STAP: Sequencing Task-Agnostic Policies

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

Christopher Agia*, Toki Migimatsu*, Jiajun Wu, Jeannette Bohg
Department of Computer Science, Stanford University, California, U.S.A.
Email: {cagia,takatoki,jiajunw,bohg}@stanford.edu

***Abstract*— Advances in robotic skill acquisition have made it possible to build general-purpose libraries of learned skills for downstream manipulation tasks. However, naively executing these skills one after the other is unlikely to succeed without accounting for dependencies between actions prevalent in long-horizon plans. We present Sequencing Task-Agnostic Policies (STAP), a scalable framework for training manipulation skills and coordinating their geometric dependencies at planning time to solve long-horizon tasks never seen by any skill during training. Given that Q-functions encode a measure of skill feasibility, we formulate an optimization problem to maximize the joint success of all skills sequenced in a plan, which we estimate by the product of their Q-values. Our experiments indicate that this objective function approximates ground truth plan feasibility and, when used as a planning objective, reduces myopic behavior and thereby promotes long-horizon task success. We further demonstrate how STAP can be used for task and motion planning by estimating the geometric feasibility of skill sequences provided by a task planner. We evaluate our approach in simulation and on a real robot.**

I. Introduction

Performing sequential manipulation tasks requires a robot to reason about dependencies between actions. Consider the example in Fig. 1, where the robot needs to retrieve an object outside of its workspace by first using an L-shaped hook to pull the target object closer. How the robot picks up the hook affects whether the target object will be reachable.

Traditionally, planning actions to ensure the geometric feasibility of a sequential manipulation task is handled by motion planning [1–3], which typically requires full observability of the environment state and knowledge of its dynamics. Learning-based approaches [4–6] can acquire skills without this privileged information. However, using independently learned skills to perform unseen long-horizon manipulation tasks is an unsolved problem. The skills could be myopically executed one after another to solve a simpler subset of tasks, but solving more complex tasks requires planning with these skills to ensure the feasibility of the entire skill sequence.

Prior work focuses on sequencing skills at *train time* to solve a small set of sequential manipulation tasks [7, 8]. To contend with long-horizons, these methods often learn skills [9] that consist of a policy and parameterized manipulation primitive [10]. The policy predicts the parameters of the primitive, thereby governing its motion. Such methods are *task-specific* in that they need to be trained on skill sequences that reflect the tasks they might encounter at test time. In our framework, we assume that a task planner provides a novel sequence of skills at *test time* that will then be grounded with parameters for manipulation primitives through optimization. This makes our method *task-agnostic*, as skills can be sequenced to solve long-horizon tasks not seen during training.

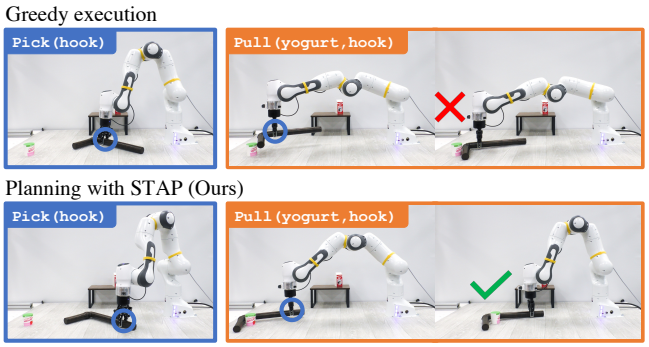


Fig. 1: Sequential manipulation tasks often contain geometric dependencies between actions. In this example, the robot needs to use the hook to pull the block into its kinematic workspace so it is close enough to pick up. The top row shows how greedy execution of skills results in the robot picking up the hook in a way that prevents it from reaching the block. We present a method for planning with skills to maximize long-horizon success without the need to train the skills on long-horizon tasks.

At the core of our method, *Sequencing Task-Agnostic Policies* (STAP), we use Q-functions to optimize the parameters of manipulation primitives in a given sequence. Policies and Q-functions for each skill are acquired through off-the-shelf Reinforcement Learning. We then define a planning objective to maximize all Q-functions in a skill sequence, ensuring its geometric feasibility. To evaluate downstream Q-functions of future skills, we learn a dynamics model that can predict future states. We also use *Uncertainty Quantification* (UQ) to avoid visiting states that are *Out-Of-Distribution* (OOD) for the learned skills. We train all of these components independently per skill, making it easy to gradually expand a library of skills without the need to retrain existing ones.

Our contributions are three-fold: we propose 1) a framework to train an extensible library of task-agnostic skills, 2) a planning method that optimizes arbitrary sequences of skills to solve long-horizon tasks, and 3) a method to solve Task and Motion Planning (TAMP) problems with learned skills. In extensive experiments, we demonstrate that planning with STAP promotes long-horizon success on tasks with complex geometric dependencies between actions. We also demonstrate that our framework works on a real robot.

II. Related Work

A. Robot skill learning

How to represent and acquire composable manipulation skills is a widely studied problem in robotics. A broad class of methods uses Learning from Demonstration (LfD) [5].

*Authors contributed equally to this work.
Toyota Research Institute provided funds to support this work.

Dynamic Movement Primitives (DMPs) [11–13] are a form of LfD that learns the parameters of dynamical systems encoding movements [14–17]. More recent extensions integrate DMPs with deep neural networks to learn more flexible policies [18, 19]—for instance, to build a large library of skills from human video demonstrations [20]. Skill discovery methods instead identify action patterns in offline datasets [21] and either distill them into policies [22, 23] or extract skill priors for use in downstream tasks [24, 25]. Robot skills can also be acquired via active learning [26], Reinforcement Learning (RL) [27–31], and offline RL [32].

An advantage of our planning framework is that it is agnostic to the types of skills employed, requiring only that it is possible to predict the probability of the skill's success given the current state and action. Here, we learn skills [9] that consist of a policy and a parameterized manipulation primitive [10]. The actions output by the policy are the parameters of the primitive determining its motion. In STAP, we will use the Q-functions of the policy to optimize suitable parameters [20, 28] for a sequence of manipulation primitives.

B. Long-horizon robot planning

Once manipulation skills have been acquired, using them to perform sequential manipulation tasks remains an open challenge. [33–36] propose data-driven methods to determine the *symbolic* feasibility of skills and only control their timing, while we seek to ensure the *geometric* feasibility of skills by controlling their trajectories. Other techniques rely on task planning [37, 38], subgoal planning [39], or meta-adaptation [40, 41] to sequence learned skills to novel long-horizon goals. However, the tasks considered in these works do not feature rich geometric dependencies between actions that necessitate motion planning or skill coordination.

The options framework [42] and the parameterized action Markov Decision Process (MDP) [43] train a high-level policy to engage low-level policies [44, 45] or primitives [8, 46–48] towards long-horizon goals. [49] proposes a hierarchical RL method that uses the value functions of lower-level policies as the state space for a higher-level RL policy. Our work is also related to model-based RL methods which jointly learn dynamics and reward models to guide planning [50–52], policy search [53, 54], or combine both [55, 56]. While these methods demonstrate that policy hierarchies and model-based planning can enable RL to solve long-horizon problems, they are typically trained in the context of a single task. In contrast, we seek to *plan* with lower-level skills to solve tasks never seen before.

Closest in spirit to our work is that of Xu et al. [7], Deep Affordance Foresight (DAF), which proposes to learn a dynamics model, skill-centric affordances (value functions), and a skill proposal network that serves as a higher-level RL policy. We identify several drawbacks with DAF: first, because DAF relies on multi-task experience for training, generalizing beyond the distribution of training tasks may be difficult; second, the dynamics, affordance models, and skill proposal network need to be trained synchronously, which complicates expanding the current library of trained skills; third, their planner samples actions from uniform random distributions, which prevents DAF from scaling to high-dimensional action spaces and long horizons. STAP differs in that our dynamics, policies, and affordances (Q-functions) are learned independently per skill. Without any additional training, we combine the skills at planning time to solve unseen long-horizon tasks. We compare our method against DAF in the planning experiments (Sec. VII-B).

C. Task and motion planning

TAMP solves problems that require both symbolic and geometric reasoning [2, 57]. DAF learns a skill proposal network to replace the typical task planner in TAMP, akin to [58]. Another prominent line of research learns components of the TAMP system, often from a dataset of precomputed solutions [59–64]. The problems we consider involve complex geometric dependencies between actions that are typical in TAMP. However, STAP only performs geometric reasoning and by itself is not a TAMP method. We demonstrate in experiments (Sec. VII-C) that STAP can be combined with symbolic planners to solve TAMP problems.

III. Problem Setup

A. Long-horizon planning

Our objective is to solve long-horizon manipulation tasks that require sequential execution of learned skills. These skills come from a skill library $\mathcal{L} = \{\psi^1, \ldots, \psi^K\}$, where each skill ψ^k consists of a parameterized manipulation primitive [10] ϕ^k and a learned policy π^k. A primitive $\phi^k\left(a^k\right)$ takes in parameters a^k and executes a series of motor commands on the robot, while a policy $\pi^k\left(a^k \mid s^k\right)$ is trained to predict a distribution of suitable parameters a^k from the current state s^k. For example, the Pick(a, b) skill may have a primitive which takes as input an end-effector pose and executes a trajectory to pick up object a, where the robot first moves to the commanded pose, closes the gripper to grasp a, and then lifts a off of b. The learned policy π^k for this skill will then try to predict end-effector poses to pick up a.

We assume access to a high-level planner that computes *plan skeletons* (i.e. skill sequences) to achieve a high-level goal. STAP aims to solve the problem of turning plan skeletons into geometrically feasible *action plans* (i.e. parameters for each manipulation primitive in the plan skeleton).

STAP is agnostic to the choice of high-level planner. For instance, it can be used in conjunction with Planning Domain Definition Language (PDDL) [65] task planners to perform hierarchical TAMP [66]. In this setup, the task planner and STAP will be queried numerous times to find multiple plan skeletons grounded with optimized action plans. STAP will also evaluate each action plan's probability of success (i.e. its geometric feasibility). After some termination criterion is met, such as a timeout, the candidate plan skeleton and action plan with the highest probability of success is returned.

B. Task-agnostic policies

We aim to learn policies $\{\pi^1, \ldots, \pi^K\}$ for the skill library $\mathcal{L}$ that can be sequenced by a high-level planner in arbitrary

ways to solve any long-horizon task. We call these policies task-agnostic because they are not trained to solve a specific long-horizon task. Instead, each policy π^k is associated with a skill-specific contextual bandit (i.e. a single timestep MDP)

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

where $\mathcal{S}^k$ is the state space, $\mathcal{A}^k$ is the action space, $T^k\left(s'^k \mid s^k, a^k\right)$ is the transition model, $R^k\left(s^k, a^k, s'^k\right)$ is the binary reward function, and $\rho^k\left(s^k\right)$ is the initial state distribution. Given a state s^k, the policy π^k produces an action a^k, and the state evolves according to the transition model $T^k\left(s'^k \mid s^k, a^k\right)$. Thus, the transition model encapsulates the execution of the manipulation primitive ϕ^k (Sec. III-A).

A long-horizon domain is one in which each timestep involves the execution of a single policy, and it is specified by

$$\overline{\mathcal{M}} = \left(\mathcal{M}^{1:K}, \overline{\mathcal{S}}, \overline{T}^{1:K}, \overline{\rho}^{1:K}, \Gamma^{1:K}\right), \tag{2}$$

where $\mathcal{M}^{1:K}$ is the set of MDPs whose policies can be executed in the long-horizon domain, $\overline{\mathcal{S}}$ is the state space of the long-horizon domain, $\overline{T}^k\left(\overline{s}' \mid \overline{s}, a^k\right)$ is an extension of dynamics $T^k\left(s'^k \mid s^k, a^k\right)$ that models how the entire long-horizon state evolves with action a^k, $\overline{\rho}^k\left(\overline{s}\right)$ is an extension of initial state distributions $\rho^k\left(s^k\right)$ over the long-horizon state space, and $\Gamma^k : \overline{\mathcal{S}} \to \mathcal{S}^k$ is a function that maps from the long-horizon state space to the state space of policy k. We assume that the dynamics $T^k\left(s'^k \mid s^k, a^k\right)$, $\overline{T}^k\left(\overline{s}' \mid \overline{s}, a^k\right)$ and initial state distributions $\rho^k\left(s^k\right)$, $\overline{\rho}^k\left(\overline{s}^k\right)$ are unknown.

Note that while the policies may have different state spaces $\mathcal{S}^k$, policy states s^k must be obtainable from the long-horizon state space $\overline{\mathcal{S}}$ via $s^k = \Gamma^k(\overline{s})$. This is to ensure that the policies can be used together in the same environment to perform long-horizon tasks. In the base case, all the state spaces are identical and Γ^k is simply the identity function. Another case is that $\overline{s}$ is constructed as the concatenation of all $s^{1:K}$ and $\Gamma^k(\overline{s})$ extracts the slice in $\overline{s}$ corresponding to s^k.

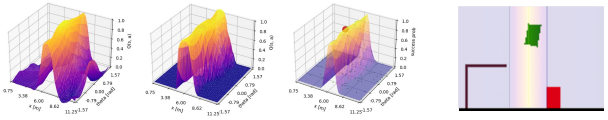
IV. Sequencing Task-Agnostic Policies

Given a task in the form of a sequence of skills to execute, our planning framework constructs an optimization problem with the policies, Q-functions, and dynamics models of each skill. Solving the optimization problem results in parameters for all manipulation primitives in the skill sequence such that the entire sequence's probability of success is maximized.

We formalize our planning methodology in this section and outline its implementation in Sec. V. Lastly, we describe our procedure for training modular skill libraries in Sec. VI.

A. Grounding skill sequences with action plans

We assume that we are given a plan skeleton of skills $\tau = [\psi_1, \ldots, \psi_H] \in \mathcal{L}^H$ (hereafter denoted by $\tau = \psi_{1:H}$) that should be successfully executed to solve a long-horizon task. Let $\mathcal{M}_h$ with subscript h denote the MDP corresponding to the h-th skill in the sequence—in contrast to $\mathcal{M}^k$ with superscript k, which denotes the k-th MDP in the skill library. A long-horizon task is considered successful if every skill reward $r_1, \ldots, r_H$ received during execution is 1.



(a) Place: $Q_1(s,a)$ (b) Push: $Q_2(s,a)$ (c) Objective: $Q_1 \cdot Q_2$ (d) Action: $\arg\max_a\ Q_1 \cdot Q_2$

Fig. 2: Planning in a 2D toy domain. The agent needs to get the green block under the brown receptacle with two skills: Place() and Push() that operate on the horizontal position x of the green block. Plots (a) and (b) show the Q-functions across (x, θ) for each skill. Place() is only trained to get the green block on the ground, so the planner must determine $a = x$ *s.t.* Push() is unobstructed. The optimal action maximizes the probability of long-horizon task success (Eq. 3), approximated by the product of Q-functions in plot (c).

Given an initial state $\overline{s}_1 \in \overline{\mathcal{S}}$, our problem is to ground the plan skeleton $\tau = \psi_{1:H}$ with an action plan $\xi = [a_1, \ldots, a_H] \in \mathcal{A}_1 \times \cdots \times \mathcal{A}_H$ that maximizes the probability of succeeding at the long-horizon task. This is framed as an optimization problem $\arg\max_{a_{1:H}} J$, where the maximization objective J is the task success probability

$$J(a_{1:H}; \overline{s}_1) = p\left(r_1 = 1, \ldots, r_H = 1 \mid \overline{s}_1, a_{1:H}\right).$$

Here, $r_{1:H}$ are the skill rewards received at each timestep.

With the long-horizon dynamics models $\overline{T}^k\left(\overline{s}' \mid \overline{s}, a^k\right)$, the objective can be cast as the expectation

$$J = \mathrm{E}_{\overline{s}_{2:H} \sim \overline{T}_{1:H-1}}\left[p\left(r_1 = 1, \ldots, r_H = 1 \mid \overline{s}_{1:H}, a_{1:H}\right)\right].$$

By the Markov assumption, rewards are conditionally independent given states and actions. We can express the probability of task success as the product of reward probabilities

$$J = \mathrm{E}_{\overline{s}_{2:H} \sim \overline{T}_{1:H-1}}\left[\Pi_{h=1}^{H}\ p\left(r_h = 1 \mid \overline{s}_h, a_h\right)\right].$$

Because the skill rewards are binary, the skill success probabilities are equivalent to Q-values:

$$\begin{aligned} p\left(r_h = 1 \mid \overline{s}_h, a_h\right) &= \mathrm{E}_{\overline{s}_{h+1} \sim \overline{T}_h}\left[r_h \mid \overline{s}_h, a_h\right] \\ &= Q_h\left(\Gamma_h\left(\overline{s}_h\right), a_h\right). \end{aligned}$$

The final objective is expressed in terms of Q-values:

$$J = \mathrm{E}_{\overline{s}_{2:H} \sim \overline{T}_{1:H-1}}\left[\Pi_{h=1}^{H}\ Q_h\left(\Gamma_h\left(\overline{s}_h\right), a_h\right)\right]. \tag{3}$$

This planning objective is simply the product of Q-values evaluated along the trajectory $(\overline{s}_1, a_1, \ldots, \overline{s}_H, a_H)$, where the states are predicted by the long-horizon dynamics model: $\overline{s}_2 \sim \overline{T}_1\left(\cdot \mid \overline{s}_1, a_1\right), \ldots, \overline{s}_H \sim \overline{T}_{H-1}\left(\cdot \mid \overline{s}_{H-1}, a_{H-1}\right)$.[1]

B. Ensuring action plan feasibility

A plan skeleton $\tau = \psi_{1:H}$ is feasible only if, for every pair of consecutive skills ψ_i and ψ_j, there is a non-zero overlap between the terminal state distribution of i and the initial state distribution of j. More formally,

$$\mathrm{E}_{\overline{s}_i \sim \overline{\rho}_i, a_i \sim \mathcal{A}_i, \overline{s}_j \sim \overline{\rho}_j}\left[\overline{T}_i(\overline{s}_j \mid \overline{s}_i, a_i)\right] > 0, \tag{4}$$

[1]One might consider maximizing the *sum* of Q-values instead of the product, but this may not reflect the probability of task success. For example, if we want to optimize a sequence of ten skills, consider a plan that results in nine Q-values of 1 and one Q-value of 0, for a total sum of 9. One Q-value of 0 would indicate just one skill failure, but this is enough to cause a failure for the entire task. Compare this to a plan with ten Q-values of 0.9. This plan has an equivalent sum of 9, but it is preferable because it has a non-zero probability of succeeding.

where $\overline{\rho}_i$ and $\overline{\rho}_j$ are the initial state distributions for skills ψ_i and ψ_j, respectively, and a_i is uniformly distributed with respect to action space $\mathcal{A}_i$ for skill ψ_i. Given a state $\overline{s}_i \sim \overline{\rho}_i$, it is part of the planner's job to determine an action a_i that induces a valid subsequent state $\overline{s}_j \sim \overline{\rho}_j$ if one exists. Failing to do so constitutes an OOD event for skill ψ_j, where the state $\overline{s}_j$ has drifted beyond the region of the state space where $Q_j\left(\Gamma_j\left(\overline{s}_j\right), a_j\right)$ is well-defined and ψ_j is executable.

Neglecting state distributional shift over an action plan ξ may degrade the quality of objective function J with spuriously high Q-values (Eq. 3). Moreover, Eq. 4 cannot be explicitly computed to determine the validity of actions because the initial state distributions of all skills $\overline{\rho}^k\left(\overline{s}^k\right)$ are unknown. We can detect OOD states (and actions) by performing UQ on the Q-functions $Q^k(s^k, a^k)$. Filtering out Q-values with high uncertainty would result in action plans ξ that are robust (i.e. have low uncertainty) while maximizing the task feasibility objective. We discuss efficient methods for training UQ models on learned Q-functions in Sec. VI-C.

V. Planning Action Sequences

To find action sequences that maximize the probability of long-horizon task success (Eq. 3), we use sampling-based optimization techniques: shooting and cross-entropy method (CEM) [68]. In shooting, we simply sample action plans $\xi = a_{1:H} \in \mathcal{A}_1 \times \cdots \times \mathcal{A}_H$ and select the one with the highest predicted objective score. CEM is an extension of shooting that iteratively refines the action sampling distribution to fit a fraction of the population with the highest objective scores.

Sampling action plans from uniform distributions may be sufficient for small action spaces and short skill sequences. However, this strategy suffers from the curse of dimensionality and may not scale desirably to the large action spaces and long skill sequences that we consider. Meanwhile, directly executing actions $a^k \sim \pi^k\left(\cdot|s^k\right)$ from policies that are trained to solve skill-specific tasks produces myopic behavior that rarely succeeds for long-horizon tasks with complex geometric dependencies between actions.

The policies can be leveraged to initialize a sampling-based search by producing an action plan that is likely to be closer to an optimal plan than one sampled uniformly at random. We therefore use two variants of shooting and CEM, termed policy shooting and policy CEM, which sample actions from Gaussian distributions $a^k \sim \mathcal{N}\left(\pi^k\left(s^k\right), \sigma\right)$, where the mean is the action predicted by the policy and the standard deviation is a planning hyperparameter.

VI. Training Skills

A. Policies and Q-functions

One of the key advantages of our approach is that the policies can be trained independently and then composed at test time to solve unseen sequential tasks. For each skill ψ_k, we want to obtain a policy $\pi^k : \mathcal{S}^k \to \mathcal{A}^k$ that solves the task specified by the skill-specific MDP $\mathcal{M}^k$ (Eq. 1), along with a Q-function Q^k modeling the policy's expected success

$$Q^k(s^k, a^k) = \mathrm{E}_{s'^k \sim T^k(\cdot \mid s^k, a^k)}\left[R^k\left(s^k, a^k, s'^k\right)\right].$$

Our framework is agnostic to the method for acquiring the policy and Q-function. Many deep RL algorithms are able to simultaneously learn the policy (i.e. actor) and Q-function (i.e. critic) with unknown dynamics [69, 70]. We therefore leverage off-the-shelf RL algorithms to learn a policy and Q-function for each skill (Fig. 3 - Left (c)). In our experiments, we specifically use Soft Actor-Critic (SAC) [71]. For other policy acquisition methods, policy evaluation can be performed to obtain a Q-function after a policy has been learned.

B. Dynamics

The dynamics models are used to predict future states at which each downstream Q-function in the plan skeleton will be evaluated. We learn a deterministic model $\overline{T}^k\left(\overline{s}, a^k\right)$ for each skill ψ_k using the single-step forward prediction loss

$$L_{\text{dynamics}}\left(\overline{T}^k; \overline{s}, a^k, \overline{s}'\right) = \left\| \overline{T}^k\left(\overline{s}, a^k\right) - \overline{s}' \right\|_2^2.$$

Each dynamics model $\overline{T}^k$ is trained on the state transition experience $(\overline{s}, a^k, \overline{s}')$ collected during the training of policy π^k (Sec. VI-A), stored in the replay buffer $\mathcal{D}^k$ (Fig. 3 - Left (d)). Training our dynamics models on existing state transitions is efficient and circumvents the challenges associated with learning dynamics in the context of a long-horizon task [53].

C. Uncertainty quantification

Measuring the epistemic uncertainty over the Q-values allows us to identify when dynamics-predicted states and planned actions drift OOD for downstream critics Q^k. We leverage recent advances in neural network UQ to obtain an explicit Gaussian posterior predictive distribution

$$p\left(Q^k \mid s^k, a^k, \mathcal{D}^k; w^k\right) = \mathcal{N}\left(\mu_{Q^k}, \sigma_{Q^k}; w^k\right) \tag{5}$$

with sketching curvature for OOD detection (SCOD) [67]. SCOD computes the weights w^k that parameterizes the posterior distribution over each critic Q^k using only the experience $(\overline{s}, a^k) \sim \mathcal{D}^k$ collected over the course of training policy π^k (Fig. 3 - Left (e)). An advantage of SCOD over common UQ techniques [72, 73] is that it imposes no train-time dependencies on any algorithms used in our framework.

VII. Experiments

In our experiments, we test the following hypotheses:

- **H1** Maximizing the product of learned Q-functions (Eq. 3) translates to maximizing long-horizon task success.
- **H2** Skills trained with our framework are able to generalize to unseen long-horizon tasks by optimizing Eq. 3.
- **H3** Our planning method can be combined with a task planner and UQ (Eq. 5) to solve TAMP problems.

We evaluate our method on a 3D manipulation domain with 4 skills: Pick(a, b): pick a from b; Place(a, b): place a onto b; Pull$(a, hook)$: pull a into the robot's workspace with a $hook$; and Push$(a, hook)$: push a with a $hook$.

The long-horizon state space $\overline{\mathcal{S}}$ is a sequence of low-dimensional object states that contains information such as 6D poses. The policy state spaces $\mathcal{S}^k$ are constructed so that the first m object states correspond to the m arguments of

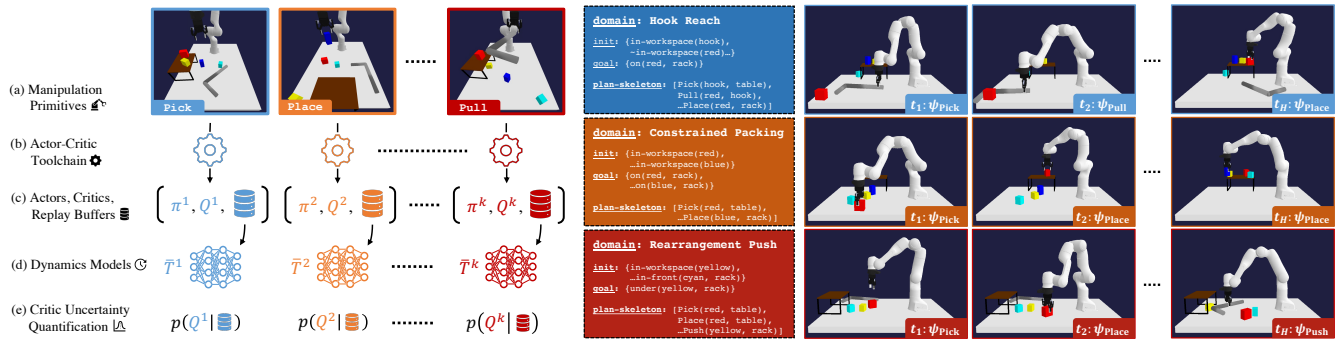


Fig. 3: **Left:** Training pipeline. We train each skill independently on single-step environments with skill-specific rewards. We use the experience collected by each skill to train (1) a dynamics model and (2) a SCOD [67] model to predict OOD critic inputs per skill. The benefit of training each skill independently is that we can easily add skills to the library or even mix skill acquisition strategies (e.g. RL, imitation learning, and handcrafted skills). Our planning framework ensures that the skills can be composed to solve any long-horizon task even if the skills were not explicitly trained to perform those tasks. **Right:** Example evaluation tasks. We evaluate our method on 9 tasks: 3 `Hook Reach` tasks, where the robot needs to use the hook to bring objects closer, 3 `Constrained Packing` tasks, where the robot needs to place blocks on the rack, and 3 `Rearrangement Push` tasks, where the robot needs to remove obstacles to push a target block under the rack. The 9 tasks feature a range of geometric complexities and plan skeleton lengths.

the corresponding skill. For example, a state for the policy of Pick$(box, rack)$ will contain first the box's state, then the $rack$'s state, followed by a random permutation of the remaining object states. The policy action spaces $\mathcal{A}^k$ are all 4D. For example, a policy-predicted action for Pick(a, b) specifies the 3D grasp position of the end-effector relative to the target a and orientation about the world z-axis.

Our evaluation is on 9 different long-horizon tasks (i.e. plan skeletons τ). The tasks cover a range of symbolic and geometric complexities (Fig. 3 - Right), with plan skeleton lengths ranging from 4 to 10 skills. Each task involves geometric dependencies between actions, which motivates the need for planning. We use 100 randomly generated instances (i.e. object configurations) for evaluation on each task.

A. *Product of Q-functions approximates task success (H1)*

We test **H1** by comparing STAP to an **Oracle** baseline that runs forward simulations with policy shooting to find action plans that achieve ground-truth task success. Our method uses learned Q-functions and dynamics to predict task success as the product of Q-functions. We expect that planning with this objective will come close to matching the task success upper bound provided by **Oracle**.

We compare several planning methods: **Policy Shooting** and **Policy CEM**, which use the learned policies to initialize the action sampling distributions (Sec. V), as well as **Random Shooting** and **Random CEM**, which use uniform action priors. We also compare with **Greedy**, which does not plan but greedily executes the skills. The evaluation metrics are ground-truth task success, sub-goal completion rate (what percentage of skills in a plan are successfully executed), and predicted task success computed from Eq. 3.

Due to the significant amount of time required to run forward simulations for **Oracle**, we limit the number of sampled trajectories evaluated during planning to 1000 for all methods. This is not enough to succeed at the most complex tasks, and thus, we evaluate on the simplest task from the `Hook Reach` and `Constrained Packing` domains.

The results from both tasks are averaged and presented in Fig. 4. As expected, **Oracle** achieves the highest success rate,

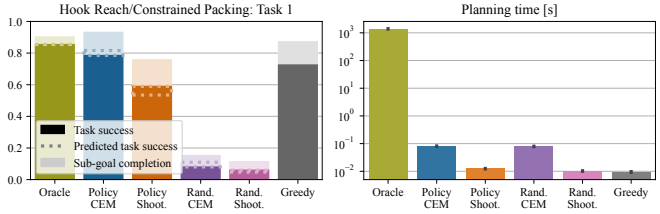


Fig. 4: A small-scale experiment comparing the performance of our method to **Oracle** planning. The left plot shows the average success rates across two domains (`Hook Reach` and `Constrained Packing`). The dark bars indicate the ground truth task success, and the light bars indicate sub-goal completion rate, which measures how close the plan was to successfully completing the task. The predicted task success computed from the product of Q-values is indicated by a dotted line. Our method with **Policy CEM** is able to nearly match the success rate of **Oracle** while taking 4 orders of magnitude less time, as shown in the plot on the right.

although not perfect because 1000 samples are not enough to solve all of the tasks. **Policy CEM** nearly matches **Oracle**'s success rate, which demonstrates that maximizing the product of Q-functions is a good proxy for maximizing task success. **Policy CEM** also exhibits a low success prediction error, which demonstrates that the learned Q-functions and dynamics generalize well to these unseen long-horizon tasks. Meanwhile, planning with these learned models runs 4 orders of magnitude faster than **Oracle** and does not require ground-truth knowledge about the environment state or dynamics.

Policy Shooting performs slightly worse than **Policy CEM**, which demonstrates CEM's strength in finding local maxima through iterative refinement. **Random CEM** and **Random Shooting** perform quite poorly, indicating that the planning space is too large (16D for these tasks) for random sampling. **Greedy** performs strongly, perhaps indicating that these simpler tasks can be solved without planning.

B. *STAP skills generalize to long-horizon tasks (H2)*

In this experiment, we test the ability of our framework to solve 9 long-horizon tasks with geometric dependencies between actions. We compare against DAF [7], a state-of-the-art method for learning to solve TAMP problems. As task planning is outside the scope of this paper, we omit DAF's skill proposal network and compare only to the skills trained with DAF (**DAF-Skills**), which are comprised of dynamics

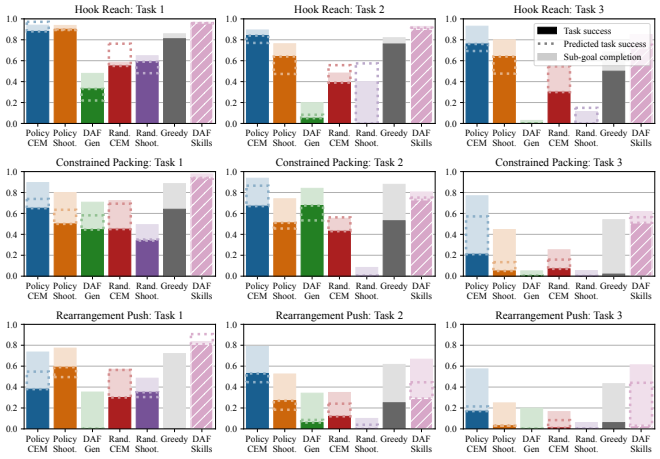


Fig. 5: Planning experiment with 3 domains, each with 3 tasks. Our method with **Policy CEM** is able to generalize to all of these tasks without ever seeing them during training. On 6 out of the 9 tasks, our method either matches or outperforms **DAF-Skills**, which is trained directly on the evaluation task. **DAF-Gen** shows the generalization performance of the **DAF-Skills** models when evaluated on unseen tasks within the same domain.

and affordance models. DAF's planning objective is similar to ours, except that it evaluates the product of affordances rather than Q-functions. We give **DAF-Skills** the same plan skeleton τ that is given to our method and augment DAF's shooting planner with CEM for a more even comparison.

Like other model-based RL methods, DAF requires training on a set of long-horizon tasks that is representative of the evaluation task distribution. We therefore train one **DAF-Skills** model per task (9 total) and run evaluation on the same task. We also test the ability of these models to generalize to the other two tasks within the same domain (**DAF-Gen**). Since **DAF-Skills** is trained on its evaluation task, we expect it to perform at least as well as STAP, if not better. However, we expect **DAF-Gen** to perform slightly worse than STAP, since the evaluation tasks differ from the training tasks, even if they are similar. We train all models for 48 hours each and allow 1000 samples per dimension for planning.

The results are presented in Fig. 5. Our method with **Policy CEM** achieves competitive success rates with **DAF-Skills** on 4 out of the 9 tasks and outperforms it on 2 tasks with highly complex action dependencies (`Rearrangement Push`). While **DAF-Gen** matches the performance of **Policy CEM** on 2 tasks, it gets relatively low success on the others. This indicates that skills trained on one long-horizon task may not effectively transfer to other tasks with similar action dependencies. Our method of training skills in independent environments and then generalizing to long-horizon tasks via planning is efficient from a training perspective, since the same trained skills can be used for all downstream tasks.

C. STAP can be extended for TAMP with UQ (*H3*)

In this experiment, we combine our framework with a PDDL task planner as described in Sec. III-A and evaluate it on two TAMP problems. In `Hook Reach`, the robot needs to decide the best way to pick up a block, which may or may not be in its workspace. In `Constrained Packing`, the robot needs to place a fixed number of objects on the rack

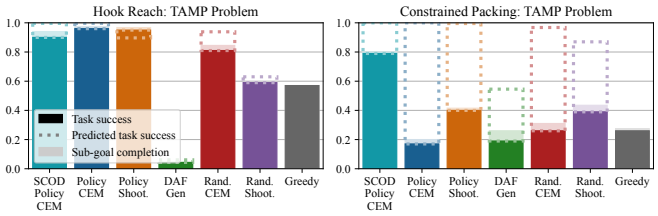


Fig. 6: Integration of our planning framework with task planning and UQ to solve TAMP problems. The poor performance of the methods without SCOD in the `Constrained Packing` problem highlights the importance of UQ when attempting to solve unseen TAMP problems with learned skills.

but is free to choose among any of the objects on the table. To mimic what the robot might find in an unstructured, real-world environment, some of these objects are distractor objects that are initialized in ways not seen by the skills during training (e.g. the blocks can be stacked, placed behind the robot base, or tipped over). The task planner may end up selecting these distractor objects for placing on the rack, but since the skills have not been trained to handle these objects, their predicted success (Q-values) may be unreliable. UQ is particularly important for such scenarios, so we introduce **SCOD Policy CEM**, which filters out candidate action plans with high uncertainty in the predicted task success score (Eq. 3). That is, the n action plans with the highest skill uncertainties (Eq. 5) are not considered for execution.

The results are presented in Fig. 6. **Policy CEM** achieves 97% success on the `Hook Reach` TAMP problem, while **SCOD Policy CEM** suffers a slight performance drop. However, for `Constrained Packing`, which contains OOD states, **SCOD Policy CEM** strongly outperforms the other methods. Exploring different ways to integrate UQ into our planning framework is a promising direction for future work.

D. Real world sequential manipulation

We demonstrate that skills trained with our framework can be used to perform sequential manipulation tasks in a real robot environment. We take RGB-D images from a Kinect v2 camera and use manually tuned color thresholds to segment objects in the scene. With these segmentations, we estimate object poses using the depth image, which is then used to construct the initial environment state $\overline{s}_1$. Qualitative results are provided in the supplementary video.

VIII. Conclusion

We present a framework for sequencing task-agnostic policies that have been trained independently. The key to generalization is planning actions that maximize the probability of long-horizon task success, which we model using the product of learned Q-values. This requires learning a dynamics model to predict future states and using UQ to filter out OOD states that the skills do not support. The result is a library of skills that can be composed to solve arbitrary long-horizon tasks with complex geometric dependencies between actions. Future work includes the investigation of methods for scaling skills to high-dimensional observations, combining the library of learned skills with a set of handcrafted skills, and exploring planning objectives that capture other desirable properties of trajectories, beyond their geometric feasibility.

REFERENCES

[1] S. M. Lavalle, *Planning Algorithms*. Cambridge University Press, 2006.

[2] M. Toussaint, "Logic-geometric programming: An optimization-based approach to combined task and motion planning," in *Twenty-Fourth International Joint Conference on Artificial Intelligence*, 2015.

[3] M. Toussaint, K. Allen, K. Smith, and J. B. Tenenbaum, "Differentiable physics and stable modes for tool-use and manipulation planning," in *Proceedings of Robotics: Science and Systems*, Pittsburgh, Pennsylvania, June 2018.

[4] L. Kaelbling, M. Littman, and A. Moore, "Reinforcement learning: A survey," *Journal of Artificial Intelligence Research*, vol. 4, pp. 237–285, 1996.

[5] B. D. Argall, S. Chernova, M. Veloso, and B. Browning, "A survey of robot learning from demonstration," *Robotics and autonomous systems*, vol. 57, no. 5, pp. 469–483, 2009.

[6] S. Schaal, "Is imitation learning the route to humanoid robots?" *Trends in Cognitive Sciences*, vol. 3, no. 6, pp. 233–242, 1999.

[7] D. Xu, A. Mandlekar, R. Martín-Martín, Y. Zhu, S. Savarese, and L. Fei-Fei, "Deep affordance foresight: Planning through what can be done in the future," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 6206–6213.

[8] M. Dalal, D. Pathak, and R. R. Salakhutdinov, "Accelerating robotic reinforcement learning via parameterized action primitives," *Advances in Neural Information Processing Systems*, vol. 34, pp. 21 847–21 859, 2021.

[9] B. Da Silva, G. Konidaris, and A. Barto, "Learning parameterized skills," *International Conference on Machine Learning (ICML)*, 2012.

[10] J. Felip, J. Laaksonen, A. Morales, and V. Kyrki, "Manipulation primitives: A paradigm for abstraction and execution of grasping and manipulation tasks," *Robotics and Autonomous Systems*, vol. 61, no. 3, pp. 283–296, 2013.

[11] S. Schaal, "Dynamic movement primitives-a framework for motor control in humans and humanoid robotics," in *Adaptive motion of animals and machines*. Springer, 2006, pp. 261–280.

[12] P. Pastor, H. Hoffmann, T. Asfour, and S. Schaal, "Learning and generalization of motor skills by learning from demonstration," in *2009 IEEE International Conference on Robotics and Automation*. IEEE, 2009, pp. 763–768.

[13] S. M. Khansari-Zadeh and A. Billard, "Learning stable nonlinear dynamical systems with gaussian mixture models," *IEEE Transactions on Robotics*, vol. 27, no. 5, pp. 943–957, 2011.

[14] A. J. Ijspeert, J. Nakanishi, and S. Schaal, "Movement imitation with nonlinear dynamical systems in humanoid robots," in *Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No. 02CH37292)*, vol. 2. IEEE, 2002, pp. 1398–1403.

[15] T. Matsubara, S.-H. Hyon, and J. Morimoto, "Learning stylistic dynamic movement primitives from multiple demonstrations," in *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2010, pp. 1277–1283.

[16] A. Ude, A. Gams, T. Asfour, and J. Morimoto, "Task-specific generalization of discrete and periodic dynamic movement primitives," *IEEE Transactions on Robotics*, vol. 26, no. 5, pp. 800–815, 2010.

[17] T. Kulvicius, K. Ning, M. Tamosiunaite, and F. Worgötter, "Joining movement sequences: Modified dynamic movement primitives for robotics applications exemplified on handwriting," *IEEE Transactions on Robotics*, vol. 28, no. 1, pp. 145–157, 2011.

[18] S. Bahl, M. Mukadam, A. Gupta, and D. Pathak, "Neural dynamic policies for end-to-end sensorimotor learning," *Advances in Neural Information Processing Systems*, vol. 33, pp. 5058–5069, 2020.

[19] S. Bahl, A. Gupta, and D. Pathak, "Hierarchical neural dynamic policies," *Robotics: Science and Systems*, 2021.

[20] L. Shao, T. Migimatsu, Q. Zhang, K. Yang, and J. Bohg, "Concept2robot: Learning manipulation concepts from instructions and human demonstrations," *The International Journal of Robotics Research*, vol. 40, no. 12-14, pp. 1419–1434, 2021.

[21] T. Shankar, S. Tulsiani, L. Pinto, and A. Gupta, "Discovering motor programs by recomposing demonstrations," in *International Conference on Learning Representations*, 2019.

[22] T. Shankar and A. Gupta, "Learning robot skills with temporal variational inference," in *International Conference on Machine Learning*. PMLR, 2020, pp. 8624–8633.

[23] A. Ajay, A. Kumar, P. Agrawal, S. Levine, and O. Nachum, "Opal: Offline primitive discovery for accelerating offline reinforcement learning," *International Conference on Learning Representations (ICLR)*, 2021.

[24] A. Singh, H. Liu, G. Zhou, A. Yu, N. Rhinehart, and S. Levine, "Parrot: Data-driven behavioral priors for reinforcement learning," *International Conference on Learning Representations (ICLR)*, 2021.

[25] K. Pertsch, Y. Lee, and J. J. Lim, "Accelerating reinforcement learning with learned skill priors," *Conference on Robot Learning (CoRL)*, 2020.

[26] B. Da Silva, G. Konidaris, and A. Barto, "Active learning of parameterized skills," in *International Conference on Machine Learning*. PMLR, 2014, pp. 1737–1745.

[27] A. Rajeswaran, V. Kumar, A. Gupta, G. Vezzani, J. Schulman, E. Todorov, and S. Levine, "Learning Complex Dexterous Manipulation with Deep Reinforcement Learning and Demonstrations," in *Proceedings of Robotics: Science and Systems (RSS)*, 2018.

[28] D. Kalashnikov, A. Irpan, P. Pastor, J. Ibarz, A. Herzog, E. Jang, D. Quillen, E. Holly, M. Kalakrishnan, V. Vanhoucke *et al.*, "Scalable deep reinforcement learning for vision-based robotic manipulation," in *Conference on Robot Learning*. PMLR, 2018, pp. 651–673.

[29] D. Kalashnikov, J. Varley, Y. Chebotar, B. Swanson, R. Jonschkowski, C. Finn, S. Levine, and K. Hausman, "Mt-opt: Continuous multi-task robotic reinforcement learning at scale," *arXiv preprint arXiv:2104.08212*, 2021.

[30] A. Sharma, S. Gu, S. Levine, V. Kumar, and K. Hausman, "Dynamics-aware unsupervised discovery of skills," *International Conference on Learning Representations (ICLR)*, 2020.

[31] Y. Lu, K. Hausman, Y. Chebotar, M. Yan, E. Jang, A. Herzog, T. Xiao, A. Irpan, M. Khansari, D. Kalashnikov *et al.*, "Aw-opt: Learning robotic skills with imitation andreinforcement at scale," in *Conference on Robot Learning*. PMLR, 2022, pp. 1078–1088.

[32] Y. Chebotar, K. Hausman, Y. Lu, T. Xiao, D. Kalashnikov, J. Varley, A. Irpan, B. Eysenbach, R. Julian, C. Finn *et al.*, "Actionable models: Unsupervised offline reinforcement learning of robotic skills," *International Conference on Machine Learning*, 2021.

[33] D. Kappler, P. Pastor, M. Kalakrishnan, M. Wüthrich, and S. Schaal, "Data-driven online decision making for autonomous manipulation." in *Robotics: science and systems*, vol. 11, 2015.

[34] L. P. Kaelbling and T. Lozano-Pérez, "Learning composable models of parameterized skills," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2017, pp. 886–893.

[35] B. Ames, A. Thackston, and G. Konidaris, "Learning symbolic representations for planning with parameterized skills," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 526–533.

[36] T. Migimatsu, W. Lian, J. Bohg, and S. Schaal, "Symbolic state estimation with predicates for contact-rich manipulation tasks," in *IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2022.

[37] B. Wu, S. Nair, L. Fei-Fei, and C. Finn, "Example-driven model-based reinforcement learning for solving long-horizon visuomotor tasks," *5th Conference on Robot Learning (CoRL)*, 2021.

[38] D.-A. Huang, D. Xu, Y. Zhu, A. Garg, S. Savarese, L. Fei-Fei, and J. C. Niebles, "Continuous relaxation of symbolic planner for one-shot imitation learning," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2019, pp. 2635–2642.

[39] A. Simeonov, Y. Du, B. Kim, F. R. Hogan, J. Tenenbaum, P. Agrawal, and A. Rodriguez, "A long horizon planning framework for manipulating rigid pointcloud objects," *Conference on Robot Learning (CoRL)*, 2020.

[40] D. Xu, S. Nair, Y. Zhu, J. Gao, A. Garg, L. Fei-Fei, and S. Savarese, "Neural task programming: Learning to generalize across hierarchical tasks," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 3795–3802.

[41] D.-A. Huang, S. Nair, D. Xu, Y. Zhu, A. Garg, L. Fei-Fei, S. Savarese, and J. C. Niebles, "Neural task graphs: Generalizing to unseen tasks from a single video demonstration," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 8565–8574.

[42] R. S. Sutton, D. Precup, and S. Singh, "Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning," *Artificial intelligence*, vol. 112, no. 1-2, pp. 181–211, 1999.

[43] W. Masson, P. Ranchod, and G. Konidaris, "Reinforcement learning with parameterized actions," in *Thirtieth AAAI Conference on Artificial Intelligence*, 2016.

[44] P.-L. Bacon, J. Harb, and D. Precup, "The option-critic architecture," in *Proceedings of the AAAI Conference on Artificial Intelligence*,

vol. 31, no. 1, 2017.

[45] O. Nachum, S. S. Gu, H. Lee, and S. Levine, "Data-efficient hierarchical reinforcement learning," *Advances in neural information processing systems*, vol. 31, 2018.

[46] R. Chitnis, S. Tulsiani, S. Gupta, and A. Gupta, "Efficient bimanual manipulation using learned task schemas," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 1149–1155.

[47] N. Vuong, H. Pham, and Q.-C. Pham, "Learning sequences of manipulation primitives for robotic assembly," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 4086–4092.

[48] S. Nasiriany, H. Liu, and Y. Zhu, "Augmenting reinforcement learning with behavior primitives for diverse manipulation tasks," in *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, 2022, pp. 7477–7484.

[49] D. Shah, P. Xu, Y. Lu, T. Xiao, A. Toshev, S. Levine, and B. Ichter, "Value function spaces: Skill-centric state abstractions for long-horizon reasoning," *International Conference on Learning Representations (ICLR)*, 2022.

[50] C. Finn and S. Levine, "Deep visual foresight for planning robot motion," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2017, pp. 2786–2793.

[51] K. Chua, R. Calandra, R. McAllister, and S. Levine, "Deep reinforcement learning in a handful of trials using probabilistic dynamics models," *Advances in neural information processing systems*, vol. 31, 2018.

[52] D. Hafner, T. Lillicrap, I. Fischer, R. Villegas, D. Ha, H. Lee, and J. Davidson, "Learning latent dynamics for planning from pixels," in *International conference on machine learning*. PMLR, 2019, pp. 2555–2565.

[53] M. Janner, J. Fu, M. Zhang, and S. Levine, "When to trust your model: Model-based policy optimization," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[54] D. Hafner, T. Lillicrap, J. Ba, and M. Norouzi, "Dream to control: Learning behaviors by latent imagination," *International Conference on Learning Representations (ICLR)*, 2020.

[55] K. Xie, H. Bharadhwaj, D. Hafner, A. Garg, and F. Shkurti, "Skill transfer via partially amortized hierarchical planning," in *International Conference on Learning Representations (ICLR)*, 2021.

[56] R. Sekar, O. Rybkin, K. Daniilidis, P. Abbeel, D. Hafner, and D. Pathak, "Planning to explore via self-supervised world models," in *International Conference on Machine Learning*. PMLR, 2020, pp. 8583–8592.

[57] C. R. Garrett, R. Chitnis, R. Holladay, B. Kim, T. Silver, L. P. Kaelbling, and T. Lozano-Pérez, "Integrated task and motion planning," *Annual review of control, robotics, and autonomous systems*, vol. 4, pp. 265–293, 2021.

[58] C. Wang, D. Xu, and L. Fei-Fei, "Generalizable task planning through representation pretraining," *IEEE Robotics and Automation Letters*, vol. 7, no. 3, 2022.

[59] Z. Wang, C. R. Garrett, L. P. Kaelbling, and T. Lozano-Pérez, "Active model learning and diverse action sampling for task and motion planning," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 4107–4114.

[60] D. Driess, J.-S. Ha, and M. Toussaint, "Deep visual reasoning: Learning to predict action sequences for task and motion planning from an initial scene image," *Robotics: Science and Systems*, 2020.

[61] R. Chitnis, T. Silver, B. Kim, L. P. Kaelbling, and T. Lozano-Perez, "Camps: Learning context-specific abstractions for efficient planning in factored mdps," *4th Conference on Robot Learning (CoRL)*, 2020.

[62] T. Silver, R. Chitnis, A. Curtis, J. B. Tenenbaum, T. Lozano-Perez, and L. P. Kaelbling, "Planning with learned object importance in large problem instances using graph neural networks," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 35, no. 13, 2021, pp. 11 962–11 971.

[63] D. Driess, J.-S. Ha, R. Tedrake, and M. Toussaint, "Learning geometric reasoning and control for long-horizon tasks from visual input," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 14 298–14 305.

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

[65] D. McDermott, M. Ghallab, A. Howe, C. Knoblock, A. Ram, M. Veloso, D. Weld, and D. Wilkins, *PDDL—The Planning Domain Definition Language Version 1.2*, 1998.

[66] L. P. Kaelbling and T. Lozano-Pérez, "Hierarchical task and motion planning in the now," in *2011 IEEE International Conference on Robotics and Automation*. IEEE, 2011, pp. 1470–1477.

[67] A. Sharma, N. Azizan, and M. Pavone, "Sketching curvature for efficient out-of-distribution detection for deep neural networks," in *Uncertainty in Artificial Intelligence*. PMLR, 2021, pp. 1958–1967.

[68] R. Rubinstein, "The cross-entropy method for combinatorial and continuous optimization," *Methodology and computing in applied probability*, vol. 1, no. 2, pp. 127–190, 1999.

[69] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *CoRR*, 2015.

[70] S. Fujimoto, H. Hoof, and D. Meger, "Addressing function approximation error in actor-critic methods," in *International conference on machine learning*. PMLR, 2018, pp. 1587–1596.

[71] T. Haarnoja, A. Zhou, K. Hartikainen, G. Tucker, S. Ha, J. Tan, V. Kumar, H. Zhu, A. Gupta, P. Abbeel *et al.*, "Soft actor-critic algorithms and applications," *International Conference on Machine Learning*, 2018.

[72] M. Ganaie, M. Hu, A. Malik, M. Tanveer, and P. Suganthan, "Ensemble deep learning: A review," *Engineering Applications of Artificial Intelligence*, vol. 115, p. 105151, 2022.

[73] Y. Gal and Z. Ghahramani, "Dropout as a bayesian approximation: Representing model uncertainty in deep learning," in *international conference on machine learning*. PMLR, 2016, pp. 1050–1059.

[74] J. Hoffmann, "Ff: The fast-forward planning system," *AI magazine*, vol. 22, no. 3, pp. 57–57, 2001.

[75] K. Lin, C. Agia, T. Migimatsu, M. Pavone, and J. Bohg, "Text2motion: From natural language instructions to feasible plans," *arXiv preprint arXiv:2303.12153*, 2023.

[76] B. Wu, S. Nair, R. Martin-Martin, L. Fei-Fei, and C. Finn, "Greedy hierarchical variational autoencoders for large-scale video prediction," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 2318–2328.

[77] S. Nair, A. Rajeswaran, V. Kumar, C. Finn, and A. Gupta, "R3m: A universal visual representation for robot manipulation," *arXiv preprint arXiv:2203.12601*, 2022.

[78] S. Karamcheti, S. Nair, A. S. Chen, T. Kollar, C. Finn, D. Sadigh, and P. Liang, "Language-driven representation learning for robotics," *arXiv preprint arXiv:2302.12766*, 2023.

[79] D. Brown, S. Niekum, and M. Petrik, "Bayesian robust optimization for imitation learning," *Advances in Neural Information Processing Systems*, vol. 33, pp. 2479–2491, 2020.

APPENDIX – **STAP**: SEQUENCING TASK-AGNOSTIC POLICIES

The appendix discusses commonly asked questions about our planning framework and provides details on the manipulation skill library used for planning. Qualitative results and code are made available at sites.google.com/stanford.edu/stap.

APPENDIX A
FREQUENTLY ASKED QUESTIONS

Q1: *What enables our planning framework to generalize to unseen tasks?*

A core hypothesis of this paper is that it is easier to learn *task-agnostic skills* that support long-horizon reasoning than it is to learn a single *long-horizon policy* that can generalize to arbitrary tasks (i.e. skill sequences). This hypothesis is motivated by two facts: 1) the state-action space associated with all possible long-horizon skill sequences grows exponentially $O(c^H)$ with the length H of the skill sequences considered, where c is some constant; 2) adequately exploring this exponentially sized state-action space during the training of a single long-horizon policy, as required to solve arbitrary tasks, is challenging from both a methodology and engineering standpoint. In contrast, the state-action that must be sufficiently explored to plan with STAP grows only linearly $O(K)$ with the number of task-agnostic skills K in the skill library. However, it is essential that STAP's task-agnostic skills are trained (at least in-part) on states that are likely to occur when solving tasks of interest.

Having learned a library of task-agnostic skills, our method plans with the skills at test-time to maximize the feasibility of a specified sequence of skills. Because the skills have been trained independent of each other and of the planner, every specified sequence of skills can be regarded as an *unseen task* that STAP must generalize to. Our method accomplishes this via optimization of a planning objective (see Sec. IV-A) in a process that involves the skills' policies, Q-functions, and dynamics models. Thus, the generality of our method to unseen tasks stems from the compositionality of independently learned skills.

Q2: *Why is STAP useful for Integrated Task and Motion Planning?*

Task and Motion Planning (TAMP) seeks to solve long-horizon tasks by integrating symbolic and geometric reasoning. Symbolic task planners and PDDL are commonly used to produce candidate plan skeletons or skill sequences that satisfy a user-specified symbolic goal. Robotics subroutines, e.g. motion planners, collision checkers, inverse kinematics solvers, are then procedurally invoked to verify the feasibility of the plan skeleton and return a corresponding motion plan.

Different than prior work, STAP presents an avenue to develop general TAMP algorithms centered around learned skills. While in Sec. III-A, we use STAP to perform geometric reasoning on candidate skill sequences proposed by a symbolic planner, many other instantiations are possible. For example, success probabilities (Eq. 3) predicted by STAP can serve as a geometric feasibility heuristic to guide task planning with classical search algorithms [74] or foundation models [75]. Such TAMP algorithms would inherit the efficiency of planning with STAP, as shown in Fig. 4. They would also benefit from the modularity associated with skill libraries, where new skills can be added to support a larger set of tasks and old skills can be updated to improve overall planning performance without the need to modify any other components of the TAMP framework.

Q3: *What would it take to scale STAP to high-dimensional observation spaces?*

Our framework is not limited to the low-dimensional state space described in Sec. VII, however, several challenges must be addressed in order to use STAP in conjunction with high-dimensional sensory data such as images or 3D point clouds.

- **Skills:** Q-functions must accurately characterize the skill's success probability given the high-dimensional observation. Challenge: the fidelity of skill Q-functions obtained via model-free Reinforcement Learning (RL) [70, 71] may be too low for long-horizon planning. Potential solution(s): acquire skills from large-scale datasets and couple controllable, data-driven learning methods (e.g. offline RL, imitation learning, supervised learning) with data augmentation techniques.
- **Dynamics:** The planning state space (Eq. 2) must be amenable to accurate forward prediction over long-horizon skill sequences. Challenge: predicted high-dimensional states become coarse over long-horizons [50, 76] which complicates their use in manipulation planning settings that demand fine-grained geometric detail. Potential solution(s): leverage pretrained representations for robotics [77, 78] and learn latent dynamic models [7, 54] for forward prediction.

Since STAP is reliant on the quality of the underlying skill library, we expect the capabilities of our method to improve with advancements in robot skill acquisition and visuomotor policies, representation learning, and video prediction for robotics.

Q4: *How else can uncertainty quantification be incorporated into planning?*

In our TAMP experiments (Sec. V), we take a filtering-based approach to robustify STAP planning in out-of-distribution scenarios. Specifically, we use Sketching Curvature for Out-of-Distribution Detection (SCOD) [67] for uncertainty quantification (UQ) of Q-functions and disregard the n plans with the most uncertain Q-values at each planning iteration.

While SCOD imposes no train-time dependencies on any algorithms in our framework, its forward pass is computationally and memory intensive and slows planning as a result. For faster planning, UQ alternatives such as deep ensembles [72] or

Monte-Carlo dropout [73] can be employed which, in contrast to SCOD, require modifying the algorithms used to learn skills. We further note that the described filtering-based optimization approach can be substituted with more sophisticated planning techniques, several of which have been implemented and verified to work with STAP. For example, we could formulate a distributionally robust variant of our planning objective (Eq. 3) to optimize a lower-confidence bound of the Q-values:

$$J(a_{1:H}; \overline{s}_1) = \mathrm{E}_{\overline{s}_{2:H} \sim \overline{T}_{1:H-1}} \left[\Pi_{h=1}^{H} \; Q_h \left(\Gamma_h \left(\overline{s}_h \right), a_h \right) - \alpha F_{\text{unc}}(\overline{s}_h, a_h; Q_h, \Gamma_h) \right],$$

where F_{unc} quantifies the uncertainty of Q-function Q_h evaluated at $s_h = \Gamma(\overline{s}_h)$ and a_h, and α is a scalar hyperparameter (e.g. the z-score). The uncertainty function F_{unc} is determined by the choice of UQ method. For deep ensembles, this would correspond to the empirical variance of the ensemble predictions $F_{\text{unc}}(\overline{s}_h, a_h; Q_h, \Gamma_h) = \mathrm{Var}\left[Q_h(\Gamma_h(\overline{s}_h), a_h)\right]$. Other choices of F_{unc} include posterior predictive variances provided by SCOD or coherent risk measures such as Conditional Value at Risk (CVaR) [79]. We have experimetally validated such formulations of STAP, and ultimately, the appropriate choice of planning objective and optimization scheme should correspond to the evaluation tasks and domains of interest.

Q5: *What are the main limitations of our method?*

Currently, our method is limited in settings with high degrees of partial observability and stochastic dynamics. Examples include mobile manipulation in unknown environments and interacting with dynamic agents, where it may be intractable to predict future states necessary for look-ahead planning. While these tasks are beyond the scope of this work, we note that our method is agnostic to the specific choice of skills and dynamics models, and thus, components that account for stochasticity could be used interchangeably. In this work, we focus on generalization to geometrically challenging manipulation problems.

APPENDIX B
MANIPULATION SKILL LIBRARY

A. Parameterized manipulation primitives

All variants of STAP interface with a library of manipulation skills $\mathcal{L} = \{\psi^1, \ldots, \psi^K\}$. Each skill ψ consists of a learned policy $\pi(a \mid s)$ and a parameterized manipulation primitive [10] $\phi(a)$. The policy is trained to output parameters $a \sim \pi(a \mid s)$ that results in successful actuation of the primitive $\phi(a)$ in a contextual bandit setting (Eq. 1) with a binary reward function $R(s, a, s')$. Our library consists of four skills, $\mathcal{L} = \{\psi^{\text{Pick}}, \psi^{\text{Place}}, \psi^{\text{Pull}}, \psi^{\text{Push}}\}$, used to solve tasks in simulation and in the real-world. The policies must learn to manipulate objects with different geometries (e.g. π^{Pick} is used for both Pick(box) and Pick$(hook)$). We describe the parameterization and reward function of each skill below. A reward of $r = 0$ is provided if any collision occurs with a non-argument object. For example, if ψ^{Place} collides with *rack* while executing Place$(box, table)$.

- **Pick**(obj): the parameter a represents the grasp pose of *obj* in the coordinate frame of *obj*. The policy π^{Pick} receives a reward of $r = 1$ if the primitive ϕ^{Pick} successfully grasps and picks up *obj*.
- **Place**(obj, rec): the parameter a represents the placement pose of *obj* in the coordinate frame of *rec*. The policy π^{Place} receives a reward of $r = 1$ if the primitive ϕ^{Place} places *obj* stable atop *rec*.
- **Pull**$(obj, tool)$: the parameter a represents the initial position, direction, and distance of a pull on *obj* with *tool* in the coordinate frame of *obj*. The policy π^{Pull} receives a reward of $r = 1$ if the primitive ϕ^{Pull} moves *obj* toward the robot by a minimum of $0.05m$.
- **Push**$(obj, tool, rec)$: the parameter a represents the initial position, direction, and distance of a push on *obj* with *tool* in the coordinate frame of *obj*. The policy π^{Push} receives a reward of $r = 1$ if the primitive ϕ^{Push} moves *obj* away from the robot by a minimum of $0.05m$ and if the final pose of *obj* is underneath *rec*.

B. Training manipulation skills

We use the Soft Actor-Critic [71] (SAC) algorithm with original hyperparameters to simultaneously learn a stochastic policy $\pi(a \mid s)$ and Q-function $Q^\pi(s, a)$ for each skill ψ. All models are trained for $200k$ single-step episodes and the (s, a, s') transitions are stored in a replay buffer $\mathcal{D}^\psi = \{(s^i, a^i, s'^i)\}_{i=1}^{200k}$ for each skill ψ. The replay buffer data is later used to train the dynamics models $\overline{T}^k(\overline{s}, a^k)$ (Sec. VI-B) and calibrate the weights w^k used by SCOD for UQ (Sec. VI-C).