-monotonic task structure.**

I. Introduction

Task and motion planning (TAMP) combines discrete task-level reasoning with continuous motion planning to solve long-horizon robot planning problems [29, 11]. Recent work has extended TAMP-style methods beyond purely kinematic actions by incorporating learned skills [36, 31, 23], closed-loop controllers [28, 7], and uncertainty-aware execution [7, 10]. However, these methods typically assume a particular policy structure or controller class for the integrated skills. Therefore, the remaining challenge is not whether skills can be used within hierarchical planning, but how to plan effectively when a robot is equipped with a heterogeneous, pre-existing inventory of general-purpose skills.

In practice, robot skills often span multiple implementation classes, including learned policies, force-controlled behaviors, built-in robot skills, and trajectory playback routines. Skill controllers may be closed-loop, stochastic, or otherwise opaque to the planner, and may not permit a uniform internal model or parameterization. Nevertheless, a planner must reason about when such skills can be executed and how their effects may enable or obstruct future actions. This is especially important for non-downward-refinable actions [24], where resolving failure may require reasoning about object obstructions and counterfactual scene modifications.

[1]Department of Computer Science, Brown University, Providence, RI. [2]Allen Institute for Artificial Intelligence, Seattle, WA. [§]Work primarily completed while at Brown University.
[†]Equal contribution
[*]Corresponding author: benned_hedegaard@brown.edu



(a) `Pick(?eraser)` (b) `Erase(?board)`

(c) `Grasp(?jar)` (d) `Spread(?knife, ?bread)`

Fig. 1. A Boston Dynamics Spot and a bimanual manipulator use task and skill planning (TASP) to solve long-horizon hybrid robot planning problems. On the left, the robots execute motion-planning-based skills to pick up an eraser (Fig. 1(a)) and a jar of peanut butter (Fig. 1(c)). On the right, the robots use general-purpose skills to maintain force-controlled contact while erasing a whiteboard (Fig. 1(b)) and spreading peanut butter (Fig. 1(d)).

We address this setting by exposing planner-facing geometric structure for otherwise black-box skills. We use a *kinematic envelope* to characterize where a skill can be initiated and its nominal geometric effects, allowing us to package skills as Composable Interaction Primitives [1] that synthesize motion plans between consecutive skills. We instantiate this idea as *Task and Skill Planning (TASP)*, a hierarchical planning framework that retains motion-planning-based feasibility checking and object-centric geometric reasoning without requiring explicit internal models of skill controllers. We evaluate our approach on two real-world robots, a bimanual manipulator and a mobile manipulator, demonstrating that TASP can plan using motion-planned, learned, force-controlled, and built-in robot skills to solve long-horizon real-world tasks, including mobile manipulation problems with non-monotonic task structure.

II. Preliminaries

This paper brings together ideas from different domains of robotics, including motion planning, symbolic reasoning, task and motion planning, and skill learning. This section

outlines the foundational concepts necessary to understand the proposed methodology.

A. Motion Planning

A robot state is represented by a *configuration*, i.e., an assignment of values to all robot joints. A motion plan moves the robot from one collision-free configuration to another. Formally, let $\mathcal{C} = \mathcal{C}_{\text{free}} \cup \mathcal{C}_{\text{obs}}$ denote the robot's configuration space [18], where $\mathcal{C}_{\text{free}}$ is the set of collision-free configurations and $\mathcal{C}_{\text{obs}}$ is the set of configurations in collision with an obstacle. Let $c_i \in \mathcal{C}_{\text{free}}$ and $c_g \in \mathcal{C}_{\text{free}}$ be the initial and goal configurations, respectively.

Definition 1: A **motion planning problem** is a 4-tuple $\langle \mathcal{C}, F, c_i, c_g \rangle$, where $\mathcal{C}$ is the configuration space, $F : \mathcal{C} \to \{0, 1\}$ is a collision indicator such that $F(c) = 1$ iff $c \in \mathcal{C}_{\text{obs}}$, and c_i and c_g are the initial and goal configurations.

A solution to a motion planning problem is a collision-free trajectory $\tau : [0, 1] \to \mathcal{C}$ such that $\tau(0) = c_i$, $\tau(1) = c_g$, and $F(\tau(t)) = 0$ for all $t \in [0, 1]$. Although motion planning is a fundamental component of robot control, it is insufficient for very long-horizon problems due to its continuous state space and large branching factor. Moreover, it does not capture the perceptual reasoning or closed-loop feedback required by many robot skills.

B. Symbolic Task Planning

Symbolic task planning is widely used for long-horizon reasoning because it operates over a discrete state space with a restricted branching factor. Planning domains and problems are often represented in relational PDDL [25]. Formally, a PDDL domain is a tuple $\langle \mathcal{V}, \mathcal{A} \rangle$, where $\mathcal{V}$ is a vocabulary of parameterized predicates and $\mathcal{A}$ is a set of high-level actions defined over that vocabulary. A high-level state is a set of true grounded predicates, i.e., predicates whose parameters are bound to concrete objects. Each action $a \in \mathcal{A}$ is defined as a tuple $\langle \Theta_a, \textsc{Pre}_a, \textsc{Eff}_a \rangle$, where Θ_a is the set of action parameters, $\textsc{Pre}_a$ is the set of preconditions, and $\textsc{Eff}_a$ is the set of effects.

A symbolic planning problem is a tuple $\langle \mathcal{U}, s_i, \mathcal{G} \rangle$, where $\mathcal{U}$ is the universe of objects, s_i is the initial symbolic state, and $\mathcal{G}$ is a set of goal conditions specified as grounded predicates. A solution to a symbolic planning problem is a sequence of actions that transforms s_i into a state that satisfies $\mathcal{G}$. Although symbolic task planning is well-suited to long-horizon reasoning, its plans are not directly executable on a robot and must be refined into motion plans or other low-level controllers.

C. Task and Motion Planning

Task and motion planning (TAMP) refers to a class of methods [32, 29, 8, 11] that interleave high-level symbolic task planning with low-level motion planning to solve complex robot planning problems. TAMP approaches search for a high-level plan such that each symbolic action can be refined into a feasible motion plan for real-world execution. Following Shah et al. [29], we define a task and motion planning problem as follows:

Definition 2: A **task and motion planning problem** is a tuple $\langle \mathcal{W}, \alpha, \mathcal{V}_\alpha, \mathcal{A}_\alpha, \mathcal{U}, w_i, \mathcal{W}_g, \mathcal{G}, \Gamma \rangle$. Here, $\mathcal{W}$ is the system configuration space of the robot and environment; α is an abstraction function; $\mathcal{V}_\alpha$ is a set of symbolic relations defined using α; $\mathcal{A}_\alpha$ is a set of high-level robot actions, where each action $a \in \mathcal{A}_\alpha$ is executed via motion planning; $\mathcal{U}$ is a universe of objects; $w_i \in \mathcal{W}$ is the initial system configuration, inducing the initial symbolic state $s_i = \alpha(w_i)$; $\mathcal{W}_g \subseteq \mathcal{W}$ is a set of goal configurations; $\mathcal{G}$ is the corresponding set of high-level goal conditions; and Γ is the inverse abstraction function used to generate motion plans for symbolic actions.

A solution to a task and motion planning problem is a task-level plan with corresponding motion plans that, when executed from w_i, reach a configuration in $\mathcal{W}_g$. This formulation is sufficient when every high-level action can be refined into motion planning. However, some robot actions involve non-kinematic constraints, such as sustained contact or closed-loop feedback, and therefore cannot be captured by motion planning alone. The next section extends this formulation to such hybrid settings.

III. Hybrid Robot Planning with General-Purpose Policies

The task and motion planning formulation in Def. 2 focuses on kinematic properties of the robot and environment, such as configurations and poses. Many real-world tasks require additional forms of control, including compliant control, sustained contact, or feedback-driven execution. These tasks may also modify non-spatial properties of objects, such as whether a surface is wet or a dish is dirty.

Environment Model: We model the environment as a collection of objects and robots. Let $\mathcal{O}$ denote the set of objects and $\mathcal{R}$ the set of robots. For each object, a spatial function $P^o : \mathcal{O} \to SE(3)$ maps the object to its 6-DoF pose in a fixed reference frame. We also define object-specific observation functions $\Phi = \{\Phi_o\}_{o \in \mathcal{O}}$, where each Φ_o is a set of attribute classifiers $\{\phi_i^o\}$ for object o, each classifying a distinct attribute such as color, temperature, or cleanliness.

Each robot is modeled as a kinematic tree of rigid-body links connected by joints. A configuration function $C : \mathcal{R} \to \mathcal{C}$ maps each robot $r \in \mathcal{R}$ to its current configuration, and a spatial function $P^r : \mathcal{R} \to SE(3)$ maps each robot to the pose of its base link. An attachment function $\zeta : \mathcal{R} \to 2^{\mathcal{O}}$ maps a robot to the set of objects currently attached to it.

We use first-order logic to represent the symbolic state of the environment. The universe is $\mathcal{U} = \mathcal{O} \cup \mathcal{R}$, and the vocabulary $\mathcal{V}$ includes the spatial functions P^o, P^r, and C, the attachment function ζ, and the observation functions Φ. The high-level state space is defined as the set of grounded predicate assignments over $\mathcal{U}$ and $\mathcal{V}$.

General-Purpose Robot Skills: We assume that each robot is equipped with object-centric skills that modify the state of the environment. Let $\mathcal{A}$ denote the set of such skills. Each skill $a^r \in \mathcal{A}$ for robot $r \in \mathcal{R}$ is modeled as a parameterized option $\langle \Theta_a, \mathcal{I}_a, \beta_a, \pi_a \rangle$, where $\Theta_a \subseteq \mathcal{O}$ is the set of object arguments of the skill, $\mathcal{I}_a$ is an initiation

condition, β_a is a termination condition, and π_a is the skill policy. Both $\mathcal{I}_a$ and β_a are first-order logic formulas over the vocabulary $\mathcal{V}$. A skill can be executed from any state $x \models \mathcal{I}_a$ and follows policy π_a until reaching a state $x' \models \beta_a$, where the skill terminates.

Some skills can be implemented using motion planning because they exploit the structure of the robot configuration space, whereas others (e.g., sustained contact behaviors) require control strategies not easily expressed in kinematic terms. These non-kinematic skills may be implemented using techniques such as behavior cloning [27] or reinforcement learning [13]. We now define a hybrid robot planning problem to model such skills.

Definition 3: A **hybrid robot planning problem** is a tuple $\mathcal{M} = \langle \mathcal{U}, \mathcal{V}, \mathcal{X}, \mathcal{A}, x_i, \mathcal{X}_g \rangle$, where $\mathcal{U} = \mathcal{O} \cup \mathcal{R}$ is the universe, $\mathcal{V}$ is the vocabulary, $\mathcal{X}$ is the state space, $\mathcal{A}$ is a set of robot skills, $x_i \in \mathcal{X}$ is the initial state, and $\mathcal{X}_g \subseteq \mathcal{X}$ is the set of goal states.

A solution to a hybrid robot planning problem is a sequence of skills $[a_1, \ldots, a_n]$ that, when executed from x_i, reaches a state $x_n \in \mathcal{X}_g$. In principle, such problems could be solved by model-based search if the initiation and termination sets of all skills were explicitly available. In our setting, however, these sets are induced by continuous geometric and controller-level constraints, and may depend on scene context not explicitly captured by the symbolic state. As a result, even when the symbolic actions predict that two skills a_i and a_j are compatible, there may not exist states $x, x' \in \mathcal{X}$ such that $x \models \beta_{a_i}$, $x' \models \mathcal{I}_{a_j}$, and the robot can transition between x and x' through collision-free motion planning or skill execution.

To support planning in this setting, each skill $a \in \mathcal{A}$ must expose three properties to the planner: (i) the robot must be able to reach a state $x \in \mathcal{I}_a$ through collision-free motion; (ii) after executing the skill, the robot must be able to return from a state $x \in \beta_a$ to free space; and (iii) the planner must be able to identify compatible initiation and termination states $x_i, x_j \in \mathcal{X}$ such that $x_i \models \mathcal{I}_a$, $x_j \models \beta_a$, and executing π_a from x_i can realize the transition from x_i to x_j. These requirements motivate the planning abstractions introduced in the next section, which enable structured motion planning to connect and compose otherwise black-box robot skills.

IV. Hierarchical Planning for Hybrid Robot Planning Problems

We now present *Task and Skill Planning (TASP)*, our approach for solving hybrid robot planning problems. TASP is designed to satisfy the three planner-facing properties above, enabling the composition of general-purpose object-centric robot skills. We first discuss our solution to the first two properties (Sec. IV-A) that make individual skills composable, and then describe how our solution to the third property (Sec. IV-B) facilitates composing skills using TAMP-style hierarchical planning.

A. General-Purpose Object-Centric Robot Skills as CIPs

The core properties needed to enable the composition of general-purpose object-centric robot skills are the robot's ability to transition into and out of the skills' initiation and termination sets. Abbatematteo et al. [1] develop a key policy class named *Composable Interaction Primitives (CIPs)* that wraps free-space motion plans around object-centric skills. These motion plans allow the robot to use the structure of the configuration space to transition to configurations that satisfy the initiation and termination conditions of its skills.

Intuitively, a CIP is composed of three components: a motion plan that takes the robot to the initiation set of the object-centric skill, the object-centric skill policy, and a motion plan that takes the robot out of the termination set of the skill. The two free-space motion planning components provide a natural means to compose different skills, enabling the robot to transition between the initiation and termination sets of separate skills without requiring them to intersect.

Consider the example of the Spot robot that erases a whiteboard in Fig. 2. Here, the initial configuration in Fig. 2(a) does not satisfy the initiation condition of the `Erase(?board)` skill. However, the robot can use a motion plan to move from its current configuration to the configuration shown in Fig. 2(b), which is in the initiation set of the skill. The robot can then use the pre-trained policy to erase the whiteboard, as shown in Fig. 2(c). Lastly, the Spot uses a motion plan to reach a state, shown in Fig. 2(d), that does not satisfy the termination conditions of the `Erase(?board)` skill. From there, Spot can again use a motion plan to reach the initiation set of the next skill.

Formally, let $a \in \mathcal{A}$ be a robot skill operating on object(s) $o_1, \ldots, o_k$, with initiation and termination conditions $\mathcal{I}_a(P^r, P^{o_{1:k}}, C, \Phi_{o_{1:k}})$ and $\beta_a(P^r, P^{o_{1:k}}, C, \Phi_{o_{1:k}})$, respectively. Here, $\mathcal{I}_a(P^r, P^{o_{1:k}}, C, \Phi_{o_{1:k}})$ expresses that the initiation condition for the robot skill a is a formula over spatial pose functions P^r and $P^{o_{1:k}}$, the configuration function C, and object-specific observation functions $\Phi_{o_{1:k}}$. For example, for the skill `Erase(?board)`, the initiation condition is a function of the robot pose, the robot configuration, the pose of the whiteboard, and whether the whiteboard is dirty.

We define a kinematic projection λ that abstracts away object-specific observation functions from a skill's initiation condition. Specifically, the projection λ maps the initiation condition $\mathcal{I}_a(P^r, P^{o_{1:k}}, C, \Phi_{o_{1:k}})$ to its *kinematic envelope* $\mathcal{I}_a(P^r, P^{o_{1:k}}, C)$. The kinematic envelopes of initiation and termination conditions are only defined over spatial functions, and hence, can be achieved using kinematic motion planning. We formally define a composable interaction primitive for an object-centric robot skill as follows.

Definition 4: Let $a = \langle \Theta_a, \mathcal{I}_a, \beta_a, \pi_a \rangle$ be an object-centric robot skill with initiation condition $\mathcal{I}_a$ and termination condition β_a. A **composable interaction primitive** (CIP) $\tilde{a}$ for the robot skill a is defined as a tuple $\tilde{a} = \langle \Theta_a, \mathcal{I}_a, \beta_a, \pi_a, h, t, \lambda \rangle$. Here, the parameters Θ_a, initiation condition $\mathcal{I}_a$, termination condition β_a, and policy π_a are identical to the robot skill a. λ is the kinematic projection; h is the head motion plan from a configuration $x \not\models \mathcal{I}_a$ to a configuration $x \models \lambda(\mathcal{I}_a)$ and t is the tail motion plan from a configuration $x \models \beta_a$ to a configuration $x \not\models \lambda(\beta_a)$.

Using CIPs, we can compose individual skills using

(a) (b) (c) (d)

Fig. 2. Example keyframes showing the environment state before, during, and after the `Erase(?board)` force-controlled skill. In Fig. 2(a), the Spot robot can reach the whiteboard, but its gripper is stowed. Therefore, the robot is not in a configuration that fulfills the initiation condition for the `Erase(?board)` skill. Fig. 2(b) shows the state after Spot has executed a *head motion plan* to reach a configuration that fulfills the initiation condition for the `Erase(?board)` skill. Fig. 2(c) depicts a state during the execution of the `Erase(?board)` skill. Finally, Fig. 2(d) shows the state after Spot has executed a *tail motion plan* to stow its arm, resulting in a configuration that is no longer in the termination set of the `Erase(?board)` skill.

motion planning. We next describe how TASP uses these abstractions within a hierarchical planner to identify long-horizon plans.

B. Hierarchically Composing CIPs

In this section, we describe our approach to efficiently composing CIPs into long-horizon plans. We also discuss how we compute endpoints for head and tail motion plans, enabling skill composition and successful skill execution.

Let $\tilde{a}_i$ and $\tilde{a}_j$ be two CIPs for the object-centric robot skills a_i and a_j, respectively. The major challenges in composing the skills a_i and a_j are: (i) we must ensure that the head motion plan $h_{\tilde{a}_i}$ reaches a configuration $x_n \models \lambda(\mathcal{I}_{a_i})$ such that π_{a_i} can be successfully executed given the other objects in the environment, (ii) we must ensure that the tail motion plan $t_{\tilde{a}_i}$ ends at a configuration x_m from which the next head motion plan $h_{\tilde{a}_j}$ can be executed, and (iii) we must select the configurations x_n and x_m such that there exist collision-free head and tail motion plans. One could attempt to identify such motion plans using sampling-based motion planners such as the RRT [19] or PRM [16]. However, this is poorly suited to the structure of our problem: as in multi-modal motion planning, feasible solutions may depend on transitions through narrow skill-conditioned compatibility regions in the continuous state space [14]. We therefore cast CIP composition as a hierarchical TAMP problem.

Specifically, we use the ATAM algorithm [29] and entity abstraction [29] to reduce the problem of efficiently combining CIPs to a TAMP problem. Entity abstraction converts each CIP into a high-level action with discrete parameters, including symbols for the end configurations, the head and tail motion plans, and the skill policy.

We modify the ATAM approach to use black-box skill policies instead of calling a motion planner while refining the high-level action arguments. ATAM uses a high-level symbolic planner to compute a high-level plan, applies the inverse abstraction function Γ to reduce each high-level action into a motion planning problem, and uses a sampling-based motion planner to compute motion plans for these refinements. It performs backtracking-based search to recompute a high-level plan until it finds one with valid low-level refinements for each of its high-level actions. This allows the ATAM algorithm to efficiently sample endpoints for head and tail motion plans, overcoming the aforementioned challenges.

V. Real-World Experiments

We validate our method on two robot platforms: a KUKA iiwa bimanual setup and a Boston Dynamics Spot. We first describe the experimental setup for each platform (Sec. V-A) and then discuss our results (Sec. V-B). Videos, code, and supplementary material are available at `https://benned-h.github.io/tasp/`.

A. Experimental Setup

We now describe the experimental setup of our real-world evaluation, beginning with the shared problem setting and then specifying the platform-specific hardware, robot skill inventories, and perception pipelines.

Problem Setting: In each experiment, our Task and Skill Planning (TASP) framework is given as input a PDDL-style symbolic planning problem specifying $\langle \mathcal{U}, s_i, \mathcal{G} \rangle$, which are the universe, initial symbolic state, and goal conditions (see Sec. II-B). The initial low-level state is modeled as a kinematic tree comprised of known object poses and collision geometry. Given this information, our method must plan a feasible sequence of skills that can be executed on the robot to satisfy the goal conditions $\mathcal{G}$.

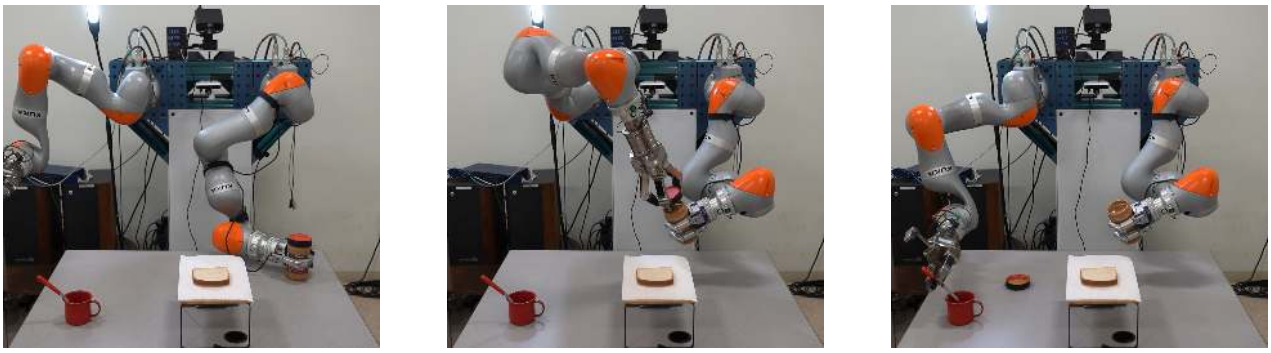
1) Bimanual Manipulator Setting:

Robot Hardware: The bimanual manipulator uses two KUKA LBR iiwa 7 R800 manipulators, one equipped with a BarrettHand BH8-282 gripper, and the other with a SCHUNK Dextrous Hand 2.0 gripper. The robot collects RGB-D data using an Intel RealSense D455 depth camera.

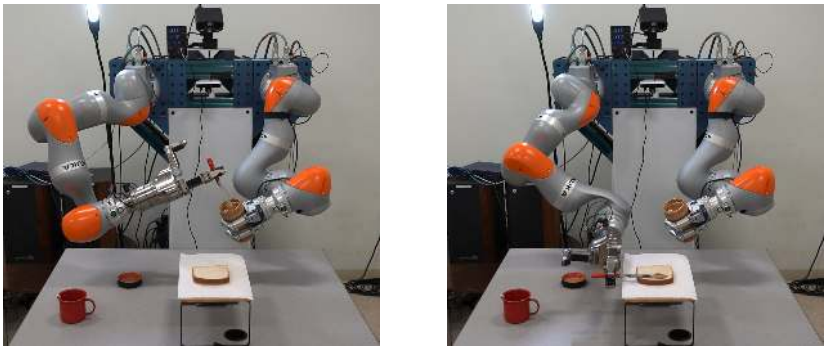
Skill Inventory: The bimanual manipulator is provided with four skills implemented using:

1) *Motion planning*: A `Grasp` skill is based on the CBiRRT algorithm [3] in the OpenRAVE simulator [9].
2) *Trajectory playback*: `Open`, `Scoop`, and `Spread` skills are implemented using relative end-effector trajectory playback modulated by impedance control.

Perception: We use FoundationPose [37] for 6D object pose estimation, with poses initialized from CNOS [26] object segmentations. These models require a textured mesh



(a) `Grasp(j0)` (b) `Open(j0)` (c) `Grasp(k0)`



(d) `Scoop(k0, j0)` (e) `Spread(k0, b0)`

Fig. 3. We conduct real-world experiments on two robot platforms to demonstrate the versatility of our task and skill planning approach. In our bimanual manipulation setting, the robot can pick up a jar and a kitchen knife using a `Grasp(?object)` skill implemented using motion planning (Figs. 3(a) and 3(c)). The bimanual manipulator has three additional skills that combine trajectory playback and impedance control: the robot uses `Open(?jar)` to open a jar of peanut butter using both grippers (Fig. 3(b)); scoops peanut butter onto a knife using `Scoop(?knife, ?jar)` (Fig. 3(d)); and spreads peanut butter by executing `Spread(?knife, ?bread)` (Fig. 3(e)).

for each pose-estimated object. We also provide a collision model of each object to the motion planner.

2) Mobile Manipulation Setting:

Robot Hardware: In our mobile manipulation experiments, we use a Boston Dynamics Spot robot equipped with the Spot Arm manipulator and the Spot EAP sensor payload. The robot's sensors include onboard RGB-D cameras, an RGB-D end-effector sensor, and a Velodyne VLP-16 LiDAR. Due to onboard compute limitations, we run planning, learned policy control, and perception on an external machine with an NVIDIA GeForce RTX 4090 GPU.

Robot Skill Inventory: In our experiments, the Spot robot is equipped with an inventory of seven skills implemented using a diverse set of controllers:

1) *Motion planning*: We implement `Pick` and `Place` using MoveIt! [6] and the Open Motion Planning Library [33] for motion planning.
2) *Navigation*: The `GoTo` skill performs global planning with a grid-based A* planner and local control through the Boston Dynamics Spot SDK.
3) *Trajectory playback*: The `OpenDrawer` skill executes an object-relative end-effector trajectory with timed pauses and gradual gripper closure, exploiting gripper compliance to grasp the drawer handle reliably.
4) *Force control*: We implement an `Erase` skill using force control through the Boston Dynamics Spot SDK.
5) *Behavior cloning*: We implement a `CloseDoor` skill by training an Action Chunking Transformer [38] policy on real-world demonstrations using LeRobot [4].
6) *Off-the-shelf skill*: The `OpenDoor` skill detects the door handle using Gemini Robotics-ER 1.5 [12] and then calls Spot's built-in door-opening functionality as a black-box skill.

Across these seven skills, we highlight that only three rely primarily on motion planning (i.e., `Pick`, `Place`, and `GoTo`). Even for these skills, execution noise requires online pose estimation and, when needed, motion replanning.

Perception Pipeline: Before the experiment, the robot is teleoperated through the environment to construct an occupancy grid map. During this phase, the robot also performs object pose estimation using AprilTag visual fiducial markers [35]. The occupancy grid and estimated object poses are used to initialize the TASP system during planning. We do not provide full kinematic models of articulated objects such as drawers or doors. Instead, the system receives pre- and post-skill static models for each applicable skill, corresponding to the skill's kinematic envelope.

B. Experimental Results

We evaluate our approach using two real-world experiments designed to require long-horizon hierarchical planning

(a) `OpenDrawer(c1)` (b) `OpenDoor(d1)` (c) `GoTo(f1)`

(d) `Place(e1, f1)` (e) `CloseDoor(d1)` (f) `Erase(b1)`

Fig. 4. We evaluate our method on a multi-room mobile manipulation task with non-monotonic structure and a heterogeneous skill inventory. The Spot robot opens a drawer using a trajectory playback skill `OpenDrawer(?drawer)` (Fig. 4(a)), opens a door using an off-the-shelf skill `OpenDoor(?door)` (Fig. 4(b)), navigates and places the eraser using motion planning skills `GoTo(?location)` and `Place(?object, ?surface)` (Figs. 4(c) and 4(d)), closes the door with a learned policy `CloseDoor(?door)` (Fig. 4(e)), and erases the board using a force control skill `Erase(?board)` (Fig. 4(f)).

that integrates motion planning and general-purpose skills.

1) Bimanual Manipulation Experiment: In our bimanual manipulation experiment, the environment initially contains a kitchen knife `k0`, a closed jar of peanut butter `j0`, and a slice of bread `b0` (Fig. 3(a)). The goal is to spread peanut butter on the bread. In the solution plan produced by our planner, the robot first executes `Grasp(j0)` to pick up the jar, followed by `Open(j0)` to open it using both grippers (Figs. 3(a)–3(b)). The robot then runs `Grasp(k0)` to pick up the knife using its available gripper (Fig. 3(c)). Finally, the robot executes `Scoop(k0, j0)` to scoop peanut butter onto the knife, then spreads it onto the bread using `Spread(k0, b0)` (Figs. 3(d) and 3(e)).

2) Mobile Manipulation Experiment: Our mobile manipulation experiment tasks the Spot robot with erasing a wall-mounted writable board `b1` in a connecting room behind an initially closed door `d1`. When the door is open, it blocks access to the board, but when it is shut, Spot cannot enter the room. That room also contains a filing cabinet `f1` near the board. The robot begins in the same room as a closed chest of drawers `c1` containing a whiteboard eraser `e1`.

Given this task, the solution plan produced by our planner includes 14 high-level actions, seven of which are `GoTo(?location)` navigation skills alternating with manipulation skills. For brevity, we therefore omit `GoTo` skills in the following description.

While in the initial room, Spot first uses `OpenDrawer(c1)` to open the chest's top drawer, then navigates to the door and executes `OpenDoor(d1)` (Figs. 4(a) and 4(b)). Spot then returns to the chest of drawers and runs `Pick(e1)` to pick the whiteboard eraser out of the opened drawer, as depicted in Figure 1(a). With the eraser in hand, Spot heads into the other room and places the eraser onto the filing cabinet using `Place(e1, f1)` (Figs. 4(c) and 4(d)). The robot then runs `CloseDoor(d1)` to close the door and make the board accessible (Fig. 4(e)). Finally, Spot picks the eraser back up using `Pick(e1)` and erases the text written on the board by executing `Erase(b1)` (Fig. 4(f)). In both experiments, we observe that our approach can solve hybrid robot planning problems by integrating motion planning with general-purpose skills, including in scenarios with non-monotonic task progression.

VI. Related Work

This work is most closely related to task and motion planning (TAMP), which combines discrete task-level reasoning with continuous motion planning to solve long-horizon problems [32, 34, 8, 29, 11]. Recent work has extended TAMP-style methods beyond purely kinematic actions by incorporating learned skills [36, 31, 23], closed-loop controllers [28, 7], and uncertainty-aware execution [7, 10]. Accordingly, our contribution is not that skills can be combined with TAMP in general, but rather that our approach plans over a *heterogeneous, pre-existing* inventory of robot skills while retaining geometric failure reasoning for non-downward-refinable actions [24].

Prior approaches have incorporated learned skill models, learned feasibility or effect predictors, learned value functions, or controller libraries with explicitly modeled planning operators [36, 31, 20, 2, 28, 7]. In contrast, we focus on planning with *pre-existing, black-box* skills drawn from multiple

implementation classes, including skills whose internal mechanics may be unavailable to the planner. Our approach uses kinematic envelopes as a planner-facing geometric surrogate for such skills, allowing the planner to reason about where a skill can be executed and about its nominal geometric effects without modeling its internal controller dynamics.

Our methods are also related to prior work that combines task-level planning or TAMP with contact-rich, forceful, or reinforcement-learned behaviors [15, 21, 22, 23, 5]. We view these approaches as complementary: rather than emphasizing a single class of skill policy, we aim to support a mixed inventory of motion-planned, learned, force-controlled, trajectory-playback, and built-in robot skills within one planning framework.

VII. Conclusion and Future Work

This paper formalizes a hybrid robot planning problem that requires the composition of general-purpose skills and motion planning. We present a hierarchical approach to solving such planning problems, and demonstrate it on a bimanual manipulator and a mobile manipulator in challenging real-world scenarios. Currently, we assume access to a predefined predicate vocabulary and per-skill kinematic envelopes. In future work, we intend to acquire these representations using world-model learning approaches that automatically learn initiation and termination sets, symbolic models, and kinematic envelopes for robot motor skills [17, 30].

Acknowledgment

This work was supported in part by ONR REPRISM MURI N00014-24-1-2603, ONR grant 00014-22-1-2592, and the Robotics and AI Institute (RAI).

References

[1] B. Abbatematteo, E. Rosen, S. Thompson, T. Akbulut, S. Rammohan, and G. Konidaris, "Composable interaction primitives: A structured policy class for efficiently learning sustained-contact manipulation skills," in *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2024, pp. 7522–7529.

[2] C. Agia, T. Migimatsu, J. Wu, and J. Bohg, "Stap: Sequencing task-agnostic policies," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*, 2023, pp. 7951–7958.

[3] D. Berenson, S. S. Srinivasa, D. Ferguson, and J. J. Kuffner, "Manipulation planning on constraint manifolds," in *Proceedings of IEEE International Conference on Robotics and Automation*. IEEE, 2009, pp. 625–632.

[4] R. Cadene, S. Alibert, A. Soare, Q. Gallouedec, A. Zouitine, S. Palma, P. Kooijmans, M. Aractingi, M. Shukor, D. Aubakirova, M. Russi, F. Capuano, C. Pascal, J. Choghari, J. Moss, and T. Wolf, "Lerobot: State-of-the-art machine learning for real-world robotics in pytorch," https://github.com/huggingface/lerobot, 2024.

[5] S. Cheng and D. Xu, "League: Guided skill learning and abstraction for long-horizon manipulation," *IEEE Robotics and Automation Letters*, vol. 8, no. 10, pp. 6451–6458, 2023.

[6] D. Coleman, I. A. Sucan, S. Chitta, and N. Correll, "Reducing the barrier to entry of complex robotic software: a MoveIt! case study," *Journal of Software Engineering for Robotics*, vol. 5, pp. 3–16, 2014.

[7] A. Curtis, G. Matheos, N. Gothoskar, V. Mansinghka, J. B. Tenenbaum, T. Lozano-Pérez, and L. P. Kaelbling, "Partially Observable Task and Motion Planning with Uncertainty and Risk Awareness," in *Proceedings of Robotics: Science and Systems*, July 2024.

[8] N. T. Dantam, Z. K. Kingston, S. Chaudhuri, and L. E. Kavraki, "An incremental constraint-based framework for task and motion planning," *The International Journal of Robotics Research*, vol. 37, no. 10, pp. 1134–1151, 2018.

[9] R. Diankov, "Automated construction of robotic manipulation programs," Ph.D. dissertation, Carnegie Mellon University, Robotics Institute, August 2010. [Online]. Available: http://www.programmingvision.com/rosen_diankov_thesis.pdf

[10] C. R. Garrett, C. Paxton, T. Lozano-Pérez, L. P. Kaelbling, and D. Fox, "Online replanning in belief space for partially observable task and motion problems," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, 2020, pp. 5678–5684.

[11] C. R. Garrett, R. Chitnis, R. Holladay, B. Kim, T. Silver, L. P. Kaelbling, and T. Lozano-Pérez, "Integrated task and motion planning," *Annual review of control, robotics, and autonomous systems*, vol. 4, no. 1, pp. 265–293, 2021.

[12] Gemini Robotics Team *et al.*, "Gemini Robotics 1.5: Pushing the Frontier of Generalist Robots with Advanced Embodied Reasoning, Thinking, and Motion Transfer," 2025. [Online]. Available: https://arxiv.org/abs/2510.03342

[13] D. Haramati, T. Daniel, and A. Tamar, "Entity-centric reinforcement learning for object manipulation from pixels," *arXiv preprint arXiv:2404.01220*, 2024.

[14] K. Hauser and J.-C. Latombe, "Multi-modal Motion Planning in Non-expansive Spaces," *The International Journal of Robotics Research*, vol. 29, no. 7, pp. 897–915, 2010. [Online]. Available: https://doi.org/10.1177/0278364909352098

[15] R. Holladay, T. Lozano-Pérez, and A. Rodriguez, "Planning for multi-stage forceful manipulation," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, 2021, pp. 6556–6562.

[16] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Transactions on Robotics and Automation*, vol. 12, no. 4, pp. 566–580, 2002.

[17] G. Konidaris, L. P. Kaelbling, and T. Lozano-Perez, "From Skills to Symbols: Learning Symbolic Repre-

sentations for Abstract High-Level Planning," *Journal of Artificial Intelligence Research*, vol. 61, pp. 215–289, 2018.

[18] S. M. LaValle, *Planning Algorithms*. USA: Cambridge University Press, 2006.

[19] S. M. LaValle and J. J. Kuffner, "Rapidly-exploring random trees: Progress and prospects," *Algorithmic and computational robotics*, pp. 303–307, 2001.

[20] J. Liang, M. Sharma, A. LaGrassa, S. Vats, S. Saxena, and O. Kroemer, "Search-based task planning with learned skill effect models for lifelong robotic manipulation," in *2022 International Conference on Robotics and Automation (ICRA)*, 2022, pp. 6351–6357.

[21] J. Liang, X. Cheng, and O. Kroemer, "Learning preconditions of hybrid force-velocity controllers for contact-rich manipulation," in *Proceedings of The 6th Conference on Robot Learning*, ser. Proceedings of Machine Learning Research, K. Liu, D. Kulic, and J. Ichnowski, Eds., vol. 205. PMLR, 14–18 Dec 2023, pp. 679–689. [Online]. Available: https://proceedings.mlr.press/v205/liang23a.html

[22] G. Liu, J. de Winter, D. Steckelmacher, R. K. Hota, A. Nowe, and B. Vanderborght, "Synergistic task and motion planning with reinforcement learning-based non-prehensile actions," *IEEE Robotics and Automation Letters*, vol. 8, no. 5, pp. 2764–2771, 2023.

[23] G. Liu, J. de Winter, Y. Durodié, D. Steckelmacher, A. Nowe, and B. Vanderborght, "Optimistic reinforcement learning-based skill insertions for task and motion planning," *IEEE Robotics and Automation Letters*, vol. 9, no. 6, pp. 5974–5981, 2024.

[24] B. Marthi, S. Russell, and J. Wolfe, "Angelic Semantics for High-Level Actions," in *Proceedings of the Seventeenth International Conference on International Conference on Automated Planning and Scheduling*, ser. ICAPS'07. AAAI Press, 2007, pp. 232–239. [Online]. Available: https://aaai.org/papers/icaps-07-030-angelic-semantics-for-high-level-actions/

[25] D. McDermott, M. Ghallab, A. Howe, C. Knoblock, A. Ram, M. Veloso, D. S. Weld, and D. Wilkins, "PDDL – The Planning Domain Definition Language," Yale Center for Computational Vision and Control, Tech. Rep. CVC TR-98-003/DCS TR-1165, 1998.

[26] V. N. Nguyen, T. Groueix, G. Ponimatkin, V. Lepetit, and T. Hodan, "CNOS: A strong baseline for cad-based novel object segmentation," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023, pp. 2134–2140.

[27] A. O'Neill, A. Rehman, A. Maddukuri, A. Gupta, A. Padalkar, A. Lee, A. Pooley, A. Gupta, A. Mandlekar, A. Jain *et al.*, "Open x-embodiment: Robotic learning datasets and rt-x models: Open x-embodiment collaboration 0," in *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2024, pp. 6892–6903.

[28] T. Pan, R. Shome, and L. E. Kavraki, "Task and motion planning for execution in the real," *IEEE Transactions on Robotics*, vol. 40, pp. 3356–3371, 2024.

[29] N. Shah, D. K. Vasudevan, K. Kumar, P. Kamojjhala, and S. Srivastava, "Anytime integrated task and motion policies for stochastic environments," in *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 9285–9291.

[30] N. Shah, J. Nagpal, P. Verma, and S. Srivastava, "From the Real World to Logic and Back: Learning Symbolic World Models for Long-Horizon Planning ," in *Proceedings of Conference on Robot Learning*, 2025.

[31] T. Silver, A. Athalye, J. B. Tenenbaum, T. Lozano-Pérez, and L. P. Kaelbling, "Learning neuro-symbolic skills for bilevel planning," in *Proceedings of The 6th Conference on Robot Learning*, ser. Proceedings of Machine Learning Research, K. Liu, D. Kulic, and J. Ichnowski, Eds., vol. 205. PMLR, 14–18 Dec 2023, pp. 701–714. [Online]. Available: https://proceedings.mlr.press/v205/silver23a.html

[32] S. Srivastava, E. Fang, L. Riano, R. Chitnis, S. Russell, and P. Abbeel, "Combined task and motion planning through an extensible planner-independent interface layer," in *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2014, pp. 639–646.

[33] I. A. Şucan, M. Moll, and L. E. Kavraki, "The Open Motion Planning Library," *IEEE Robotics & Automation Magazine*, vol. 19, no. 4, pp. 72–82, December 2012, https://ompl.kavrakilab.org.

[34] M. Toussaint, "Logic-geometric programming: An optimization-based approach to combined task and motion planning." in *Proceedings of International Joint Conference on Artificial Intelligence*, 2015, pp. 1930–1936.

[35] J. Wang and E. Olson, "AprilTag 2: Efficient and robust fiducial detection," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, October 2016.

[36] Z. Wang, C. R. Garrett, L. P. Kaelbling, and T. Lozano-Pérez, "Learning compositional models of robot skills for task and motion planning," *The International Journal of Robotics Research*, vol. 40, no. 6–7, pp. 866–894, 2021. [Online]. Available: https://doi.org/10.1177/02783649211004615

[37] B. Wen, W. Yang, J. Kautz, and S. Birchfield, "Foundationpose: Unified 6d pose estimation and tracking of novel objects," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024, pp. 17 868–17 879.

[38] T. Z. Zhao, V. Kumar, S. Levine, and C. Finn, "Learning fine-grained bimanual manipulation with low-cost hardware," in *Proceedings of Robotics: Science and Systems*, 2023. [Online]. Available: https://roboticsconference.org/2023/program/papers/016/