Text2Motion: From Natural Language Instructions to Feasible Plans

Project page: sites.google.com/stanford.edu/text2motion

Kevin Lin*†, Christopher Agia*†‡, Toki Migimatsu†, Marco Pavone‡ and Jeannette Bohg†

† Department of Computer Science, Stanford University, California, U.S.A.

‡ Department of Aeronautics & Astronautics, Stanford University, California, U.S.A.

Email: {kevinlin0,cagia,takatoki,pavone,bohg}@stanford.edu

Shao ZHANG 2023.04.21

Motivation

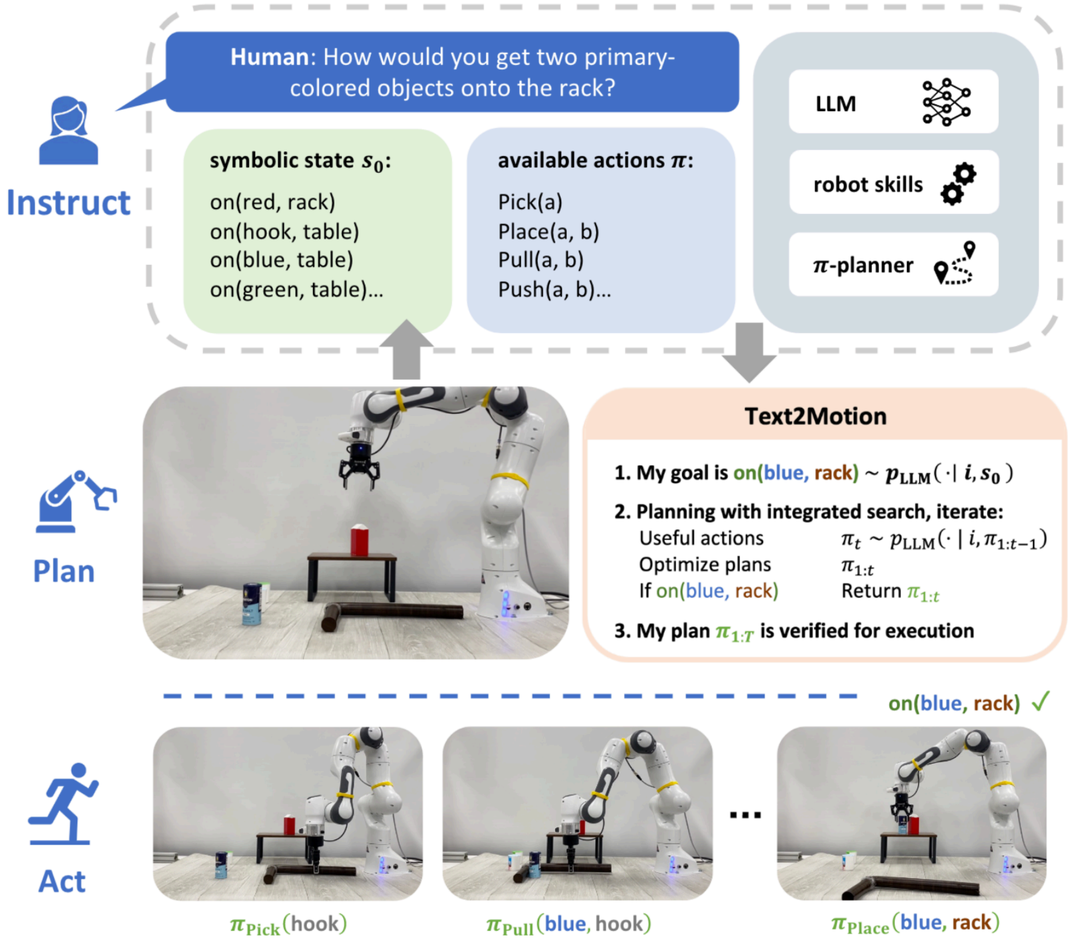
- **Task and Motion Planning (TAMP)** refers to a problem setting in which a robot has to solve long-horizon tasks that require both symbolic and geometric reasoning

- The emergence of Large Language Models (LLMs) as a task-agnostic reasoning module presents a promising pathway to general robot planning capabilities.

- How can we verify the correctness and feasibility of long-horizon LLM-generated plans on the symbolic and geometric level?

Solution

- Text2Motion
  - a language-based planning framework
  - interfaces an LLM with a library of learned skill policies and a policy sequence optimizer to solve geometrically complex sequential manipulation tasks.



Contribution

- (i) an integrated search algorithm which interleaves LLM task planning with policy sequence optimization to construct geometrically feasible plans for tasks not seen by the skill policies during training;

- (ii) a plan termination method that infers goal states from a natural language instruction to verify the completion of plans. We find that our integrated method achieves a success rate of 64% on a suite of challenging table top manipulation tasks, while prior language-based planning methods achieve a 13% success rate.

Settings

- A library of skill $\mathscr{L} = \{\pi^1, \ldots, \pi^N\}$
- Each skill has a natural language description and comes with a policy $\pi(a \mid s)$,a Q-function $\mathcal{Q}^\pi(s, a)$, a dynamics model$T^\pi(s' \mid s, a)$
- Actions output by the policy $a \sim \pi(\cdot \mid s)$ are the parameters of a corresponding manipulation primitive $\rho(a)$ which consumes the action and executes a series of motor commands on the robot
- We also assume that a method exists for conveying the environment state s to the LLM **as natural language**.

Settings

- The task planning problem is to find a sequence of skills $[\pi_1, \ldots, \pi_H]$ that is likely to satisfy the instruction i (for notational convenience, we will hereafter represent sequences with range subscripts, e.g. $\pi_{1:H}$).
- The task planning objective is to maximize the language model likelihood of the skill sequence $\pi_{1:H}$ given instruction i and initial state s_1:

$$p(\pi_{1:H} \mid i, s_1) \tag{1}$$

Settings

- For example, if the goal is to move a box from the table to the rack, a symbolically correct sequence of actions might be Pick(box), Place(box, rack). However, we must also consider whether the **skill sequence can succeed from a geometric perspective**.
- Specifically, for each skill π_h, we need to consider the geometric feasibility of the underlying continuous parameters a_h.
- A geometrically feasible plan is one where each skill π_h and its continuous action parameter ah receives a binary reward r_h ; if just one action fails, then the entire plan fails.
- The **geometric feasibility** is defined to be the probability that all skills $\pi_{1:H}$ achieve rewards

$$p(r_{1:H} \mid i, s_1, \pi_{1:H}) \tag{2}$$

Objective

$$p(\pi_{1:H} \mid i, s_1) \tag{1}$$

$$p(r_{1:H} \mid i, s_1, \pi_{1:H}) \tag{2}$$

$$p(\pi_{1:H}, r_{1:H} \mid i, s_1) \tag{3}$$

$$= p(\pi_{1:H} \mid i, s_1) p(r_{1:H} \mid i, s_1, \pi_{1:H})$$

the probability that a skill sequence $\pi_{1:H}$ is both likely to satisfy instruction i and is geometrically feasible:

TEXT2MOTION

- We follow a modular approach similar to traditional TAMP methods but replace the commonly used symbolic task planner with an LLM. The core idea of this paper is ensure **the geometric feasibility of an LLM task plan**—and thereby its correctness—by predicting the success probability of learned skills that are sequenced according to the task plan.

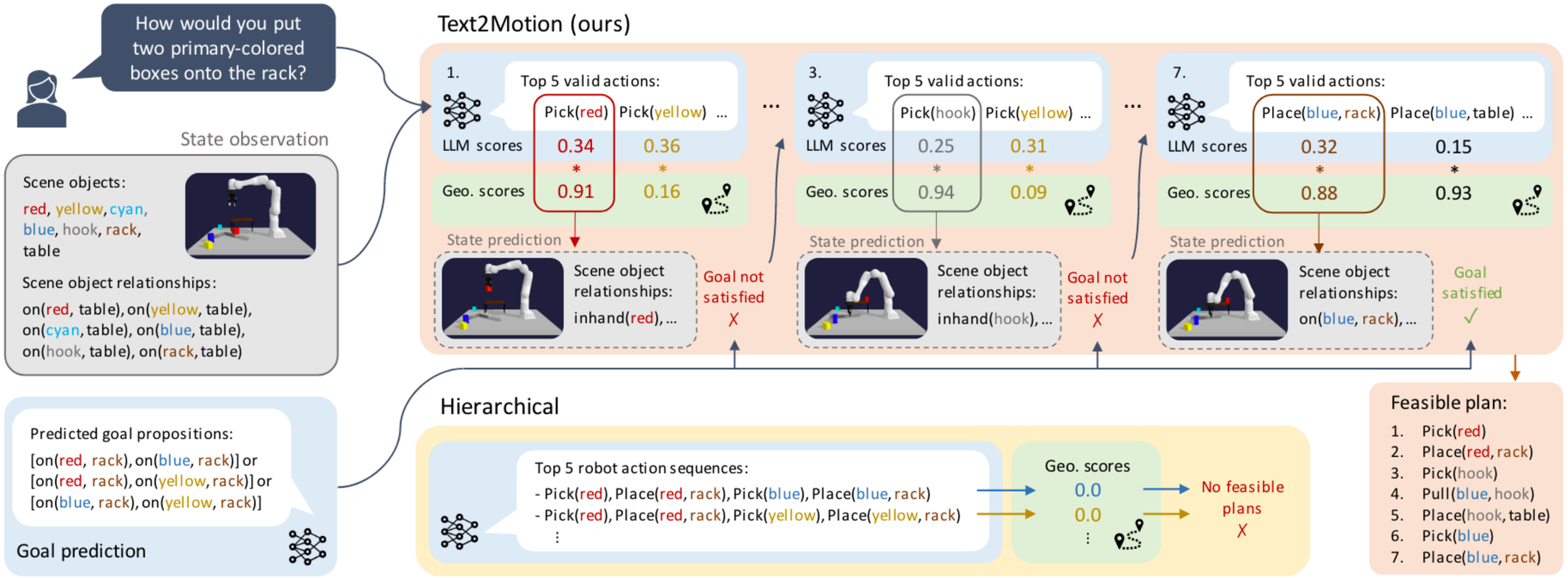


Fig. 2. Text2Motion planning overview. The user provides a natural language instruction for the robot, and then Text2Motion outputs a feasible sequence of skills to solve the given task. First, we use the LLM to predict the set of valid goal state propositions given the instruction and current state (left). This goal prediction will be used during planning to decide when the goal is satisfied and planning can be terminated. In the first planning iteration, Text2Motion uses the LLM 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 predicted state does not satisfy the goal propositions, then it is given to the LLM to plan the next skill. If the goal propositions are satisfied, then the planner returns. Text2Motion interleaves LLM planning with policy sequence optimization at each iteration. The HIERARCHICAL method, which is used as an experiment baseline, uses the LLM to propose entire plans first and then runs policy sequence optimization afterwards. As shown in the experiments, this approach fails when the space of candidate task plans is large but few skills are geometrically feasible.

TEXT2MOTION

- This iterative approach can be described as a decomposition of the joint probability in Eq. 3 by timestep h

$$p(\pi_{1:H}, r_{1:H} \mid i, s_1) = \prod_{h=1}^{H} p(\pi_h, r_h \mid i, s_1, \pi_{1:h-1}, r_{1:h-1}) \tag{4}$$

$$\approx \prod_{h=1}^{H} p(\pi_h, r_h \mid i, s_1, \pi_{1:h-1}) \tag{5}$$

- This allows us to further decompose Eq. 5 into the joint probability of π_h and r_h, which we define as the skill score S_{skill} :

$$S_{\text{skill}}(\pi_h) = p(\pi_h, r_h \mid i, s_1, \pi_{1:h-1}). \tag{6}$$

TEXT2MOTION

- Each planning iteration is responsible for finding the skill π_h that maximizes the skill score at timestep h. We decompose this score into the conditional probabilities of π_h and r_h:

$$S_{\text{skill}}(\pi_h) = p(\pi_h \mid i, s_1, \pi_{1:h-1})\, p(r_h \mid i, s_1, \pi_{1:h})$$

- We define the first factor in this product to be the language model likelihood score:

$$S_{\text{llm}}(\pi_h) = p(\pi_h \mid i, s_1, \pi_{1:h-1}). \tag{7}$$

- We thus define the geometric feasibility score:

$$S_{\text{geo}}(\pi_h) = p(r_h \mid s_1, \pi_{1:h}). \tag{8}$$

- The skill score to be optimized at each iteration of planning is therefore the product of the LLM likelihood and the geometric feasibility of the planned skill sequence:

$$S_{\text{skill}}(\pi_h) = S_{\text{llm}}(\pi_h) \cdot S_{\text{geo}}(\pi_h). \tag{9}$$

Geometric feasibility planning

- we first resolve geometric dependencies across the full skill sequence $\pi_{1:h}$ by maximizing the product of step reward probabilities of the skills' individual actions $a_{1:h}$:

$$a^*_{1:h} = \arg\max_{a_{1:h}} \prod_{t=1}^{h} p(r_t \mid s_t, a_t), \qquad (10)$$

- where future states $s_{2:h}$ are predicted by dynamics models $s_{t+1} = T^{\pi}(s_t, a_t)$. Note that the reward probability $p(r_t = 1 \mid s_t, a_t)$ is equivalent to a Q-function $Q^{\pi_t}(s_t, a_t)$ for skill π_t in a contextual bandit setting with binary rewards.

- We can then estimate the geometric feasibility $S_{geo}(\pi_h)$ (Eq. 8) in the context of the full skill sequence $\pi_{1:h}$ by the Q-value of the last action,

$$p(r_h = 1 \mid s_1, \pi_{1:h}) \approx Q^{\pi_h}(s_h, a^*_h), \qquad (11)$$

$$S_{geo}(\pi_h)$$

- where a$*$ h is determined by STAP and s_h is predicted by the dynamics model. The Q-value is multiplied by the language model likelihood (Eq. 7) to produce the combined overall score (Eq. 9) for this skill.

LLM goal prediction

- Given a library of predicate classifiers $\mathscr{L}^P$ describing simple geometric relationships of objects in the scene (e.g. {on(a, b), inhand(a), under(a, b)}), a list of objects O in the scene, and an instruction i, we use the LLM to predict j goal proposition sets $g_{1:j}$ that would satisfy the instruction.

- We define a satisfaction function $f_{sat}^{g_{1:j}}(s) \in \{0,1\}$ that checks whether the current symbolic state s—obtained from the predicate classifiers $\mathscr{L}^P$ — satisfies any of the goal proposition sets $g_{1:j}$.

- We use f_{sat} in two ways: i) as a success classifier to check whether a given state satisfies the natural language instruction and ii) as a chain-of-thought prompt when prompting the LLM for action sequences.

Experiments

Long-horizon (LH): task requires skill sequences π1:H of length six or greater to solve. (Task 12356)

Lifted goals (LG): Goals are expressed over object classes rather than object instances. (Task 456)

Partial affordance perception (PAP): Skill affordances cannot be perceived solely from the spatial relationships described in the initial state s1.(Task 456)

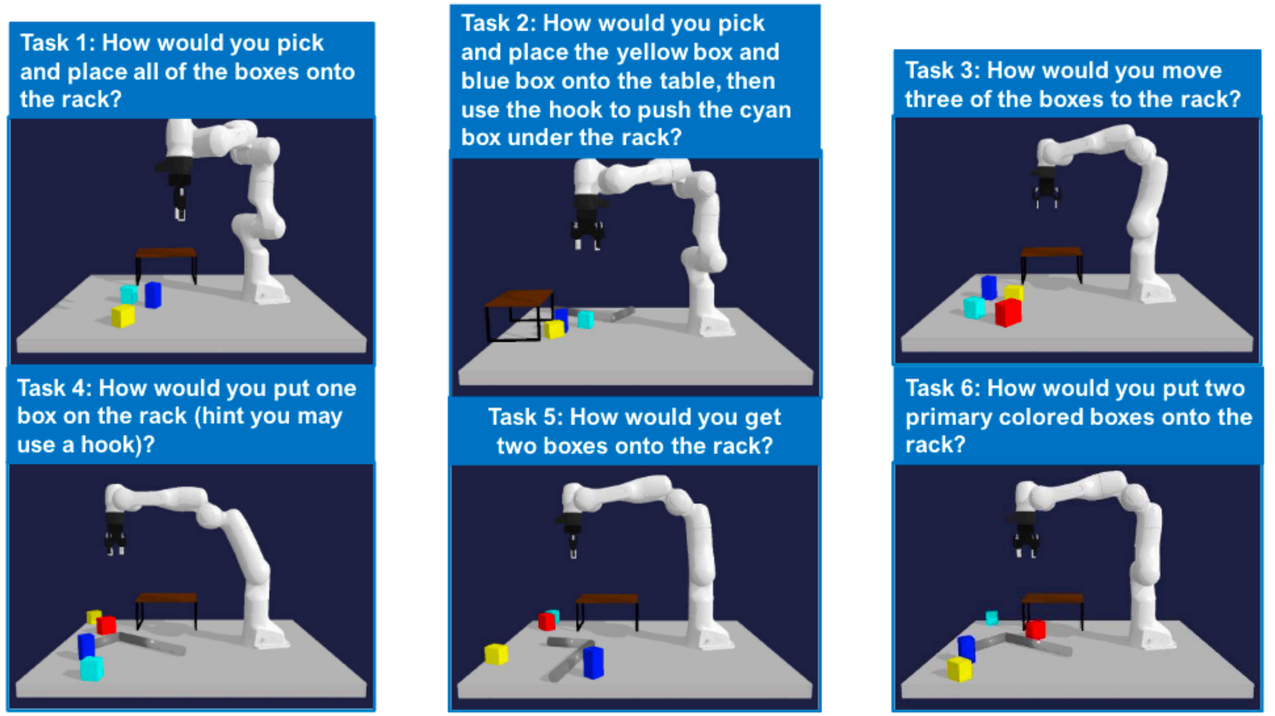


Fig. 3. **TableEnv Manipulation evaluation task suite**. We evaluate the performance of all methods on a task suite based on the above manipulation domain. The tasks considered vary in terms of difficult and each contains a subset of three properties: being long-horizon (Tasks 1, 2, 3, 5, 6), containing 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 for each random seed.

Experiments

generate task plans:
text-davinci-003

Other:
code-davinci-002

```
 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)’]
```

Experiments

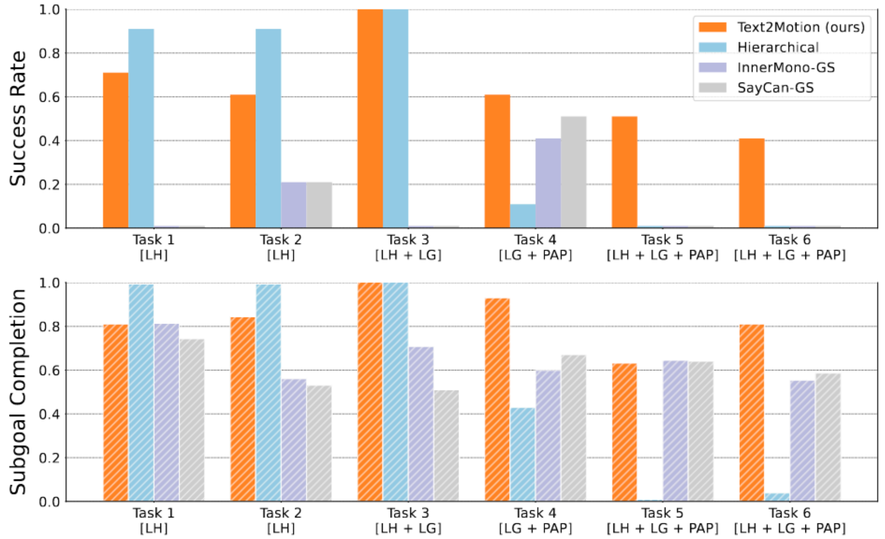


Fig. 4. **Results on the TableEnv manipulation domain** with 10 seeds for each task. **Top:** Our method (Text2Motion) significantly outperforms all baselines on tasks involving partial affordance perception (Task 4, 5, 6). For the tasks without partial affordance perception, the methods that use policy sequence optimization (ours and HIERARCHICAL) both convincingly outperform the methods (SAYCAN-GS and INNERMONO-GS) that do not use policy sequence optimization. We note that HIERARCHICAL performs well on the tasks without partial affordance perception as it has the advantage of outputting *multiple* goal-reaching candidate task plans and selecting the most geometrically feasible. **Bottom:** Methods without policy sequence optimization 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.

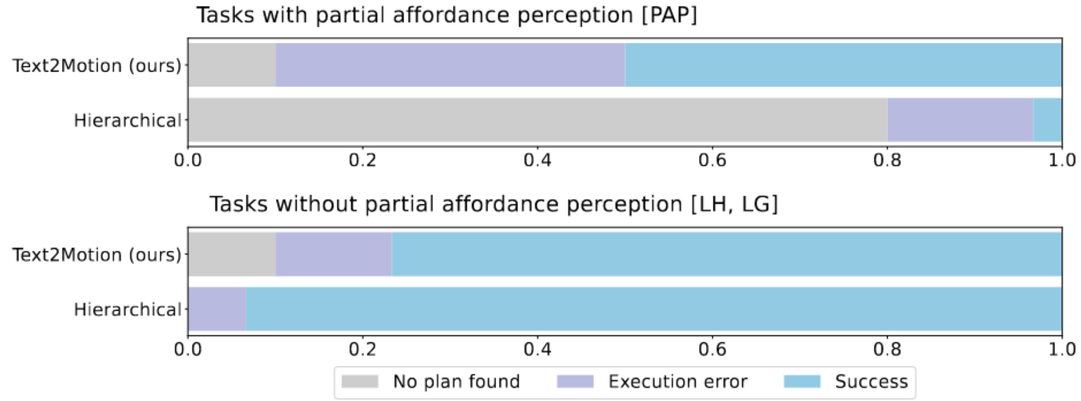


Fig. 5. **Failure modes of planning based methods on two types of tasks** In this plot, we analyse the various types of failure modes that occur with the integrated planner and the hierarchical planner when evaluated on tasks with partial affordance perception (PAP; see Sec. V-D for an explanation) and tasks without partial affordance perception (non-PAP). **Top**: For the PAP tasks, the hierarchical planner outperforms the integrated planner. We attribute this difference to the hierarchical planner's ability to output multiple task plans while the integrated planner can only output a single plan. **Bottom**: For the non-PAP tasks, the hierarchical planner is much less likely to output a plan than the integrated approach as the integrated approach is able to use its value functions to prune away geometrically infeasible action sequences.