# Text2Motion: from natural language instructions to feasible plans

Kevin Lin[1] · Christopher Agia[1,2] · Toki Migimatsu[1] · Marco Pavone[2] · Jeannette Bohg[1]

## Abstract

We propose Text2Motion, a language-based planning framework enabling robots to solve sequential manipulation tasks that require long-horizon reasoning. Given a natural language instruction, our framework constructs both a task- and motion-level plan that is verified to reach inferred symbolic goals. Text2Motion uses feasibility heuristics encoded in Q-functions of a library of skills to guide task planning with Large Language Models. Whereas previous language-based planners only consider the feasibility of individual skills, Text2Motion actively resolves geometric dependencies spanning skill sequences by performing geometric feasibility planning during its search. We evaluate our method on a suite of problems that require long-horizon reasoning, interpretation of abstract goals, and handling of partial affordance perception. Our experiments show that Text2Motion can solve these challenging problems with a success rate of 82%, while prior state-of-the-art language-based planning methods only achieve 13%. Text2Motion thus provides promising generalization characteristics to semantically diverse sequential manipulation tasks with geometric dependencies between skills. Qualitative results are made available at https://sites.google.com/stanford.edu/text2motion.

**Keywords** Long-horizon planning · Robot manipulation · Large language models

## 1 Introduction

Long-horizon robot planning is traditionally formulated as a joint symbolic and geometric reasoning problem, where the symbolic reasoner is supported by a formal logic representation [e.g. first-order logic (Aeronautiques et al., 1998)]. Such

✉ Kevin Lin
  kevinlin0@stanford.edu
  https://kevin-thankyou-lin.github.io/

✉ Christopher Agia
  cagia@stanford.edu
  https://www.chrisagia.com/

  Toki Migimatsu
  takatoki@stanford.edu
  https://cs.stanford.edu/~takatoki/

  Marco Pavone
  pavone@stanford.edu
  https://profiles.stanford.edu/marco-pavone

  Jeannette Bohg
  bohg@stanford.edu
  https://web.stanford.edu/~bohg/

1  Department of Computer Science, Stanford University, Stanford, CA, USA

2  Department of Aeronautics and Astronautics, Stanford University, Stanford, CA, USA

systems can generalize within the logical planning domain specified by experts. However, many desirable properties of plans that can be conveniently expressed in language by non-expert users may be cumbersome to specify in formal logic. Examples include the specification of user intent or preferences.

The emergence of *Large Language Models* (LLMs) (Bommasani et al., 2021) as a task-agnostic reasoning module presents a promising pathway to general robot planning capabilities. Several recent works (Ahn et al., 2022; Huang et al., 2022; Liang et al., 2022; Wu et al., 2023) capitalize on their ability to perform task planning for robot systems without needing to manually specify symbolic planning domains. Nevertheless, these prior approaches adopt myopic or open-loop execution strategies, trusting LLMs to produce correct plans without verifying them on the symbolic or geometric level. Such strategies are challenged in long-horizon settings, where the task planning abilities of even the most advanced LLMs appear to degrade (Valmeekam et al., 2022), and the overall success of a seemingly correct task plan depends as well on how it is executed to ensure long-horizon feasibility. Therefore, we ask in this paper: how can we verify the correctness and feasibility of LLM-generated plans prior to execution?

We propose **Text2Motion**, a language-based planning framework that interfaces an LLM with a library of learned skills and a geometric feasibility planner (Agia et al., 2022) to solve complex sequential manipulation tasks (Fig. 1). Our contributions are two-fold: (i) a hybrid LLM planner that synergistically integrates shooting-based and search-based planning strategies to construct geometrically feasible plans for tasks not seen by the skills during training; and (ii) a plan termination method that infers goal states from a natural language instruction to verify the completion of plans before executing them. We find that our planner achieves a success rate of 82% on a suite of challenging table top manipulation tasks, while prior language-based planning methods achieve a 13% success rate.

## 2 Related work

### 2.1 Language for robot planning

Language is increasingly being explored as a medium for solving long-horizon robotics problems. For instance, *Language-conditioned policies* (LCPs) are not only used to learn short-horizon skills (Stepputtis et al., 2020; Jang et al., 2021; Shao et al., 2021; Shridhar et al., 2022a, b; Jiang et al., 2022), but also long-horizon policies (Mees et al., 2022; Brohan et al., 2022; Dalal et al., 2023). However, LCPs require expensive data collection and training procedures if they are to generalize to a wide distribution of long-horizon tasks with diverse instructions.

Several recent works leverage the generative qualities of LLMs by prompting them to predict long-horizon plans. Huang et al. (2022) grounds an LLM planner to admissible action sets for task planning, Silver et al. (2022), Liu et al. (2023) explore the integration of LLMs with PDDL (Aeronautiques et al., 1998), and (Wang et al., 2023; Skreta et al., 2023) focuses on task-level replanning with LLMs. Tangential works shift the representation of plans from action sequences to code (Liang et al., 2022; Singh et al., 2022; Zelikman et al., 2022; Vemprala et al., 2023) and embed task queries, robot actions, solution samples, and fallback behaviors as programs in the prompt. In contrast to these works, which primarily address challenges in task planning with LLMs, we focus on verifying LLM-generated plans for feasibility on the geometric level.

Closest in spirit to our work are SayCan (Ahn et al., 2022) and Inner Monologue (IM) (Huang et al., 2022) which at each timestep score the *usefulness* and *feasibility* of all possible skills and execute the one with the highest score. Termination occurs when the score of the `stop` "skill" is larger than any other. IM provides additional sources of feedback to the LLM in the form of skill successes and task-progress cues.
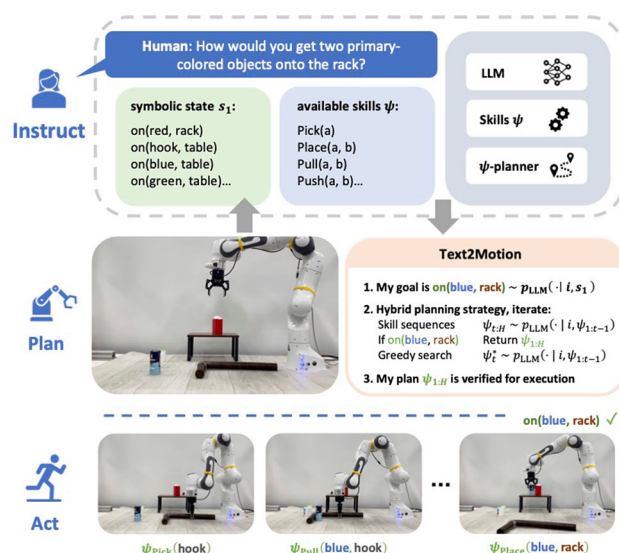


**Fig. 1** To carry out the instruction "get two primary-colored objects onto the rack," the robot must apply symbolic reasoning over the scene description and language instruction to deduce what skills should be executed to acquire a second primary-colored object, after noticing that a red object is already on the rack (i.e. on(red, rack)). It must also apply geometric reasoning to ensure that skills are sequenced in a manner that is likely to succeed. Unlike prior work (Ahn et al., 2022; Huang et al., 2022) that myopically executes skills at the current timestep, Text2Motion constructs *sequences* of skills and coordinates their geometric dependencies with geometric feasibility planning (Agia et al., 2022). Upon planning the skill sequence Pick(hook), Pull(blue, hook), Pick(blue), Place(blue, rack), our method computes a grasp position on the hook that enables pulling the blue object into the robot workspace so that it can be successfully picked up in the next step (Color figure online)

While SayCan and IM are evaluated on a diverse range of tasks, there are several limitations that impede their performance in the settings we study. First, by only myopically executing the next skill at each timestep, they may fail to account for geometric dependencies that exist over the extent of a skill sequence. For an example, see Fig. 1. Second, they do not explicitly predict a multi-step plan, which prevents verification of desired properties or outcomes prior to execution. Examples of such properties could include whether the final state induced by the plan satisfies symbolic constraints or whether the plan adheres to safety criteria. Lastly, these methods ignore the uncertainty of skill feasibility predictions (i.e. affordances), which Agia et al. (2022) demonstrates is important when sequencing learned skills to solve long-horizon tasks. By addressing these limitations, **Text2Motion** outperforms SayCan and IM by a large margin on tasks with geometric dependencies, as demonstrated in the experiments.

### 2.2 Task and motion planning

Task and Motion Planning (TAMP) refers to a problem setting in which a robot solves long-horizon tasks through sym-

bolic and geometric reasoning (Kaelbling & Lozano-Pérez, 2012; Garrett et al., 2021). The hierarchical approach (Kaelbling and Lozano-Pérez, 2011) characterizes the most common family of solution methods. Such methods typically employ a) a symbolic task planner (Bonet & Geffner, 2001; Helmert, 2006) to produce a candidate plan skeleton, and b) a motion planner to verify the plan skeleton for its geometric feasibility and compute a motion trajectory subject to robot and environmental constraints (Garrett et al., 2020; Toussaint, 2015; Driess et al., 2019).

For complex tasks, classical TAMP solvers (Kaelbling and Lozano-Pérez, 2011; Lagriffoul et al., 2014; Toussaint, 2015; Dantam et al., 2016; Bidot et al., 2017; Garrett et al., 2020) may iterate between task planning and motion planning for minutes until a plan is found. To amortize planning costs, works learn sampling distributions (Wang et al., 2018; Xu et al., 2021; Kim et al., 2019; Kim et al., 2022; Agia et al., 2022), visual feasibility heuristics (Driess et al., 2020a, b; Driess et al., 2021a), low-level controllers (Driess et al., 2021b; Silver et al., 2022), or state sparsifiers (Chitnis et al., 2021; Silver et al., 2021a), from datasets of solutions computed by classical TAMP solvers. Another line of works learn symbolic representations for TAMP (Kroemer and Sukhatme, 2016; Ames et al., 2018; Konidaris et al., 2018; Silver et al., 2021; Wang et al., 2021; Curtis et al., 2022; Chitnis et al., 2022; Silver et al., 2022), often from task-specific symbolic transition experience.

As is common in TAMP, **Text2Motion** also assumes knowledge of task-relevant objects and their poses in order to plan feasible trajectories for long-horizon tasks. However, central to our work is the use of LLMs instead of symbolic task planners often used in TAMP (Garrett et al., 2021), and language as convenient medium to express tasks that may be cumbersome to specify in formal logic [e.g. user preferences (Wu et al., 2023)]. Accordingly, we address early challenges concerning the reliable use of LLMs (discussed in Sect. 2.1) in the long-horizon settings typically solved by TAMP. **Text2Motion** thereby presents several qualitative differences from TAMP: (i) the ability to interpret free-form language instructions for the construction of multi-step plans, and (ii) the capacity to reason over an unrestricted set of object classes and object properties, both of which are supported by the commonsense knowledge of LLMs (Brown et al., 2020). We leave the extension of our framework to open-world settings [e.g. via environment exploration (Chen et al., 2022a) or interaction (Curtis et al., 2022)] to future work.

## 3 Problem setup

We aim to solve long-horizon sequential manipulation problems that require symbolic and geometric reasoning from a natural language instruction $i$ and the initial state of the environment $s_1$. We assume a closed-world setting, whereby the initial state $s_1$ contains knowledge of task-relevant objects and their poses as provided by an external perception system ("Appendix D.1"). Fulfillment of the instruction $i$ corresponds to achieving a desired goal configuration of the task-relevant objects which can be symbolically expressed with a closed set of predicates ("Appendix B.1").

### 3.1 LLM and skill library

We assume access to an LLM and a library of skills $\mathcal{L}^\psi = \{\psi^1, \ldots, \psi^N\}$. Each skill $\psi$ consists of a policy $\pi(a|s)$ and a parameterized manipulation primitive $\phi(a)$ (Felip et al., 2013), and is associated with a contextual bandit, or a single-timestep Markov Decision Process (MDP):

$$\mathcal{M} = (\mathcal{S}, \mathcal{A}, T, R, \rho), \tag{1}$$

where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $T(s'|s, a)$ is the transition model, $R(s, a, s')$ is the binary reward function, and $\rho(s)$ is the initial state distribution. When a skill $\psi$ is executed, an action $a \in \mathcal{A}$ is sampled from its policy $\pi(a|s)$ and fed to its primitive $\phi(a)$, which consumes the action and executes a series of motor commands on the robot. If the skill succeeds, it receives a binary reward of $r$ (or $\neg r$ if it fails). We subsequently refer to policy actions $a \in \mathcal{A}$ as *parameters* for the primitive, which, depending on the skill, can represent grasp poses, placement locations, and pulling or pushing distances ("Appendix A.1").

A timestep in our environment corresponds to the execution of a single skill. We assume that each skill comes with a language description and that methods exist to obtain its policy $\pi(a|s)$, Q-function $Q^\pi(s, a)$, and dynamics model $T^\pi(s'|s, a)$. Our framework is agnostic to the approach used to obtain these models. We also assume a method to convey the environment state $s \in \mathcal{S}$ to the LLM as natural language.

### 3.2 The planning objective

Our objective is to find a plan in the form of a sequence of skills $[\psi_1, \ldots, \psi_H]$ (for notational convenience, we hereafter represent sequences with range subscripts, e.g. $\psi_{1:H}$) that is both likely to satisfy the instruction $i$ and can be successfully executed from the environment's initial state $s_1$. This objective can be expressed as the joint probability of skill sequence $\psi_{1:H}$ and binary rewards $r_{1:H}$ given the instruction $i$ and initial state $s_1$:

$$\begin{aligned} & p(\psi_{1:H}, r_{1:H} \mid i, s_1) \\ & = p(\psi_{1:H} \mid i, s_1)\, p(r_{1:H} \mid i, s_1, \psi_{1:H}). \end{aligned} \tag{2}$$

The first term in this product $p(\psi_{1:H} \mid i, s_1)$ considers the probability that the skill sequence $\psi_{1:H}$ will satisfy the

instruction $i$ from a symbolic perspective. However, a symbolically correct skill sequence may fail during execution due to kinematic constraints of the robot or geometric dependencies spanning the skill sequence. We must also consider the *success probability* of the skill sequence $\psi_{1:H}$ captured by the second term in this product $p(r_{1:H} \mid i, s_1, \psi_{1:H})$. The success probability depends on the parameters $a_{1:H}$ fed to the underlying sequence of primitives $\phi_{1:H}$ that control the robot's motion:

$$p(r_{1:H} \mid i, s_1, \psi_{1:H}) = p(r_{1:H} \mid s_1, a_{1:H}). \tag{3}$$

Equation 3 represents the probability that skills $\psi_{1:H}$ achieve rewards $r_{1:H}$ when executed from initial state $s_1$ with parameters $a_{1:H}$; which is independent of the instruction $i$. If just one skill fails (reward $\neg r$), then the entire plan fails.

## 3.3 Geometric feasibility planning

The role of geometric feasibility planning is to maximize the success probability (Eq. 3) of a skill sequence $\psi_{1:H}$ by computing an optimal set of parameters $a_{1:H}$ for the underlying primitive sequence $\phi_{1:H}$. This process is essential for finding plans that maximize the overall planning objective in Eq. 2. In our experiments, we leverage Sequencing Task-Agnostic Policies (STAP) (Agia et al., 2022).

STAP resolves geometric dependencies across the skill sequence $\psi_{1:H}$ by maximizing the product of step reward probabilities of parameters $a_{1:H}$:

$$a^*_{1:H} = \arg\max_{a_{1:H}} \; E_{s_{2:H}} \left[ \prod_{t=1}^{H} p(r_t \mid s_t, a_t) \right], \tag{4}$$

where future states $s_{2:H}$ are predicted by dynamics models $s_{t+1} \sim T^{\pi_t}(\cdot \mid s_t, a_t)$. Note that the reward probability $p(r_t \mid s_t, a_t)$ is equivalent to the Q-function $Q^{\pi_t}(s_t, a_t)$ for skill $\psi_t$ in a contextual bandit setting with binary rewards (Eq. 1). The success probability of the optimized skill sequence $\psi_{1:H}$ is thereby approximated by the product of Q-functions evaluated from initial state $s_1$ along a sampled trajectory $s_{2:H}$ with parameters $a^*_{1:H}$:

$$p(r_{1:H} \mid s_1, a_{1:H}) \approx \prod_{t=1}^{H} Q^{\pi_t}(s_t, a^*_t). \tag{5}$$

In principle, our framework is agnostic to the specific approach used for geometric feasibility planning, requiring only that it is compatible with the skill formalism defined in Sect. 3.1 and provides a reliable estimate of Eq. 3.

## 4 Methods

The core idea of this paper is to ensure the geometric feasibility of an LLM task plan—and thereby its correctness—by predicting the success probability (Eq. 3) of learned skills that are sequenced according to the task plan. In the following sections, we outline two strategies for planning with LLMs and learned skills: a shooting-based planner and a search-based planner. We then introduce the full planning algorithm, **Text2Motion**, which synergistically integrates the strengths of both strategies. These strategies represent different ways of maximizing the overall planning objective in Eq. 2.

### 4.1 Goal prediction

Plans with high overall objective scores (Eq. 2) are not guaranteed to satisfy their instruction. Consider the instruction "move all the dishes from the table to the sink" issued in an environment with two dishes on the table. While a plan that picks and places one of the two dishes in the sink may have a high language model likelihood and success probability, it fails to satisfy the instruction.

The first step in all planning strategies is to convert the language instruction into a goal condition that can be checked against a candidate sequence of skills. Given an instruction $i$, a set of objects $O$ in the scene, and a library of predicate classifiers $\mathcal{L}^\chi = \{\chi^1, \ldots, \chi^M\}$, we use the LLM to predict a set of $|\mathcal{G}|$ symbolic goal propositions $\mathcal{G} = \{g_1, \ldots, g_j\}$ that would satisfy the instruction. Each goal proposition $g \in \mathcal{G}$ is a set of predicates grounded over objects in the scene. Each predicate is a binary-valued function over objects and has a one-to-one correspondence with a predicate classifier $\chi \in \mathcal{L}^\chi$ that implements the predicate (details in "Appendix B.1"). We define a satisfaction function $F^{\mathcal{G}}_{\text{sat}}(s) : \mathcal{S} \to \{0, 1\}$ which takes as input a geometric state $s$ and evaluates to 1 if any goal proposition $g \in \mathcal{G}$ predicted by the LLM holds in state $s$. A sequence of skills $\psi_{1:H}$ is said to satisfy the instruction $i$ iff:

$$\exists \, s \in s_{2:H+1} : F^{\mathcal{G}}_{\text{sat}}(s) = 1, \tag{6}$$

where the future states $s_{2:H+1}$ are predicted by the geometric feasibility planner (see Sect. 3.3). If $F^{\mathcal{G}}_{\text{sat}}(s_t)$ evaluates to 1 for a geometric state $s_t$ at timestep $t \leq H + 1$, then the planner returns the subsequence of skills $\psi_{1:t-1}$ for execution.

### 4.2 Shooting-based planning

The planner is responsible for finding geometrically feasible plans that satisfy the goal condition predicted by the LLM (Sect. 4.1). To this end, the first strategy we propose is a shooting-based planner, termed **SHOOTING** (see Fig. 2, Left), which takes a single-step approach to maximizing the

**a. Shooting-based planning**

**b. Search-based planning**



**Fig. 2** SHOOTING and GREEDY- SEARCH planning overview. Both SHOOTING and GREEDY- SEARCH planners use the LLM to predict the set of valid goal states given the user's natural language instruction and a description of the current state of the environment. These predicted goals are used to decide when the instruction is satisfied and planning can terminate. **Left:** The SHOOTING strategy uses the LLM to propose full skill sequences first and then runs geometric feasibility planning afterwards. As shown in the experiments, this approach fails when the space of candidate task plans is large but few skill sequences are geometrically feasible. **Right:** In the GREEDY- SEARCH strategy, the LLM is used to propose $K$ candidate skills with the top LLM scores.

The geometric feasibility planner then evaluates the feasibility of each candidate skill, and the one with the highest product of LLM and geometric feasibility scores is selected. The successor state of this skill is predicted by the geometric feasibility planner's dynamics model. If the successor state does not satisfy any of the predicted goals, then it is given to the LLM to plan the next skill. If a goal is satisfied, then the planner returns the skill sequence for execution. By interleaving LLM task planning with geometric feasibility planning at each planning iteration, GREEDY- SEARCH is able to reliably find feasible plans across the different families of tasks we study in the experiments

overall planning objective in Eq. 2. **SHOOTING**'s process is further outlined in Algorithm 1.

**SHOOTING** requires querying the LLM only once to generate $K$ candidate skill sequences $\{\psi_{1:H}^1, \ldots, \psi_{1:H}^K\}$ in an open-ended fashion. Each candidate skill sequence is processed by the geometric feasibility planner which returns an estimate of the sequence's success probability (Eq. 5) and its predicted future state trajectory $s_{2:H+1}$. Skill sequences that satisfy the goal condition (Eq. 6) are added to a candidate set. Invalid skill sequences as determined by Sect. 4.5 are filtered-out of the candidate set. If the candidate set is not empty, **SHOOTING** returns the skill sequence with the highest success probability, or raises a `planning failure` otherwise.

### 4.3 Search-based planning

We propose a second planner, **GREEDY-SEARCH** (see Figure 2, Right), which at each planning iteration ranks candidate skills predicted by the LLM and adds the top scoring skill to the running plan.

This iterative approach can be described as a decomposition of the planning objective in Eq. 2 by timestep $t$:

$$p(\psi_{1:H}, r_{1:H} \mid i, s_1)$$
$$= \prod_{t=1}^{H} p(\psi_t, r_t \mid i, s_1, \psi_{1:t-1}, r_{1:t-1}). \tag{7}$$

We define the joint probability of $\psi_t$ and $r_t$ in Eq. 7 as the skill score $S_{\text{skill}}$:

$$S_{\text{skill}}(\psi_t) = p(\psi_t, r_t \mid i, s_1, \psi_{1:t-1}, r_{1:t-1}),$$

which we factor using conditional probabilities:

$$S_{\text{skill}}(\psi_t) = p(\psi_t \mid i, s_1, \psi_{1:t-1}, r_{1:t-1})$$
$$p(r_t \mid i, s_1, \psi_{1:t}, r_{1:t-1}). \tag{8}$$

Each planning iteration of **GREEDY-SEARCH** is responsible for finding the skill $\psi_t$ that maximizes the skill score (Eq. 8) at timestep $t$.

**Skill usefulness:** The first factor of Eq. 8 captures the *usefulness* of a skill generated by the LLM with respect to satisfying the instruction. We define the skill usefulness score $S_{\text{llm}}$:

$$S_{\text{llm}}(\psi_t) = p(\psi_t \mid i, s_1, \psi_{1:t-1}, r_{1:t-1}) \tag{9}$$
$$\approx p(\psi_t \mid i, s_{1:t}, \psi_{1:t-1}). \tag{10}$$

In Eq. 10, the probability of the next skill $\psi_t$ (Eq. 9) is cast in terms of the predicted state trajectory $s_{2:t}$ of the running plan $\psi_{1:t-1}$, and is thus is independent of prior rewards $r_{1:t-1}$. We refer to "Appendix C.1" for a detailed derivation of Eq. 10.

At each planning iteration $t$, we optimize $S_{\text{llm}}(\psi_t)$ by querying an LLM to generate $K$ candidate skills $\{\psi_t^1, \ldots, \psi_t^K\}$. We then compute the usefulness scores $S_{\text{llm}}(\psi_t^k)$ by summing the token log-probabilities of each skill's language description (visualized in Sect. 5.3). These scores represent the likelihood that $\psi_t^k$ is the correct skill to execute from a language modeling perspective to satisfy instruction $i$.

**Skill feasibility:** The second factor of Eq. 8 captures the *feasibility* of a skill generated by the LLM. We define the skill feasibility score $S_{\text{geo}}$:

$$S_{\text{geo}}(\psi_t) = p(r_t \mid i, s_1, \psi_{1:t}, r_{1:t-1}) \tag{11}$$
$$\approx Q^{\pi_t}(s_t, a_t^*), \tag{12}$$
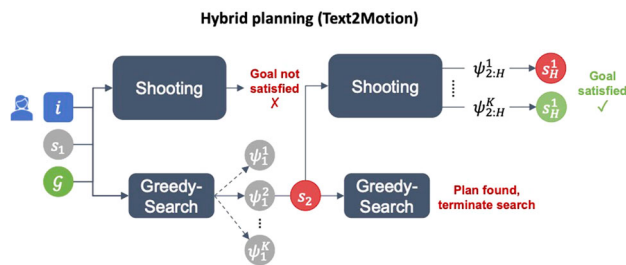
**Hybrid planning (Text2Motion)**

**Fig. 3** Proposed hybrid planner. After predicting goals for a given instruction, Text2Motion iterates the process: (i) invoke SHOOTING to plan full skill sequences, and if no goal-reaching plan is found, (ii) take a GREEDY- SEARCH step and check if executing the selected "best" skill would reach the goal. Note that the entire planning process occurs before execution. See Fig. 2 for a visualization of the SHOOTING and GREEDY- SEARCH planners

where Eq. 12 approximates Eq. 11 by the Q-value evaluated at predicted future state $s_t$ with optimized parameter $a_t^*$, both of which are computed by the geometric feasibility planner. We refer to "Appendix C.2" for a detailed derivation of Eq. 12.

**Skill selection:** The skill feasibility score (Eq. 12) and skill usefulness score (Eq. 10) are then multiplied to produce the overall skill score (Eq. 8) for each of the $K$ candidate skills $\{\psi_t^1, \ldots, \psi_t^K\}$. Invalid skills as determined by Sect. 4.5 are filtered-out of the candidate set. Of the remaining skills, the one with the highest skill score $\psi_t^*$ is added to the running plan $\psi_{1:t-1}$. If the predicted geometric state $s_{t+1}$ that results from skill $\psi_t^*$ satisfies the predicted goal condition (Eq. 6), the skill sequence $\psi_{1:t}$ is returned for execution. Otherwise, $s_{t+1}$ is used to initialize planning iteration $t+1$. The process repeats until the planner returns or a maximum search depth $d_{\max}$ is met raising a `planning failure`. This process is outlined in Algorithm 2.

The baselines we compare to Ahn et al. (2022), Huang et al. (2022) only consider the feasibility of skills $\psi_t^k$ in the current state $s_t$. In contrast, **GREEDY-SEARCH** considers the feasibility of skills $\psi_t^k$ in the context of the planned sequence $\psi_{1:t-1}$ via geometric feasibility planning.

## 4.4 Text2Motion

We present **Text2Motion**, a hybrid planning algorithm that inherits the strengths of both shooting-based and search-based planning strategies. In particular, **SHOOTING** offers efficiency when geometrically feasible skill sequences can be easily predicted by the LLM given the initial state and the instruction. **GREEDY-SEARCH** serves as a reliable fallback strategy that can determine what skills are feasible at the current timestep, should **SHOOTING** fail to find a plan. A visualization is provided in Fig. 3.

At each planning iteration $t$, **Text2Motion** optimistically invokes **SHOOTING** to plan $K$ candidate skill sequences. If **SHOOTING** raises a `planning failure`, then

**Text2Motion** falls back to a single step of **GREEDY-SEARCH**, which adds the skill $\psi_t^*$ with the highest skill score (Eq. 8) to the running plan $\psi_{1:t-1}$. The geometric feasibility planner predicts the state $s_{t+1}$ that would result from executing $\psi_t^*$. If state $s_{t+1}$ satisfies the goal condition (Eq. 6), the skill sequence $\psi_{1:t}$ is returned for execution. Otherwise, the next planning iteration starts by invoking **SHOOTING** on predicted state $s_{t+1}$. The process repeats until the planner returns or a maximum search depth $d_{\max}$ is met. **Text2Motion** is outlined in Algorithm 3.

## 4.5 Out-of-distribution detection

During planning, the LLM may propose skills that are out-of-distribution (OOD) given a state $s_t$ and optimized parameter $a_t^*$. For instance, a symbolically incorrect skill, like Place(dish, table) when the dish is not in hand, may end up being selected if we rely on learned Q-values, since the Q-value for an OOD input can be spuriously high. We therefore reject plans that contain an OOD skill.

We consider a skill $\psi_t$ to be OOD if the variance of its Q-value (Eq. 12) predicted by an ensemble (Lakshminarayanan et al., 2017) exceeds a calibrated threshold $\epsilon^{\psi_t}$:

$$F_{\text{OOD}}(\psi_t) = \mathbb{1}\left( \text{Var}_{i \sim 1:B}\left[ Q_i^{\pi_t}(s_t, a_t^*) \right] \geq \epsilon^{\psi_t} \right), \qquad (13)$$

where $\mathbb{1}$ is the indicator function and $B$ is the ensemble size. We refer to "Appendix A.2" for details on calibrating OOD thresholds $\epsilon^{\psi}$.

## 5 Experiments

We conduct experiments to test four hypotheses:

**H1** Geometric feasibility planning is a necessary ingredient when using LLMs and robot skills to solve manipulation tasks with geometric dependencies from a natural language instruction.
**H2** **GREEDY-SEARCH** is better equipped to solve tasks with partial affordance perception (as defined in Sect. 5.4) compared to **SHOOTING**.
**H3** **Text2Motion**'s hybrid planner inherits the strengths of shooting-based and search-based strategies.
**H4** *A priori* goal prediction is a more reliable plan termination strategy than `stop` scoring.

The following subsections describe the baseline methods we compare against, details on LLMs and prompts, the tasks over which planners are evaluated, and performance metrics we report.
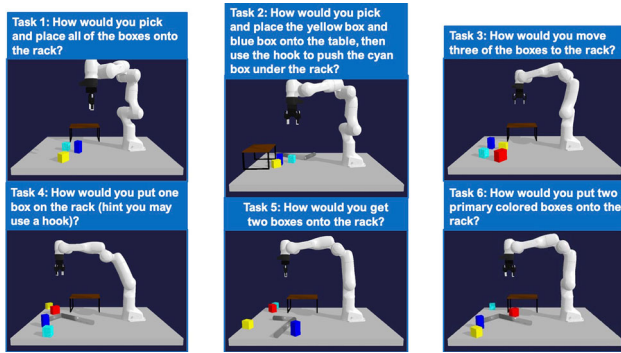
**Fig. 4** TableEnv Manipulation evaluation task suite. We evaluate the performance of all methods on a task suite based on the above manipulation domain. The tasks considered vary in terms of difficulty and each task contains a subset of three properties: being long horizon (Tasks 1, 2, 3, 5, 6), containing lifted goals (Tasks 4, 5, 6), and having partial affordance perception (Tasks 4, 5, 6). During evaluation, we randomize the geometric parameters of each task

## 5.1 Baselines

We compare **Text2Motion** with a series of language-based planners, including the proposed **SHOOTING** and **GREEDY-SEARCH** strategies. For consistency, we use the same skill library $\mathcal{L}^\psi$, with independently trained policies $\pi$ and Q-functions $Q^\pi$, the OOD rejection strategy (Sect. 4.5) and, where appropriate, the dynamics models $T^\pi(s, a)$ and geometric feasibility planner (Sect. 3.3) across all methods and tasks.

**SAYCAN-GS**: We implement a cost-considerate variant of SayCan (Ahn et al., 2022) with a module dubbed **GENERATOR-SCORER** (GS). At each timestep $t$, SayCan ranks *all possible* skills by $p(\psi_t \mid i, \psi_{1:t-1}) \cdot V^{\pi_t}(s_t)$, before executing the top scoring skill (Scorer). However, the cost of ranking skills scales unfavorably with the number of scene objects $\mathcal{O}$ and skills in library $\mathcal{L}^\psi$. **SAYCAN-GS** limits the pool of skills considered in the ranking process by querying the LLM for the $K$ most useful skills $\{\psi_t^1, \ldots, \psi_t^K\} \sim p(\psi_t \mid i, \psi_{1:t-1})$ (Generator) before engaging Scorer. Execution terminates when the score of the `stop` "skill" is larger than the other skills.

**INNERMONO-GS**: We implement the *Object + Scene* variant of Inner Monologue (Huang et al., 2022) by providing task-progress scene context in the form of the environment's symbolic state. We acquire **INNERMONO-GS** by equipping (Huang et al., 2022) with **GENERATOR-SCORER** for cost efficiency. LLM skill likelihoods are equivalent to those from **SAYCAN-GS** except they are now also conditioned on the visited state history $p(\psi_t \mid i, s_{1:t}, \psi_{1:t-1})$.

## 5.2 Large language model

We use two pretrained language models, both of which were accessed through the OpenAI API: (i) `text-davinci-003`,

a variant of the InstructGPT (Ouyang et la., 2022) language model family which is finetuned from GPT-3 with human feedback and (ii) the Codex model (Chen et al., 2021) (specifically, `code-davinci-002`). For the **SHOOTING** planner, we empirically found `text-davinci-003` to be the most capable at open-ended generation of skill sequences. For all other queries, we use `code-davinci-002` as it was found to be reliable. We do not train or finetune the LLMs and only use few shot prompting.

## 5.3 Prompt engineering

The in-context examples are held consistent across all methods and tasks in the prompts passed to the LLM. We provide an example of the prompt structure used to query **GREEDY-SEARCH** for $K = 5$ skills at the first planning iteration (prompt template is in `black` and LLM output is in `orange`):

```
Available scene objects: ['table', 'hook',
'rack', 'yellow box', 'blue box', 'red box']

Object relationships: ['inhand(hook)',
'on(yellow box, table)', 'on(rack, table)',
'on(blue box, table)']

Human instruction: How would you push two of
the boxes to be under the rack?

Goal predicate set: [['under(yellow
box, rack)', 'under(blue box, rack)'],
['under(blue box, rack)', 'under(red
box, rack)'], ['under(yellow box, rack)',
'under(red box, rack)']]

Top 5 next valid robot actions (python list):
['push(yellow box, rack)', 'push(red box,
rack)', 'place(hook, table)', 'place(hook,
rack)', 'pull(red box, hook)']
```

The prompt above chains the output of two queries together: one for goal prediction (Sect. 4.1) and another for skill generation (Sect. 4.3). To compute the skill usefulness (Eq. 10), we replace `Top 5 next valid robot actions` with `Executed action:`, append the language description of the generated skill (e.g. Push(yellow box, rack)), and sum token log-probabilities. We provide the full set of in-context examples in the Appendix ("Appendix B.2").

## 5.4 Task suite

We construct a suite of evaluation tasks (Fig. 4) in a table-top manipulation domain. Each task includes a natural language instruction $i$ and initial state distribution $\rho(s)$ from which geometric task instances are sampled. For the purpose of experimental evaluation only, tasks also contain a ground-truth goal criterion to evaluate whether a plan has satisfied

the corresponding task instruction. Finally, each task contains subsets of the following properties:

- **Long-horizon (LH):** Tasks that require skill sequences $\psi_{1:H}$ of length six or greater to solve. For example, Task 1 in Fig. 4 requires the robot to pick and place three objects for a total of six skills. In our task suite, **LH** tasks also contain geometric dependencies that span across the sequence of skills which are unlikely to be resolved by myopically executing each skill. For example, Task 2 (Fig. 4) requires the robot to pick and place obstructing boxes (i.e. blue and yellow) to enable a collision-free push of the cyan box underneath the rack using the hook.
- **Lifted goals (LG):** Goals are expressed over object classes rather than object instances. For example, the lifted goal instruction "move three boxes to the rack" specifies an object class (i.e. boxes) rather than an object instance (e.g. the red box). This instruction is used for Task 3 (Figure 4). Moreover, **LG** tends to correspond to planning tasks with many possible solutions. For instance, there may only be a single solution to the non-lifted instruction "fetch me the red box and the blue box," but an LLM must contend with more options when asked to, for example, "fetch any two boxes."
- **Partial affordance perception (PAP):** Skill affordances cannot be perceived solely from the spatial relations described in the initial state $s_1$. For instance, Task 5 (Fig. 4) requires the robot to put two boxes onto the rack. However, the scene description obtained through predicate classifiers $\mathcal{L}^\chi$ (described in Sect. 4.1) and the instruction $i$ do not indicate whether it is necessary to use a hook to pull an object closer to the robot first.

## 5.5 Evaluation and metrics

**Text2Motion**, SHOOTING, GREEDY-SEARCH: We evaluate these language planners by marking a plan as successful if, upon execution, they reach a final state $s_{H+1}$ that satisfies the instruction $i$ of a given task. A plan is executed only if the geometric feasibility planner predicts a state that satisfies the inferred goal conditions (Sect. 4.1).

Two failure cases are tracked: (i) planning failure: the method does not produce a sequence of skills $\psi_{1:H}$ whose optimized parameters $a_{1:H}^*$ (Eq. 4) results in a state that satisfies $F_{\text{sat}}^{\mathcal{G}}$ within a maximum plan length of $d_{\max}$; (ii) execution failure: the execution of a plan that satisfies $F_{\text{sat}}^{\mathcal{G}}$ does not achieve the ground-truth goal of the task.

Since the proposed language planners use learned dynamics models to optimize parameters $a_{1:H}$ with respect to (potentially erroneous) future state predictions $s_{2:H}$, we perform the low-level execution of the skill sequence $\psi_{1:H}$ in closed-loop fashion. Thus, upon executing the skill $\psi_t$ at

timestep $t$ and receiving environment feedback $s_{t+1}$, we call STAP (Agia et al., 2022) to perform geometric feasibility planning on the remaining planned skills $\psi_{t+1:H}$. We do not perform task-level replanning, which would involve querying the LLM at timestep $t + 1$ for a new sequence of skills $\psi_{t+1:H}$.

SAYCAN-GS & INNERMONO-GS: These myopic agents execute the next best admissible skill $\psi_t$ at each timestep $t$ without looking-ahead. Hence, we evaluate them in a closed-loop manner for a maximum of $d_{\max}$ steps. We mark a run as a success if the agent issues the `stop` skill and the current state $s_t$ satisfies the ground-truth goal. Note that this comparison is advantageous for these myopic agents because they are given the opportunity to perform closed-loop replanning at the task-level (e.g. re-attempting a failed skill), whereas task-level replanning does not occur for **Text2Motion**, SHOOTING, or GREEDY-SEARCH. This advantage does not lead to measurable performance gains on the challenging evaluation domains that we consider.

**Reported metrics:** We report success rates and subgoal completion rates for all methods. Success rates are averaged over ten random seeds per task, where each seed corresponds to a different geometric instantiation of the task (Sect. 5.4). Subgoal completion rates are computed over all plans by measuring the number of steps an *oracle planner* would take to reach the ground-truth goal from a planner's final state. To further delineate the performance of **Text2Motion** from SHOOTING and GREEDY-SEARCH, we also report the percentages of planning and execution failures.

## 6 Results

### 6.1 Feasibility planning is required to solve tasks with geometric dependencies (H1)

Our first hypothesis is that performing geometric feasibility planning on task plans output by the LLM is essential to task success. To test this hypothesis, we compare methods that use geometric feasibility planning (**Text2Motion**, SHOOTING, GREEDY-SEARCH) against myopic methods that do not (SAYCAN-GS and INNERMONO-GS).

Instructions $i$ provided in the first two planning tasks (**LH**) allude to skill sequences that, if executed appropriately, would solve the task. In effect, the LLM plays a lesser role in contributing to plan success, as its probabilities are conditioned to mimic the skill sequences in $i$. On such tasks, **Text2Motion**, SHOOTING and GREEDY-SEARCH which employ geometric feasibility planning over skills sequences better contend with geometric dependencies prevalent in **LH** tasks and thereby demonstrate higher success rates.
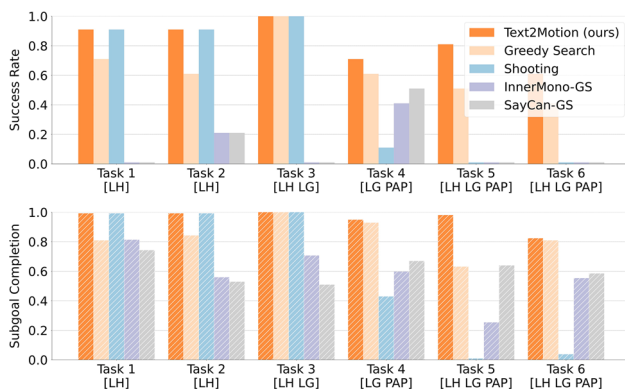
**Fig. 5** Results on the TableEnv manipulation domain with 10 random seeds for each task. **Top:** Our method (Text2Motion) significantly outperforms all baselines on tasks involving partial affordance perception (Task 4, 5, 6). For tasks without partial affordance perception, the methods that use geometric feasibility planning (Text2Motion, SHOOTING, GREEDY- SEARCH) convincingly outperform the methods (SAYCAN-GS and INNERMONO- GS) that do not. We note that SHOOTING performs well on the tasks without partial affordance perception as it has the advantage of outputting *multiple* goal-reaching candidate plans and selecting the one with the highest execution success probability. **Bottom:** Methods without geometric feasibility planning tend to have high sub-goal completion rates but very low success rates. This divergence arises because it is possible to make progress on tasks without resolving geometric dependencies in the earlier timesteps; however, failure to account for geometric dependencies results in failure of the overall task



**Fig. 6** Failure modes of language-based planners on two categories of tasks. In this plot, we analyse the various types of failure modes that occur with Text2Motion, SHOOTING and GREEDY- SEARCH when evaluated on tasks with partial affordance perception (PAP; see Sect. 5.4 for an explanation) and tasks without partial affordance perception (non-PAP). **Top**: For the PAP tasks, SHOOTING incurs many planning failures because the space of possible plans is large but only few can be feasibly executed. In contrast, GREEDY- SEARCH uses value functions during search to narrow down the space of plans to those that are feasible. Text2Motion relies on GREEDY- SEARCH as a fallback if SHOOTING fails, and thus can also contend with PAP tasks. **Bottom**: For the non-PAP tasks, SHOOTING outperforms GREEDY- SEARCH. We attribute this difference to SHOOTING's ability to output multiple task plans while GREEDY- SEARCH can only output a single plan. Finally, Text2Motion matches the performance of SHOOTING as it also outputs and selects among multiple task plans

In contrast, the myopic baselines (SAYCAN-GS and INNERMONO-GS) fail to surpass success rates of 20%, despite completing between 50%-80% of the subgoals (Fig. 5). This result is anticipated as the feasibility of downstream skills requires coordination with earlier skills in the sequence, which these methods do not consider. As the other tasks combine aspects of **LH** with **LG** and **PAP**, it remains difficult for SAYCAN-GS and INNERMONO-GS to find solutions.

Surprisingly, we see that SAYCAN-GS closely matches the performance of INNERMONO-GS, which is additionally provided with descriptions of all states encountered during execution (as opposed to just the initial scene description). This result suggests that explicit language feedback does not contribute to success on our tasks when considered in isolation from plan feasibility.

### 6.2 Search-based reasoning is required for PAP tasks (H2)

Our second hypothesis is that search-based reasoning is required to solve the **PAP** family of tasks (defined in Sect. 5.4). We test this hypothesis by comparing GREEDY-SEARCH and SHOOTING, which represent two distinct approaches to combining symbolic and geometric reasoning to maximize the overall planning objective (Eq. 2). SHOOT-
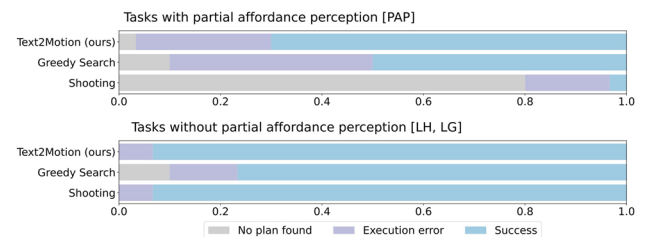
ING uses Q-functions of skills to optimize $K$ skill sequences (Eq. 4) *after* they are generated by the LLM. GREEDY-SEARCH uses Q-functions as skill feasibility heuristics (Eq. 12) to guide search *while* a skill sequence is being constructed.

In the first two tasks (**LH**, Fig. 5), we find that SHOOTING achieves slightly higher success rates than GREEDY-SEARCH, while both methods achieve 100% success rates in the third task (**LH + LG**). This result indicates a subtle advantage of SHOOTING when multiple feasible plans can be directly inferred from $i$ and $s_1$. SHOOTING can capitalize on diverse orderings of $K$ generated skill sequences (including the one specified in $i$) and select the one with the highest success probability (Eq. 3). For example, Task 1 (Fig. 4) asks the robot to put three boxes onto the rack; SHOOTING allows the robot to test multiple different skill sequences while GREEDY-SEARCH only outputs a single plan. This advantage is primarily enabled by bias in the Q-functions: Eq. 5 may indicate that Place(dish, rack) then Place(cup, rack) is more geometrically complex than Place(cup, rack) then Place(dish, rack), while they are geometric equivalents.

The plans considered by GREEDY-SEARCH at planning iteration $t$ share the same sequence of predecessor skills $\psi_{1:t-1}$. This affords limited diversity for the planner to exploit. However, GREEDY-SEARCH has a significant advantage when solving the **PAP** family of problems (Fig. 5, Tasks 4–6). Here, skill sequences with high success probabilities (Eq. 3) are difficult to infer directly from $i$, $s_1$, and the in-context examples provided in the prompt. As a result, SHOOTING incurs an 80% planning failure rate, while

**Table 1** Ablation on hybrid planning method

| Hybrid planning breakdown | Task 4 | Task 5 | Task 6 |
|---|---|---|---|
| % SHOOTING only | 14% | 0% | 0% |
| % GREEDY- SEARCH only | 0% | 0% | 0% |
| % Combination | 86% | 100% | 100% |
| Avg. GREEDY- SEARCH Steps | 1.0 | 2.6 | 3.0 |
| Avg. Plan Length | 5.0 | 7.0 | 7.0 |

We analyze the usage percentages of both SHOOTING and GREEDY-SEARCH in successful plans found by our hybrid planner (see Fig. 3). We find that, as tasks increase in difficulty (Task 4, 5, 6), the majority of solutions involve a combination of both planners. This result indicates that shooting-based and search-based planning strategies play complementing roles in the success of Text2Motion

GREEDY-SEARCH finds plans over 90% of the time (Fig. 6). In terms of success, GREEDY-SEARCH solves 40%-60% of the **PAP** tasks, while **SHOOTING** achieves a 10% success rate on Task 4 (**LG + PAP**) and fails to solve any of the latter two tasks (**LH + LG + PAP**). Moreover, **SHOOTING** does not meaningfully advance on any subgoals, unlike **SAYCAN-GS** and **INNERMONO-GS**, which consider the geometric feasibility of skills at each timestep (albeit, myopically).

### 6.3 Hybrid planning integrates the strengths of search-based and shooting-based methods (H3)

Our third hypothesis is that shooting-based planning and search-based planning have complementing strengths that can be unified in a hybrid planning framework. We test this hypothesis by comparing the performance of **Text2Motion** against **SHOOTING** and **GREEDY-SEARCH**.

The results are presented in Fig. 5. We find that **Text2Motion** matches the performance of **SHOOTING** on tasks that do not consist of **PAP** (Task 1, 2, 3). This is expected because **SHOOTING** does not exhibit planning failures on these tasks (Fig. 6) and **Text2Motion** starts by invoking **SHOOTING**, which results in their identical performance. However, on tasks with **PAP** (Task 4, 5, 6) we observe that **Text2Motion** succeeds more often than **GREEDY-SEARCH**. This suggests that interleaving **SHOOTING** and **GREEDY-SEARCH** at each planning iteration enables **Text2Motion** to consider a more diverse set of goal-reaching solutions. This result is corroborated in Fig. 6, where we see that **Text2Motion** incurs fewer planning and execution failures than **GREEDY-SEARCH**.

In Table 1, we further analyze the usage percentages of **SHOOTING** and **GREEDY-SEARCH** within successful plans executed by **Text2Motion**. The results show that, for tasks involving **PAP**, over 90% of solutions involve a combination of both shooting-based and search-based strategies, which confirms our third hypothesis.
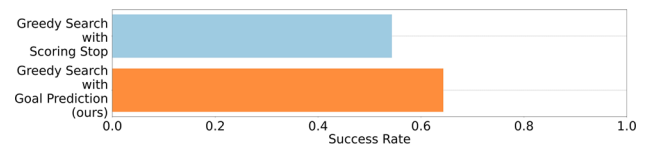


**Fig. 7** Ablation on termination method: goal proposition prediction vs stop scoring. We compare the performance of GREEDY- SEARCH using two different plan termination methods: using the LLM to predict goals *a priori* (ours) and scoring a stop skill (Ahn et al., 2022) during search. We present results averaged across all six tasks and ten seeds for each variation (120 experiments in total). We find that terminating planning when LLM-predicted goals are satisfied results in a 10% boost in success rate over stop scoring

### 6.4 Plan termination is made reliable via goal prediction (H4)

Our fourth hypothesis is that predicting goals from instructions *a priori* and selecting plans based on their satisfaction (Sect. 4.1) is more reliable than scoring plan termination with a dedicated stop skill at each timestep. We test this hypothesis in an ablation experiment (Fig. 7), comparing our plan termination method to that of SayCan and Inner Monologue's, while keeping all else constant for our **GREEDY-SEARCH** planner. We run 120 experiments (two variations, six tasks, and ten seeds each) in total on the TableEnv Manipulation task suite. The results in Fig. 7 suggest that, for the tasks we consider, our proposed goal prediction method leads to 10% higher success rates than the scoring baseline.

We also note the apparent advantages of both techniques. First, goal prediction is more efficient than scoring stop as the former requires only one LLM query, whereas the latter needs to be queried at every timestep. Second, goal prediction offers interpretability over stop scoring, as it is possible to inspect the goal that the planner is aiming towards prior to execution. Nonetheless, stop scoring does provide benefits in terms of expressiveness, as its predictions are not constrained to any specific output format. This advantage, however, is not captured in our evaluation task suite, which at most require conjunctive ($\wedge$) and disjunctive ($\vee$) goals. For instance, "Stock two boxes onto the rack" could correspond to (on(red box, rack) $\wedge$ on(blue box, rack)) $\vee$ (on(yellow box, rack) $\wedge$ on(cyan box, rack)), while in theory, stop scoring can represent all goals expressible in language.

## 7 Limitations and future work

**LLM likelihoods:** We observed an undesirable pattern emerge in the planning phase of **GREEDY-SEARCH** and the execution phase of **SAYCAN-GS** and **INNERMONO-GS**, where *recency bias* (Zhao et al., 2021) would cause the LLM to produce unreliable likelihoods (Eq. 10), inducing a cyclic

state of repeating feasible skills. While we mitigate such failures by combining **GREEDY-SEARCH** and **SHOOTING** in the hybrid **Text2Motion** algorithm, leveraging calibration techniques to increase the reliability LLM likelihoods (Li et al., 2022; Chen et al., 2022b) may improve the performance search-based planning over long-horizons.

**Skill library:** As with other methods that use skill libraries, **Text2Motion** is reliant on the fidelity of the learned skills, their value functions, and the ability to accurately predicted future states with dynamics models. Thus, incorporating skills that operate on high-dimensional observations [e.g. images (Shao et al., 2021)] into our framework may require adopting techniques for stable long-term predictions in these observation spaces (Driess et al., 2023).

**Runtime complexity: Text2Motion** is mainly comprised of learned components, which comes with its associated efficiency benefits. Nonetheless, runtime complexity was not a core focus of this work, and STAP (Agia et al., 2022) was frequently called during planning, increasing the overall planning time. LLMs also impose an inference bottleneck as each API query (Sect. 5.2) requires 2–10 seconds, but we anticipate improvements with advances in both LLM inference techniques and in methods that distill LLM capabilities into smaller, cost-efficient models (Touvron et al., 2023).

**Closed-world assumptions:** Our framework assumes a closed-world setting (Sect. 3), where we assume to know which objects are task-relevant, and the poses of objects are estimated by an external perception system at the time of receiving a language instruction. Extending **Text2Motion** to open-world settings may necessitate exploring (Chen et al., 2022a) or interacting (Curtis et al., 2022) with the environment to discover objects that are not initially observable, and training skills to support a diverse set of real-world objects (Jang et al., 2021; Kalashnkov et al., 2021).

**Future Work:** We outline several avenues for future work based on these observations. First, there remains an opportunity to increase the plan-time efficiency of our method, for instance, by warm starting geometric feasibility planning with solutions cached in earlier planning iterations (Williams et al., 2015). Second, we aim to explore the use of Visual Question and Answering (VQA) (Zhou et al., 2020) and multi-modal foundation models that are visually grounded (Driess et al., 2023; OpenAI, 2023). Such models may support scaling **Text2Motion** to higher-dimensional observation spaces and potentially serve as a substitutes for closed-world components used in our framework (e.g. detecting a variable number of predicates using VQA). Lastly, we hope to leverage **Text2Motion** as part of a broader planning system enroute to the goal of open-world operability. Such a system could, for example, use Text2Motion to produce feasible and verified plans to subgoals, while building knowledge of the environment in unobserved or partially observable regions during the execution of those subgoals.

Progress on each of these fronts would constitute steps in the direction of scalable, reliable, and real-time language planning capabilities.

## 8 Conclusion

We present a language-based planning framework that combines LLMs, learned skills, and geometric feasibility planning to solve long-horizon robotic manipulation tasks containing geometric dependencies. **Text2Motion** constructs a task- and motion-level plan and verifies that it satisfies a natural language instruction by testing planned states against inferred goals. In contrast to prior language planners, our method verifies that its plan satisfies the instruction before executing any actions in the environment. **Text2Motion** represents a hybrid planning formalism that optimistically queries an LLM for long-horizon plans and falls back to a reliable search strategy should optimistic planning fail. As a result, **Text2Motion** inherits the strengths of both shooting-based and search-based planning formalisms.

Our results highlight the following: (i) geometric feasibility planning is important when using LLMs and learned skills to solve sequential manipulation tasks from natural language instructions; (ii) search-based reasoning can contend with a family of tasks where the space of possible plans is large but only few are feasible; (iii) shooting-based and search-based planning strategies can be synergistically integrated in a hybrid planner that outperforms its constituent parts; (iv) terminating plans based on inferred symbolic goals is more reliable than prior LLM scoring techniques.

## 9 Usage of LLMs

Beyond the technical application of LLMs for robotic planning discussed in the paper, the authors used ChatGPT to brainstorm wording and phrasing choices by directly asking questions to the LLM via the OpenAI interface, but otherwise wrote the paper themselves. 2-10 seconds,

## Appendix A Implementation details

The **Text2Motion** planner integrates both **SHOOTING** and **GREEDY-SEARCH** to construct skill sequences that are feasible for the robot to execute in the environment. The planning procedure relies on four core components: (1) a library of learned robot skills, (2) a method for detecting when a skill is out-of-distribution (OOD), (3) a large language model (LLM) to perform task-level planning, and (4) a geometric feasibility planner that is compatible with the learned robot skills. All evaluated language-based planners use the above components, while **SAYCAN-GS** and **INNERMONO-GS** are myopic agents that do not perform geometric feasibility planning. We provide implementation details of these components in the following subsections.

### A.1 Learning robot skills and dynamics

**Skill library overview:** All evaluated language planners interface an LLM with a library of robot skills $\mathcal{L} = \{\psi^1, \ldots, \psi^N\}$. Each skill $\psi$ has a language description (e.g. Pick(a)) and is associated with a parameterized manipulation primitive (Felip et al., 2013) $\phi(a)$. A primitive $\phi(a)$ is controllable via its *parameter* $a$ which determines the motion (Khatib, 1987) of the robot's end-effector through a series of waypoints. For each skill $\psi$, we train a policy $\pi(a|s)$ to output parameters $a \in \mathcal{A}$ that maximize primitive's $\phi(a)$ probability of success in a contextual bandit setting (Eq. 1) with a skill-specific binary reward function $R(s, a, s')$. We also train an ensemble of Q-functions $Q_{1:B}^\pi(s, a)$ and a dynamics model $T^\pi(s'|s, a)$ for each skill, both of which are required for geometric feasibility planning. We discuss the calibration of Q-function ensembles for OOD detection of skills in "Appendix A.2".

We learn four manipulation skills to solve tasks in simulation and in the real-world: $\psi^{\text{Pick}}$, $\psi^{\text{Place}}$, $\psi^{\text{Pull}}$, $\psi^{\text{Push}}$. Only a single policy per skill is trained, and thus, the policy must learn to engage the primitive over objects with differing geometries (e.g. $\pi^{\text{Pick}}$ is used for both Pick(box) and Pick(hook)). The state space $\mathcal{S}$ for each policy is defined as the concatenation of geometric state features (e.g. pose, size) of all objects in the scene, where the first $n$ object states correspond to the $n$ skill arguments and the rest are randomized. For example, the state for the skill Pick(hook) would have be a vector of all objects' geometric state features with the first component of the state corresponding to the hook.

**Parameterized manipulation primitives:** We describe the parameters $a$ and reward function $R(s, a, s')$ of each parameterized manipulation primitive $\phi(a)$ below. A collision with a non-argument object constitutes an execution failure for all skills, and as a result, the policy receives a reward of 0. For example, $\pi^{\text{Pick}}$ would receive a reward of 0 if the robot collided with a box during the execution of Pick(hook).

- Pick(obj): $a \sim \pi^{\text{Pick}}(a|s)$ denotes the grasp pose of obj *w.r.t* the coordinate frame of obj. A reward of 1 is received if the robot successfully grasps obj.
- Place(obj, rec): $a \sim \pi^{\text{Place}}(a|s)$ denotes the placement pose of obj *w.r.t* the coordinate frame of rec. A reward of 1 is received if obj is stably placed on rec.
- Pull(obj, tool): $a \sim \pi^{\text{Pull}}(a|s)$ denotes the initial position, direction, and distance of a pull on obj with tool *w.r.t* the coordinate frame of obj. A reward of 1 is received if obj moves toward the robot by a minimum of $d_{\text{Pull}} = 0.05m$.
- Push(obj, tool, rec): $a \sim \pi^{\text{Push}}(a|s)$ denotes the initial position, direction, and distance of a push on obj with tool *w.r.t* the coordinate frame of obj. A reward of 1 is received if obj moves away from the robot by a minimum of $d_{\text{Push}} = 0.05m$ and if obj ends up underneath rec.

**Dataset generation:** All planners considered in this work rely on accurate Q-functions $Q^\pi(s, a)$ to estimate the feasibility of skills proposed by the LLM. This places a higher fidelity requirement on the Q-functions than typically needed to learn a reliable policy, as the Q-functions must characterize both skill success (feasibility) and failure (infeasibility) at a given state. Because the primitives $\phi(a)$ reduce the horizon of policies $\pi(a|s)$ to a single timestep, and the reward functions are $R(s, a, s') \in \{0, 1\}$, the Q-functions can be interpreted as binary classifiers of state-action pairs. Thus, we take a staged approach to learning the Q-functions $Q^\pi$, followed by the policies $\pi$, and lastly the dynamics models $T^\pi$.

Scenes in our simulated environment are instantiated from a symbolic specification of objects and spatial relations, which together form a symbolic state. The goal is to learn a Q-function that sufficiently covers the state-action space of each skill. We generate a dataset that meets this requirement in four steps: a) enumerate all valid symbolic states; b) sample geometric scene instances $s$ per symbolic state; c) uniformly sample actions over the action space $a \sim \mathcal{U}^{[0,1]^d}$; (d) simulate the states and actions to acquire next states $s'$ and

compute rewards $R(s, a, s')$. We slightly modify this sampling strategy to maintain a minimum success-failure ratio of 40%, as uniform sampling for more challenging skills such as Pull and Push seldom emits a success ($\sim$3%). We collect 1M $(s, a, s', r)$ tuples per skill of which 800K of them are used for training ($\mathcal{D}_t$), while the remaining 200K are used for validation ($\mathcal{D}_v$). We use the same datasets to learn the Q-functions $Q^\pi$, policies $\pi$, and dynamics models $T^\pi$ for each skill.

**Model training:** We train an ensemble of Q-functions with mini-batch gradient descent and logistic regression loss. Once the Q-functions have converged, we distill their returns into stochastic policies $\pi$ through the maximum-entropy update (Haarnoja et al., 2018):

$$\pi^* \leftarrow \arg\max_\pi \ \mathrm{E}_{(s,a)\sim\mathcal{D}_t}[\min(Q_{1:B}^\pi(s, a))$$
$$-\alpha \log \pi(a|s)].$$

Instead of evaluating the policies on $\mathcal{D}_v$, which contains states for which no feasible action exists, the policies are synchronously evaluated in an environment that exhibits only feasible states. This simplifies model selection and standardizes skill capabilities across primitives. All Q-functions achieve precision and recall rates of over 95%. The average success rates of the converged policies over 100 evaluation episodes are: $\pi_{\mathrm{Pick}}$ with 99%, $\pi_{\mathrm{Place}}$ with 90%, $\pi_{\mathrm{Pull}}$ with 86%, $\pi_{\mathrm{Push}}$ with 97%.

We train a deterministic dynamics model per skill using the forward prediction loss:

$$L_{\mathrm{dynamics}}\left(T^\pi; \mathcal{D}_t\right) = \mathrm{E}_{(s,a,s')\sim\mathcal{D}_t} \|T^\pi(s, a) - s'\|_2^2.$$

The dynamics models converge to within millimeter accuracy on the validation split.

**Hyperparameters:** The Q-functions, policies, and dynamics models are MLPs with hidden dimensions of size [256, 256] and ReLU activations. We train an ensemble of $B = 8$ Q-functions with a batch size of 128 and a learning rate of $1e^{-4}$ with a cosine annealing decay (Loshchilov and Hutter, 2017). The Q-functions for Pick, Pull, and Push converged on $\mathcal{D}_v$ in 3M iterations, while the Q-function for Place required 5M iterations. We hypothesize that this is because classifying successful placements demands carefully attending to the poses and shapes of all objects in the scene so as to avoid collisions. The policies are trained for 250K iterations with a batch size of 128 and a learning rate of $1e^{-4}$, leaving all other parameters the same as Haarnoja et al. (2018). The dynamics models are trained for 750K iterations with a batch size of 512 and a learning rate of $5e^{-4}$; only on successful transitions to avoid the noise associated with collisions and truncated episodes. The parallelized training of all models takes approximately 12 hours on an Nvidia Quadro P5000 GPU and 2 CPUs per job.

## A.2 Out-of-distribution detection

The training dataset described in Sect. 1 contain both successes and failures for symbolically valid skills like Pick(box). However, when using LLMs for robot task planning, it is often the case that the LLM will propose symbolically invalid skills, such as Pick(table), that neither the skill's policy, Q-functions, or dynamics model have observed in training. We found that a percentage of out-of-distribution (OOD) queries would result in erroneously high Q-values, causing the invalid skill to be selected. Attempting to execute such a skill leads to control exceptions or other failures.

Whilst there are many existing techniques for OOD detection of deep neural networks, we opt to detect OOD queries on the learned Q-functions via deep ensembles due to their ease of calibration (Lakshminarayanan et al., 2017). A state-action pair is classified as OOD if the empirical variance of the predicted Q-values is above a determined threshold:

$$F_{\mathrm{OOD}}(\psi) = \mathbb{1}\left(\mathrm{Var}_{i\sim1:B}\left[Q_i^\pi(s, a)\right] \geq \epsilon^\psi\right),$$

where each threshold $\epsilon^\psi$ is unique to skill $\psi$.

To determine the threshold values, we generate a calibration dataset of 100K symbolically invalid states and actions for each skill. The process takes less than an hour on a single CPU as the actions are infeasible and need not be simulated in the environment (i.e. rewards are known to be 0). We compute the mean and variance of the Q-ensemble for each $(s, a)$ sample in both the training dataset (in-distribution inputs) and the calibration dataset (out-of-distribution inputs), and produce two histograms by binning the computed ensemble variances by the ensemble means. We observe that the histogram of variances corresponding to OOD inputs is uniform across all Q-value bins and is an order of magnitude large than the ensemble variances computed over in-distribution inputs. This allows us to select thresholds $\epsilon^\psi$ which are low enough to reliably detect OOD inputs, yet will not be triggered for in-distribution inputs: $\epsilon^{\mathrm{Pick}} = 0.10$, $\epsilon^{\mathrm{Place}} = 0.12$, $\epsilon^{\mathrm{Pull}} = 0.10$, and $\epsilon^{\mathrm{Push}} = 0.06$.

## A.3 Task planning with LLMs

**Text2Motion**, GREEDY-SEARCH, and the myopic planning baselines SAYCAN-GS and INNERMONO-GS use `code-davinci-002` (Chen et al., 2021) to generate and score skills, while **SHOOTING** queries `text-davinci-003` (Ouyang et la., 2022) to directly output full skill sequences. In our experiments, we used a temperature setting of 0 for all LLM queries.

To maintain consistency in the evaluation of various planners, we allow **Text2Motion**, SAYCAN-GS, and INNERMONO-GS to generate $K = 5$ skills $\{\psi_t^1, \ldots, \psi_t^K\}$ at each timestep $t$. Thus, every search iteration of GREEDY-

**SEARCH** considers five possible extensions to the current running sequence of skills $\psi_{1:t-1}$. Similarly, **SHOOTING** generates $K = 5$ skill sequences.

As described in Sect. 4.3, skills are selected at each timestep $t$ via a combined usefulness and geometric feasibility score:

$$S_{\text{skill}}(\psi_t) = S_{\text{llm}}(\psi_t) \cdot S_{\text{geo}}(\psi_t)$$
$$\approx p(\psi_t \mid i, s_{1:t}, \psi_{1:t-1}) \cdot Q^{\pi_t}(s_t, a_t^*),$$

where **Text2Motion**, **GREEDY-SEARCH**, and **SHOOTING** use geometric feasibility planning (details below in "Appendix A.4") to compute $S_{\text{geo}}(\psi_t)$, while **SAYCAN-GS** and **INNERMONO-GS** use the current value function estimate $V^{\pi_t}(s_t) = \mathrm{E}_{a_t \sim \pi_t}[Q^{\pi_t}(s_t, a_t)]$. We find that in both cases, taking $S_{\text{llm}}(\psi_t)$ to be the SoftMax log-probability score produces a winner-takes-all effect, causing the planner to omit highly feasible skills simply because their associated log-probability was marginally lower than the LLM-likelihood of another skill. Thus, we dampen the SoftMax operation with a $\beta$-coefficient to balance the ranking of skills based on both feasibility and usefulness. We found $\beta = 0.3$ to work well.

## A.4 Geometric feasibility planning

Given a sequence of skills $\psi_{1:H}$, geometric feasibility planning computes parameters $a_{1:H}$ that maximizes the success probability of the underlying sequence of primitives $\phi_{1:H}$. For example, given a skill sequence Pick(hook), Pull(box, hook), geometric feasibility planning would compute a 3D grasp position on the hook that enables a successful pull on the box thereafter.

**Text2Motion** is agnostic to the method that fulfils the role of geometric feasibility planning. In our experiments we leverage Sequencing Task-Agnostic Policies (STAP) (Agia et al., 2022). Specifically, we consider the PolicyCEM variant of STAP, where optimization of the skill sequence's success probability (Eq. 4) is warm started with parameters sampled from the policies $a_{1:H} \sim \pi_{1:H}$. We perform ten iterations of the Cross-Entropy Method (Rubinstein, 1999), sampling 10K trajectories at each iteration and selecting 10 elites to update the mean of the sampling distribution for the following iteration. The standard deviation of the sampling distribution is held constant at 0.3 for all iterations.

## A. 5 Algorithms

In this section, we provide algorithm pseudocode for all language-based planners proposed in this work: **SHOOTING** in Algorithm 1, **GREEDY-SEARCH** in Algorithm 2, and **Text2Motion** in Algorithm 3.

---

**Algorithm 1** Shooting-based LLM planner

1: **globals:** $\mathcal{L}^\psi$, $\mathcal{L}^\chi$, SATFUNC, LLM, STAP
2: **function** SHOOTING($i, s_1, \mathcal{G}; K$)
3:    $F_{\text{sat}}^{\mathcal{G}} \leftarrow$ SATFUNC($\mathcal{G}, \mathcal{L}^\chi$)    ▷ Goal checker
4:    $\{\psi_{1:H}^{(j)}\}_{j=1}^K \leftarrow$ LLM($i, s_1, \mathcal{G}, K$)    ▷ Gen. plans
5:    $C = \{\}$    ▷ Init. candidate set
6:    **for** $j = 1 \ldots K$ **do**
7:       $s_{2:H+1}^{(j)}, a_{1:H}^{(j)} \leftarrow$ STAP($s_1, \psi_{1:H}^{(j)}, \mathcal{L}^\psi$)
8:       **if** $F_{\text{sat}}^{\mathcal{G}}(s_t^{(j)}) == 1$ **for** $t \le H+1$ **then**
9:          $\psi_{1:t-1}^{(j)} \leftarrow \psi_{1:H}^{(j)}[:t-1]$    ▷ Slice plan
10:          $C \leftarrow C \cup \{j\}$    ▷ Add to candidate set
11:       **end if**
12:       Compute $p_{\text{success}}^{(j)}$ via Eq. 5
13:    **end for**
14:    Filter OOD plans from $C$ as per Eq. 13
15:    **if** $C == \emptyset$ **then**
16:       **raise** planning failure
17:    **end if**
18:    $j^* = \arg\max_{j \in C} p_{\text{success}}^{(j)}$
19:    **return** $\psi_{1:t-1}^{(j^*)}$    ▷ Return best plan
20: **end function**

---

**Algorithm 2** Search-based LLM planner

1: **globals:** $\mathcal{L}^\psi$, $\mathcal{L}^\chi$, SATFUNC, LLM, STAP
2: **function** GREEDY- SEARCH($i, s_1, \mathcal{G}; K, d_{\max}$)
3:    $F_{\text{sat}}^{\mathcal{G}} \leftarrow$ SATFUNC($\mathcal{G}, \mathcal{L}^\chi$)    ▷ Goal checker
4:    $\Psi = [\,]; \tau = [s_1]$    ▷ Init. running plan
5:    **while** $len(\Psi) < d_{\max}$ **do**
6:       $\Psi, \tau \leftarrow$ GREEDY- STEP($i, s_1, \mathcal{G}, \Psi, \tau, K$)
7:       **if** $F_{\text{sat}}^{\mathcal{G}}(\tau[-1]) == 1$ **then**
8:          **return** $\Psi$    ▷ Return goal-reaching plan
9:       **end if**
10:    **end while**
11:    **raise** planning failure
12: **end function**
13: **function** GREEDY- STEP($i, s_1, \mathcal{G}, \Psi, \tau; K$)
14:    $t = len(\Psi) + 1$    ▷ Curr. planning iteration
15:    $\{\psi_t^{(j)}\}_{j=1}^K \leftarrow$ LLM($i, \tau, \mathcal{G}, K$)    ▷ Gen. skills
16:    $C = \{\}$    ▷ Init. candidate set
17:    **for** $j = 1 \ldots K$ **do**
18:       $\psi_{1:t}^{(j)} \leftarrow \Psi.append(\psi^{(j)})$
19:       $s_{2:t+1}^{(j)}, a_{1:t}^{(j)} \leftarrow$ STAP($s_1, \psi_{1:t}^{(j)}, \mathcal{L}^\psi$)
20:       Compute $S_{\text{llm}}(\psi_t^{(j)})$ via Eq.10
21:       Compute $S_{\text{geo}}(\psi_t^{(j)})$ via Eq.12
22:       $S_{\text{skill}}(\psi_t^{(j)}) \leftarrow S_{\text{llm}}(\psi_t^{(j)}) \times S_{\text{geo}}(\psi_t^{(j)})$
23:       **if** $\psi_t^{(j)}$ is not OOD **then**    ▷ As per Eq. 13
24:          $C \leftarrow C \cup \{j\}$    ▷ Add to candidate set
25:       **end if**
26:    **end for**
27:    $j^* = \arg\max_{j \in C} S_{\text{skill}}(\psi_t^{(j)})$
28:    **return** $\psi_{1:t}^{(j^*)}, s_{1:t+1}^{(j^*)}$    ▷ Return running plan
29: **end function**

**Algorithm 3** Text2Motion hybrid planner

```
1: globals: ℒ^χ, SATFUNC, SHOOTING, GREEDY- STEP
2: function TEXT2MOTION(i, s₁, 𝒢; K, d_max)
3:     F_sat^𝒢 ← SATFUNC(𝒢, ℒ^χ)                    ▷ Goal checker
4:     Ψ = [ ]; τ = [s₁]                           ▷ Init. running plan
5:     while len(Ψ) < d_max do
6:         try
7:             return SHOOTING(i, τ, 𝒢, K)
8:         catch planning failure
9:             Ψ, τ ← GREEDY- STEP(i, s₁, 𝒢, Ψ, τ, K)
10:            if F_sat^𝒢(τ[−1]) == 1 then
11:                return Ψ
12:            end if
13:        end try
14:    end while
15:    raise planning failure
16: end function
```

## Appendix B Experiment details

We refer to Table 2 for an overview of the tasks in the TableEnv Manipulation suite.

### B.1 Scene descriptions as symbolic states

For the remainder of this section, we use the following definitions of terms:

- **Predicate:** a binary-valued function over objects that evaluates to true or false (e.g. on(a, b))
- **Spatial Relation:** a predicate grounded over objects that evaluates to true (e.g. on(rack, table))
- **Predicate Classifier:** a function that implements whether a predicate is true or false in the scene. In this work, we use hand-crafted predicate classifiers for each spatial relation we model
- **Symbolic State:** the set of all predicates that hold true in the scene
- **Satisfaction Function:** a binary-valued function that takes as input a geometric state, uses the predicate classifiers to detect what predicates hold true in the geometric state, and collects those predicates into a set to form a

symbolic state. The satisfaction function evaluates to true if the predicted goals (predicates) hold in the symbolic state

To provide scene context to **Text2Motion** and the baselines, we take a heuristic approach to converting a geometric state $s$ into a basic symbolic state. Symbolic states consist of combinations of one or more of the following predicates that have been grounded: on(a, b), under(a, b), and inhand(a). inhand(a) = True when the height of object a is above a predefined threshold. on(a, b) = True when (i) object a is above b (determined by checking if the centroid of a's axis-aligned bounding box is greater than b's axis-aligned bounding box), (ii) a's bounding box intersects b's bounding box, and (iii) inhand(a) = False. under(a, b) = True when on(a, b) = False and a's bounding box intersects b's bounding box.

The proposed **goal proposition prediction technique** outputs goal propositions consisting of combinations of the three spatial relations above. As an example, for the natural language instruction "Put two of the boxes under the rack" and a symbolic state `[on(red box, table), on(green box, rack), on(hook, rack), on(blue box, rack)]`, the LLM might predict the set of three goals `{[under(red box, rack), under(blue box, rack)], [under(red box, rack), under(green box, rack)], [under(green box, rack), under(blue box, rack)]}`. We highlight that objects are neither specified as within or beyond the robot workspace, as we leave it to the skill's Q-functions to determine feasibility (Sect. 1).

Since planning in high-dimensional observation spaces is not the focus of this work, we assume knowledge of objects in the scene and use hand-crafted heuristics to detect spatial relations between objects. There exists several techniques to convert high-dimensional observations into scene descriptions, such as the one used in Zeng et al. (2022). We leave exploration of these options to future work.

**Table 2** TableEnv manipulation task suite

| Task ID | Properties | Instruction |
| --- | --- | --- |
| Task 1 | LH | How would you pick and place all of the boxes onto the rack?" |
| Task 2 | LH + LG | How would you pick and place the yellow box and blue box onto the table then use the hook to push the cyan box under the rack?" |
| Task 3 | LH + PAP | How would you move three of the boxes to the rack?" |
| Task 4 | LG + PAP | How would you put one box on the rack?" |
| Task 5 | LH + LG + PAP | How would you get two boxes onto the rack?" |
| Task 6 | LH + LG + PAP | How would you move two primary colored boxes to the rack?" |

We use the following shorthands as defined in the paper: LH: Long-Horizon, LG: Lifted Goals, PAP: Partial Affordance Perception

## B.2 In-context examples

For all experiments and methods, we use the following in-context examples to construct the prompt passed to the LLMs.

```
Available scene objects: ['table', 'hook',
'rack', 'yellow box', 'blue box', 'red box']
Object relationships: ['inhand(hook)',
'on(yellow box, table)', 'on(rack, table)',
'on(blue box, table)']
Human instruction: How would you push two of
the boxes to be under the rack?
Goal predicate set: [['under(yellow
box, rack)', 'under(blue box, rack)'],
['under(blue box, rack)', 'under(red
box, rack)'], ['under(yellow box, rack)',
'under(red box, rack)']]
Top 1 robot action sequences: ['push(yellow
box, hook, rack)', 'push(red box, hook,
rack)']
```

```
Available scene objects: ['table', 'cyan
box', 'hook', 'blue box', 'rack', 'red box']
Object relationships: ['on(hook, table)',
'on(rack, table)', 'on(blue box, table)',
'on(cyan box, table)', 'on(red box, table)']
Human instruction: How would you push all the
boxes under the rack?
Goal predicate set: [['under(blue box,
rack)', 'under(cyan box, rack)', 'under(red
box, rack)']]
Top 1 robot action sequences: ['pick(blue
box)', 'place(blue box, table)',
'pick(hook)', 'push(cyan box, hook, rack)',
'place(hook, table)', 'pick(blue box)',
'place(blue box, table)', 'pick(hook)',
'push(blue box, hook, rack)', 'push(red box,
hook, rack)']
```

```
Available scene objects: ['table', 'cyan
box', 'red box', 'hook', 'rack']
Object relationships: ['on(hook, table)',
'on(rack, table)', 'on(cyan box, rack)',
'on(red box, rack)']
Human instruction: put the hook on the rack
and stack the cyan box above the rack -
thanks
Goal predicate set: [['on(hook, rack)',
'on(cyan box, rack)']]
Top 1 robot action sequences: ['pick(hook)',
'pull(cyan box, hook)', 'place(hook, rack)',
'pick(cyan box)', 'place(cyan box, rack)']
```

```
Available scene objects: ['table', 'rack',
'hook', 'cyan box', 'yellow box', 'red box']
Object relationships: ['on(yellow box,
table)', 'on(rack, table)', 'on(cyan box,
table)', 'on(hook, table)', 'on(red box,
rack)']
Human instruction: Pick up any box.
Goal predicate set: [['inhand(yellow box)'],
['inhand(cyan box)']]
Top 1 robot action sequences: ['pick(yellow
box)']
```

```
Available scene objects: ['table', 'blue
box', 'cyan box', 'hook', 'rack', 'red box',
'yellow box']
Object relationships: ['inhand(hook)',
'on(red box, rack)', 'on(yellow box, table)',
'on(blue box, table)', 'on(cyan box, rack)',
'on(rack, table)']
Human instruction: could you move all the
boxes onto the rack?
Goal predicate set: [['on(yellow box, rack)',
'on(blue box, rack)']]
Top 1 robot action sequences: ['pull(yellow
box, hook)', 'place(hook, table)',
'pick(yellow box)', 'place(yellow box,
rack)', 'pick(blue box)', 'place(blue box,
rack)']
```

```
Available scene objects: ['table', 'blue
box', 'red box', 'hook', 'rack', 'yellow
box']
Object relationships: ['on(hook, table)',
'on(blue box, table)', 'on(rack, table)',
'on(red box, table)', 'on(yellow box,
table)']
Human instruction: situate an odd number
greater than 1 of the boxes above the rack
Goal predicate set: [['on(blue box, rack)',
'on(red box, rack)', 'on(yellow box, rack)']]
Top 1 robot action sequences: ['pick(hook)',
'pull(blue box, hook)', 'place(hook, table)',
'pick(blue box)', 'place(blue box, rack)',
'pick(red box)', 'place(red box, rack)',
'pick(yellow box)', 'place(yellow box,
rack)']
```

```
Available scene objects: ['table', 'cyan
box', 'hook', 'red box', 'yellow box',
'rack', 'blue box']
Object relationships: ['on(hook, table)',
'on(red box, table)', 'on(blue box, table)',
'on(cyan box, table)', 'on(rack, table)',
'under(yellow box, rack)']
Human instruction: How would you get the cyan
box under the rack and then ensure the hook
is on the table?
Goal predicate set: [['under(cyan box,
rack)', 'on(hook, table)']]
Top 1 robot action sequences: ['pick(blue
box)', 'place(blue box, table)', 'pick(red
box)', 'place(red box, table)', 'pick(hook)',
'push(cyan box, hook, rack)', 'place(hook,
table)']
```

```
Available scene objects: ['table', 'cyan
box', 'hook', 'yellow box', 'blue box',
'rack']
Object relationships: ['on(hook, table)',
'on(yellow box, rack)', 'on(rack, table)',
'on(cyan box, rack)']
Human instruction: set the hook on the rack
and stack the yellow box onto the table and
set the cyan box on the rack
Goal predicate set: [['on(hook, rack)',
'on(yellow box, table)', 'on(cyan box,
rack)']]
Top 1 robot action sequences: ['pick(yellow
box)', 'place(yellow box, table)',
'pick(hook)', 'pull(yellow box, hook)',
'place(hook, table)']
```

```
Available scene objects: ['table', 'cyan
box', 'hook', 'rack', 'red box', 'blue box']
Object relationships: ['on(hook, table)',
'on(blue box, rack)', 'on(cyan box, table)',
'on(red box, table)', 'on(rack, table)']
Human instruction: Move the warm colored box
to be underneath the rack.
Goal predicate set: [['under(red box,
rack)']]
Top 1 robot action sequences: ['pick(blue
box)', 'place(blue box, table)', 'pick(red
box)', 'place(red box, table)', 'pick(hook)',
'push(red box, hook, rack)']
```

```
Available scene objects: ['table', 'blue
box', 'hook', 'rack', 'red box', 'yellow
box']
Object relationships: ['on(hook, table)',
'on(red box, table)', 'on(blue box, table)',
'on(yellow box, rack)', 'on(rack, table)']
Human instruction: Move the ocean colored box
to be under the rack and ensure the hook ends
up on the table.
Goal predicate set: [['under(blue box,
rack)']]
Top 1 robot action sequences: ['pick(red
box)', 'place(red box, table)', 'pick(yellow
box)', 'place(yellow box, rack)',
'pick(hook)', 'push(blue box, hook, rack)',
'place(hook, table)']
```

```
Available scene objects: ['table', 'cyan
box', 'hook', 'rack', 'red box', 'blue box']
Object relationships: ['on(hook, table)',
'on(cyan box, rack)', 'on(rack, table)',
'on(red box, table)', 'inhand(blue box)']
Human instruction: How would you set the red
box to be the only box on the rack?
Goal predicate set: [['on(red box, rack)',
'on(blue box, table)', 'on(cyan box,
table)']]
Top 1 robot action sequences: ['place(blue
box, table)', 'pick(hook)', 'pull(red box,
hook)', 'place(hook, table)', 'pick(red
box)', 'place(red box, rack)', 'pick(cyan
box)', 'place(cyan box, table)']
```

# Appendix C Derivations

We provide two derivations to support our approximation of the skill score $S_{\text{skill}}$ (used to select skills while planning with **GREEDY-SEARCH** and **Text2Motion**) defined in Eq. 8. The skill score is expressed as a product of two terms:

$$S_{\text{skill}}(\psi_t) = p(\psi_t \mid i, s_1, \psi_{1:t-1}, r_{1:t-1}) \\ p(r_t \mid i, s_1, \psi_{1:t}, r_{1:t-1}). \tag{C1}$$

## C. 1 Skill usefulness derivation

Equation 9 defines the first term in the skill score product to be the skill *usefulness* score $S_{\text{llm}}$. We derive the approximation of $S_{\text{llm}}$ given in Eq. 10, which corresponds to quantity we use in our experiments.

$$S_{\text{llm}}(\psi_t) = p(\psi_t \mid i, s_1, \psi_{1:t-1}, r_{1:t-1})$$
$$= \int p(\psi_t \mid i, s_{1:t}, \psi_{1:t-1}, r_{1:t-1})$$
$$\quad p(s_{2:t} \mid i, s_1, \psi_{1:t-1}, r_{1:t-1}) \, ds_{2:t}$$
$$= \mathrm{E}_{s_{2:t}} \left[ p(\psi_t \mid i, s_{1:t}, \psi_{1:t-1}, r_{1:t-1}) \right] \tag{C2}$$
$$\approx \mathrm{E}_{s_{2:t}} \left[ p(\psi_t \mid i, s_{1:t}, \psi_{1:t-1}) \right] \tag{C3}$$
$$\approx p(\psi_t \mid i, s_{1:t}, \psi_{1:t-1}) \tag{C4}$$

The final expression is given in Eq. C4. Here, we compute a single sample Monte-Carlo estimate of Eq. C3 under the future state trajectory $s_2 \sim T^{\pi_1}(\cdot|s_1, a_1^*), \ldots, s_t \sim T^{\pi_{t-1}}(\cdot|s_{t-1}, a_{t-1}^*)$, where $a_{1:t-1}^*$ is computed by STAP (Agia et al., 2022). The key insight is that future state trajectories $s_{2:t}$ are only ever sampled after STAP has performed geometric feasibility planning to maximize the *success probability* (Eq. 3) of the running plan $\psi_{1:t-1}$. By doing so, we ensure that the future states $s_{2:t}$ correspond to a successful execution of the running plan $\psi_{1:t-1}$, i.e. achieving positive rewards $r_{1:t-1}$. This supports the independence assumption on rewards $r_{1:t-1}$ used to derive Eq. C3 from Eq. C2.

## C. 2 Skill feasibility derivation

Equation 11 defines the second term in the skill score product (Eq. C1) as the skill *feasibility* score $S_{\text{geo}}$. We derive the approximation provided in Eq. 12, which is the quantity we use in our experiments.

$$S_{\text{geo}}(\psi_t) = p(r_t \mid i, s_1, \psi_{1:t}, r_{1:t-1}) \tag{C5}$$
$$= p(r_t \mid s_1, \psi_{1:t}, r_{1:t-1}) \tag{C6}$$
$$= \int p(r_t \mid s_{1:t}, \psi_{1:t}, r_{1:t-1})$$
$$\quad p(s_{2:t} \mid s_1, \psi_{1:t}, r_{1:t-1}) \, ds_{2:t}$$
$$= \mathrm{E}_{s_{2:t}} \left[ p(r_t \mid s_{1:t}, \psi_{1:t}, r_{1:t-1}) \right] \tag{C7}$$

$$\approx \mathrm{E}_{s_{2:t}}\left[p(r_t \mid s_{1:t}, \psi_{1:t})\right] \tag{C8}$$

$$= \mathrm{E}_{s_{2:t}}\left[p(r_t \mid s_{1:t}, a_{1:t}^*)\right] \tag{C9}$$

$$= \mathrm{E}_{s_{2:t}}\left[p(r_t \mid s_t, a_t^*)\right] \tag{C10}$$

$$= \mathrm{E}_{s_{2:t}}\left[Q^{\pi_t}(s_t, a_t^*)\right] \tag{C11}$$

$$\approx Q^{\pi_t}(s_t, a_t^*) \tag{C12}$$

From Eqs. C5 to C6, the reward $r_t$ is conditionally independent of the instruction $i$ given the initial state $s_1$, running plan $\psi_{1:t}$, and previous rewards $r_{1:t-1}$. As described in "Appendix C.1", we can use STAP to make an independence assumption on the previous rewards $r_{1:t-1}$ between Eq. C7 and Eq. C8. The reward probability in Eq. C8 depends on the parameters $a_{1:t}^*$ computed by STAP and fed to the underlying primitive sequence $\phi_{1:t}$, which gives Eq. C9. Equation C10 comes from the Markov assumption, and can be reduced to Eq. C11 by observing that the reward probability $p(r_t \mid s_t, a_t^*)$ is equal to the Q-value $Q^{\pi_t}(s_t, a_t^*)$ in the contextual bandit setting we consider. The final expression given in Eq. C12, which represents a single sample Monte-Carlo estimate of Eq. C11 under a sampled future state trajectory $s_2 \sim T^{\pi_1}(\cdot|s_1, a_1^*), \ldots, s_t \sim T^{\pi_{t-1}}(\cdot|s_{t-1}, a_{t-1}^*)$.

## Appendix D Real world demonstration

### D. 1 Hardware setup

We use a Kinect V2 camera for RGB-D image capture and manually adjust the color thresholds to segment objects in the scene. Given the segmentation masks and the depth image, we can estimate object poses to construct the geometric state of the environment. For the skill library, we use the same set of policies, Q-functions, and dynamics models trained in simulation. We run robot experiments on a Franka Panda robot manipulator.

### D.2 Robot demonstration

Please see our project page for demonstrations of **Text2Motion** operating on a real robot.

## References

Aeronautiques, C., Howe, A., Knoblock, C., McDermott, I. D., Ram, A., Veloso, M., Weld, D., SRI, D. W., Barrett, A., Christianson, D., et al. (1998). PDDL| the planning domain definition language. Technical Report.

Agia, C., Migimatsu, T., Wu, J., & Bohg, J. (2022). STAP: Sequencing task-agnostic policies. arXiv preprint arXiv:2210.12250

Ahn, M., Brohan, A., Brown, N., Chebotar, Y., Cortes, O., David, B., Finn, C., Gopalakrishnan, K., Hausman, K., Herzog, A., et al. (2022). Do as i can, not as I say: Grounding language in robotic affordances. arXiv preprint arXiv:2204.01691

Ames, B., Thackston, A., & Konidaris, G. (2018). Learning symbolic representations for planning with parameterized skills. In *2018 IEEE/RSJ international conference on intelligent robots and systems (IROS)* (pp. 526–533). IEEE.

Bidot, J., Karlsson, L., Lagriffoul, F., & Saffiotti, A. (2017). Geometric backtracking for combined task and motion planning in robotic systems. *Artificial Intelligence, 247*, 229–265.

Bommasani, R., Hudson, D.A., Adeli, E., Altman, R., Arora, S., Arx, S., Bernstein, M.S., Bohg, J., Bosselut, A., Brunskill, E., et al. (2021). On the opportunities and risks of foundation models. arXiv preprint arXiv:2108.07258

Bonet, B., & Geffner, H. (2001). Planning as heuristic search. *Artificial Intelligence, 129*(1–2), 5–33.

Brohan, A., Brown, N., Carbajal, J., Chebotar, Y., Dabis, J., Finn, C., Gopalakrishnan, K., Hausman, K., Herzog, A., Hsu, J., Ibarz, J., Ichter, B., Irpan, A., Jackson, T., Jesmonth, S., Joshi, N., Julian, R., Kalashnikov, D., Kuang, Y., Leal, I., Lee, K.-H., Levine, S., Lu, Y., Malla, U., Manjunath, D., Mordatch, I., Nachum, O., Parada, C., Peralta, J., Perez, E., Pertsch, K., Quiambao, J., Rao, K., Ryoo, M., Salazar, G., Sanketi, P., Sayed, K., Singh, J., Sontakke, S., Stone, A., Tan, C., Tran, H., Vanhoucke, V., Vega, S., Vuong, Q., Xia, F., Xiao, T., Xu, P., Xu, S., Yu, T., & Zitkovich, B. (2022). Rt-1: Robotics transformer for real-world control at scale. In: arXiv Preprint arXiv:2212.06817

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. (2020). Language models are few-shot learners. *Advances in Neural Information Processing Systems, 33*, 1877–1901.

Chen, B., Xia, F., Ichter, B., Rao, K., Gopalakrishnan, K., Ryoo, M.S., Stone, A., & Kappler, D. (2022a). Open-vocabulary queryable scene representations for real world planning. arXiv preprint arXiv:2209.09874

Chen, M., Tworek, J., Jun, H., Yuan, Q., Pinto, H.P.d.O., Kaplan, J., Edwards, H., Burda, Y., Joseph, N., Brockman, G., et al. (2021). Evaluating large language models trained on code. arXiv preprint arXiv:2107.03374

Chen, Y., Yuan, L., Cui, G., Liu, Z., & Ji, H. (2022b). A close look into the calibration of pre-trained language models. arXiv preprint arXiv:2211.00151

Chitnis, R., Silver, T., Kim, B., Kaelbling, L., & Lozano-Perez, T. (2021). Camps: Learning context-specific abstractions for efficient planning in factored MDPS. In *Conference on robot learning* (pp. 64–79). PMLR.

Chitnis, R., Silver, T., Tenenbaum, J. B., Lozano-Perez, T., & Kaelbling, L. P. (2022). Learning neuro-symbolic relational transition models for bilevel planning. In *2022 IEEE/RSJ international conference on intelligent robots and systems (IROS)* (pp. 4166–4173). IEEE.

Curtis, A., Fang, X., Kaelbling, L. P., Lozano-Pérez, T., & Garrett, C. R. (2022). Long-horizon manipulation of unknown objects via task and motion planning with estimated affordances. In *2022 International Conference on Robotics and Automation (ICRA)* (pp. 1940–1946). IEEE.

Curtis, A., Silver, T., Tenenbaum, J. B., Lozano-Pérez, T., & Kaelbling, L. (2022). Discovering state and action abstractions for generalized task and motion planning. In *Proceedings of the AAAI conference on artificial intelligence* (Vol. 36, pp. 5377–5384).

Dalal, M., Mandlekar, A., Garrett, C., Handa, A., Salakhutdinov, R., & Fox, D. (2023). Imitating task and motion planning with visuomotor transformers. arXiv preprint arXiv:2305.16309

Dantam, N. T., Kingston, Z. K., Chaudhuri, S., & Kavraki, L. E. (2016). Incremental task and motion planning: A constraint-based approach. In *Robotics: Science and systems*, Ann Arbor, Michigan. https://doi.org/10.15607/RSS.2016.XII.002

Driess, D., Ha, J.-S., & Toussaint, M. (2020a). Deep visual reasoning: Learning to predict action sequences for task and motion planning from an initial scene image. arXiv preprint arXiv:2006.05398

Driess, D., Ha, J.-S., & Toussaint, M. (2021). Learning to solve sequential physical reasoning problems from a scene image. *The International Journal of Robotics Research, 40*(12–14), 1435–1466.

Driess, D., Ha, J.-S., Tedrake, R., & Toussaint, M. (2021b). Learning geometric reasoning and control for long-horizon tasks from visual input. In *2021 IEEE international conference on robotics and automation (ICRA)* (pp. 14298–14305). IEEE.

Driess, D., Huang, Z., Li, Y., Tedrake, R., & Toussaint, M. (2023). Learning multi-object dynamics with compositional neural radiance fields. In *Proceedings of the 6th conference on robot learning. Proceedings of machine learning research* (Vol. 205, pp. 1755–1768). PMLR.

Driess, D., Oguz, O., Ha, J.-S., Toussaint, M. (2020b). Deep visual heuristics: Learning feasibility of mixed-integer programs for manipulation planning. In *2020 IEEE international conference on robotics and automation (ICRA)* (pp. 9563–9569). IEEE.

Driess, D., Oguz, O., Toussaint, M. (2019). Hierarchical task and motion planning using logic-geometric programming (HLGP). In *RSS workshop on robust task and motion planning*.

Driess, D., Xia, F., Sajjadi, M.S., Lynch, C., Chowdhery, A., Ichter, B., Wahid, A., Tompson, J., Vuong, Q., Yu, T., et al. (2023). Palm-e: An embodied multimodal language model. arXiv preprint arXiv:2303.03378

Felip, J., Laaksonen, J., Morales, A., & Kyrki, V. (2013). Manipulation primitives: A paradigm for abstraction and execution of grasping and manipulation tasks. *Robotics and Autonomous Systems, 61*(3), 283–296.

Garrett, C. R., Chitnis, R., Holladay, R., Kim, B., Silver, T., Kaelbling, L. P., & Lozano-Pérez, T. (2021). Integrated task and motion planning. *Annual Review of Control, Robotics, and Autonomous Systems, 4*, 265–293.

Garrett, C. R., Lozano-Pérez, T., & Kaelbling, L. P. (2020). Pddl-stream: Integrating symbolic planners and blackbox samplers via optimistic adaptive planning. In *Proceedings of the international conference on automated planning and scheduling* (Vol. 30, pp. 440–448).

Haarnoja, T., Zhou, A., Abbeel, P., & Levine, S. (2018). Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In J. Dy, & A. Krause (Eds.), *Proceedings of the 35th international conference on machine learning. Proceedings of machine learning research* (Vol. 80, pp. 1861–1870). PMLR. https://proceedings.mlr.press/v80/haarnoja18b.html

Helmert, M. (2006). The fast downward planning system. *Journal of Artificial Intelligence Research, 26*, 191–246.

Huang, W., Abbeel, P., Pathak, D., & Mordatch, I. (2022). Language models as zero-shot planners: Extracting actionable knowledge for embodied agents. arXiv preprint arXiv:2201.07207

Huang, W., Xia, F., Xiao, T., Chan, H., Liang, J., Florence, P., Zeng, A., Tompson, J., Mordatch, I., Chebotar, Y., et al. (2022). Inner monologue: Embodied reasoning through planning with language models. arXiv preprint arXiv:2207.05608

Jang, E., Irpan, A., Khansari, M., Kappler, D., Ebert, F., Lynch, C., Levine, S., & Finn, C. (2021). BC-z: Zero-shot task generalization with robotic imitation learning. In *5th annual conference on robot learning*. https://openreview.net/forum?id=8kbp23tSGYv

Jiang, Y., Gupta, A., Zhang, Z., Wang, G., Dou, Y., Chen, Y., Fei-Fei, L., Anandkumar, A., Zhu, Y., & Fan, L. (2022). Vima: General robot manipulation with multimodal prompts. arXiv preprint arXiv:2210.03094

Kaelbling, L. P., & Lozano-Pérez, T. (2011). Hierarchical task and motion planning in the now. In *2011 IEEE international conference on robotics and automation* (pp. 1470–1477). https://doi.org/10.1109/ICRA.2011.5980391

Kaelbling, L. P., & Lozano-Pérez, T. (2012). *Integrated robot task and motion planning in the now*. Technical report: Massachusetts Inst of Tech Cambridge Computer Science and Artificial.

Kalashnkov, D., Varley, J., Chebotar, Y., Swanson, B., Jonschkowski, R., Finn, C., Levine, S., & Hausman, K. (2021). Mt-opt: Continuous multi-task robotic reinforcement learning at scale. arXiv

Khatib, O. (1987). A unified approach for motion and force control of robot manipulators: The operational space formulation. *IEEE Journal on Robotics and Automation, 3*(1), 43–53.

Kim, B., Kaelbling, L. P., & Lozano-Pérez, T. (2019). Adversarial actor-critic method for task and motion planning problems using planning experience. In *Proceedings of the AAAI conference on artificial intelligence* (Vol. 33, pp. 8017–8024).

Kim, B., Shimanuki, L., Kaelbling, L. P., & Lozano-Pérez, T. (2022). Representation, learning, and planning algorithms for geometric task and motion planning. *The International Journal of Robotics Research, 41*(2), 210–231.

Konidaris, G., Kaelbling, L. P., & Lozano-Perez, T. (2018). From skills to symbols: Learning symbolic representations for abstract high-level planning. *Journal of Artificial Intelligence Research, 61*, 215–289.

Kroemer, O., & Sukhatme, G. S. (2016). Learning spatial preconditions of manipulation skills using random forests. In *2016 IEEE-RAS 16th international conference on humanoid robots (Humanoids)* (pp. 676–683). IEEE.

Lagriffoul, F., Dimitrov, D., Bidot, J., Saffiotti, A., & Karlsson, L. (2014). Efficiently combining task and motion planning using geometric constraints. *The International Journal of Robotics Research, 33*(14), 1726–1747.

Lakshminarayanan, B., Pritzel, A., & Blundell, C. (2017). Simple and scalable predictive uncertainty estimation using deep ensembles. In *Advances in neural information processing systems* (Vol. 30).

Li, X.L., Holtzman, A., Fried, D., Liang, P., Eisner, J., Hashimoto, T., Zettlemoyer, L., & Lewis, M. (2022). Contrastive decoding: Open-ended text generation as optimization. arXiv preprint arXiv:2210.15097

Liang, J., Huang, W., Xia, F., Xu, P., Hausman, K., Ichter, B., Florence, P., & Zeng, A. (2022). Code as policies: Language model programs for embodied control. arXiv preprint arXiv:2209.07753

Liu, B., Jiang, Y., Zhang, X., Liu, Q., Zhang, S., Biswas, J., & Stone, P. (2023). Llm+ p: Empowering large language models with optimal planning proficiency. arXiv preprint arXiv:2304.11477

Loshchilov, I., & Hutter, F. (2017). SGDR: Stochastic gradient descent with warm restarts. In *International conference on learning representations*. https://openreview.net/forum?id=Skq89Scxx

Mees, O., Hermann, L., Rosete-Beas, E., & Burgard, W. (2022). Calvin: A benchmark for language-conditioned policy learning for long-horizon robot manipulation tasks. *IEEE Robotics and Automation Letters (RA-L), 7*(3), 7327–7334.

OpenAI: GPT-4 Technical Report (2023)

Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C. L., Mishkin, P., Zhang, C., Agarwal, S., Slama, K., Ray, A., et al. (2022). Training language models to follow instructions with human feedback. arXiv preprint arXiv:2203.02155

Rubinstein, R. (1999). The cross-entropy method for combinatorial and continuous optimization. *Methodology and Computing in Applied Probability, 1*(2), 127–190.

Shao, L., Migimatsu, T., Zhang, Q., Yang, K., & Bohg, J. (2021). Concept2robot: Learning manipulation concepts from instructions and human demonstrations. *The International Journal of Robotics Research, 40*(12–14), 1419–1434.

Shridhar, M., Manuelli, L., & Fox, D. (2022a). Cliport: What and where pathways for robotic manipulation. In: Conference on robot learning (pp. 894–906). PMLR.

Shridhar, M., Manuelli, L., & Fox, D. (2022b). Perceiver-actor: A multi-task transformer for robotic manipulation. arXiv preprint arXiv:2209.05451

Silver, T., Athalye, A., Tenenbaum, J. B., Lozano-Pérez, T., & Kaelbling, L. P. (2022). Learning neuro-symbolic skills for bilevel planning. In *6th annual conference on robot learning*. https://openreview.net/forum?id=OIaJRUo5UXy.

Silver, T., Chitnis, R., Curtis, A., Tenenbaum, J. B., Lozano-Perez, T., & Kaelbling, L. P. (2021). Planning with learned object importance in large problem instances using graph neural networks. In *Proceedings of the AAAI conference on artificial intelligence* (Vol. 35, pp. 11962–11971).

Silver, T., Chitnis, R., Tenenbaum, J., Kaelbling, L. P., & Lozano-Pérez, T. (2021). Learning symbolic operators for task and motion planning. In *2021 IEEE/RSJ international conference on intelligent robots and systems (IROS)* (pp. 3182–3189). IEEE.

Silver, T., Hariprasad, V., Shuttleworth, R. S., Kumar, N., Lozano-Pérez, T., & Kaelbling, L. P. (2022). PDDL planning with pretrained large language models. In *NeurIPS 2022 foundation models for decision making workshop*.

Singh, I., Blukis, V., Mousavian, A., Goyal, A., Xu, D., Tremblay, J., Fox, D., Thomason, J., & Garg, A. (2022). Progprompt: Generating situated robot task plans using large language models. arXiv preprint arXiv:2209.11302

Skreta, M., Yoshikawa, N., Arellano-Rubach, S., Ji, Z., Kristensen, L.B., Darvish, K., Aspuru-Guzik, A., Shkurti, F., & Garg, A. (2023). Errors are useful prompts: Instruction guided task programming with verifier-assisted iterative prompting. arXiv preprint arXiv:2303.14100

Stepputtis, S., Campbell, J., Phielipp, M., Lee, S., Baral, C., & Ben Amor, H. (2020). Language-conditioned imitation learning for robot manipulation tasks. *Advances in Neural Information Processing Systems, 33*, 13139–13150.

Toussaint, M. (2015). Logic-geometric programming: An optimization-based approach to combined task and motion planning. In *Twenty-fourth international joint conference on artificial intelligence*.

Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.-A., Lacroix, T., Rozière, B., Goyal, N., Hambro, E., Azhar, F., et al. (2023). Llama: Open and efficient foundation language models. arXiv preprint arXiv:2302.13971

Valmeekam, K., Olmo, A., Sreedharan, S., & Kambhampati, S. (2022). Large language models still can't plan (a benchmark for LLMS on planning and reasoning about change). arXiv preprint arXiv:2206.10498

Vemprala, S., Bonatti, R., Bucker, A., & Kapoor, A. (2023). Chatgpt for robotics: Design principles and model abilities. Technical Report MSR-TR-2023-8, Microsoft

Wang, Z., Cai, S., Liu, A., Ma, X., & Liang, Y. (2023). Describe, explain, plan and select: Interactive planning with large language models enables open-world multi-task agents. arXiv preprint arXiv:2302.01560

Wang, Z., Garrett, C. R., Kaelbling, L. P., & Lozano-Pérez, T. (2018). Active model learning and diverse action sampling for task and motion planning. In *2018 IEEE/RSJ international conference on intelligent robots and systems (IROS)* (pp. 4107–4114). IEEE.

Wang, Z., Garrett, C. R., Kaelbling, L. P., & Lozano-Pérez, T. (2021). Learning compositional models of robot skills for task and motion planning. *The International Journal of Robotics Research, 40*(6–7), 866–894.

Williams, G., Aldrich, A., & Theodorou, E. (2015). Model predictive path integral control using covariance variable importance sampling. arXiv preprint arXiv:1509.01149

Wu, J., Antonova, R., Kan, A., Lepert, M., Zeng, A., Song, S., Bohg, J., Rusinkiewicz, S., & Funkhouser, T. (2023). Tidybot: Personalized robot assistance with large language models. arXiv preprint arXiv:2305.05658

Xu, D., Mandlekar, A., Martín-Martín, R., Zhu, Y., Savarese, S., & Fei-Fei, L. (2021). Deep affordance foresight: Planning through what can be done in the future. In *2021 IEEE international conference on robotics and automation (ICRA)* (pp. 6206–6213). IEEE.

Zelikman, E., Huang, Q., Poesia, G., Goodman, N. D., & Haber, N. (2022). Parsel: A unified natural language framework for algorithmic reasoning. arXiv preprint arXiv:2212.10561

Zeng, A., Wong, A., Welker, S., Choromanski, K., Tombari, F., Purohit, A., Ryoo, M., Sindhwani, V., Lee, J., Vanhoucke, V., et al. (2022). Socratic models: Composing zero-shot multimodal reasoning with language. arXiv preprint arXiv:2204.00598

Zhao, Z., Wallace, E., Feng, S., Klein, D., & Singh, S. (2021). Calibrate before use: Improving few-shot performance of language models. In *International conference on machine learning* (pp. 12697–12706). PMLR.

Zhou, L., Palangi, H., Zhang, L., Hu, H., Corso, J., & Gao, J. (2020). Unified vision-language pre-training for image captioning and VQA. In *Proceedings of the AAAI conference on artificial intelligence* (Vol. 34, pp. 13041–13049).

**Kevin Lin** is currently pursuing a Master of Science degree in Computer Science at Stanford University. Prior to this, he obtained his Bachelor of Science degree in Electrical Engineering and Computer Science from UC Berkeley. His research focuses on developing robust and intelligent robotic systems that can meaningfully manipulate the world around them.

**Christopher Agia** is a Ph.D. candidate in the Department of Computer Science at Stanford University, where he works jointly with the Interactive Perception and Robot Learning Laboratory (IPRL) and the Autonomous Systems Laboratory (ASL). He received his B.A.Sc. degree in Engineering Science, Robotics from the University of Toronto in 2021. His research focuses on the development of learning-based methodologies for planning, decision-making, and control of robotic manipulation and aerospace systems. He is a member of the Stanford Artificial Intelligence Laboratory and the Center for Research on Foundation Models, and has held internships at the NASA Jet Propulsion Laboratory, Microsoft Mixed Reality, Google Cloud, and the Noah's Ark Research Laboratory.

**Toki Migimatsu** received his Ph.D. degree in Computer Science from Stanford University in 2023, where he conducted research on learning-based methodologies for scaling robotic task and motion planning to real world environments. Prior to this, he completed both his Master of Science and Bachelor of Science in Computer Science at Stanford University. Toki Migimatsu is currently a Robotics Engineer at Figure AI.



**Marco Pavone** is an Associate Professor of Aeronautics and Astronautics at Stanford University, where he directs the Autonomous Systems Laboratory and the Center for Automotive Research at Stanford. He is also a Distinguished Research Scientist at NVIDIA where he leads autonomous vehicle research. Before joining Stanford, he was a Research Technologist within the Robotics Section at the NASA Jet Propulsion Laboratory. He received a Ph.D. degree in Aeronautics and Astronautics from the Massachusetts Institute of Technology in 2010. His main research interests are in the development of methodologies for the analysis, design, and control of autonomous systems, with an emphasis on self-driving cars, autonomous aerospace vehicles, and future mobility systems. He is a recipient of a number of awards, including a Presidential Early Career Award for Scientists and Engineers from President Barack Obama.



**Jeannette Bohg** is an Assistant Professor of Computer Science at Stanford University. She was a group leader at the Autonomous Motion Department (AMD) of the MPI for Intelligent Systems until September 2017. Before joining AMD in January 2012, Jeannette Bohg was a Ph.D. student at the Division of Robotics, Perception and Learning (RPL) at KTH in Stockholm. In her thesis, she proposed novel methods towards multi-modal scene understanding for robotic grasping. She also studied at Chalmers in Gothenburg and at the Technical University in Dresden where she received her Master in Art and Technology and her Diploma in Computer Science, respectively. Her research focuses on perception and learning for autonomous robotic manipulation and grasping. She is specifically interested in developing methods that are goal-directed, real-time and multi-modal such that they can provide meaningful feedback for execution and learning. Jeannette Bohg has received several Early Career and Best Paper awards, most notably the 2019 IEEE Robotics and Automation Society Early Career Award and the 2020 Robotics: Science and Systems Early Career Award.