Learning Contact-rich Abstractions using Tensor Factorization

Teng Xue[1,2], Amirreza Razmjoo[1,2], Suhan Shetty[1,2], Sylvain Calinon[1, 2]
[1]Idiap Research Institute [2]École Polytechnique Fédérale de Lausanne (EPFL)

***Abstract*—Contact-rich manipulation is non-trivial due to its under-actuated dynamics and hybrid contact modes, which lead to non-convex programs and require joint reasoning over both discrete and continuous variables. This presents significant challenges for current gradient-based and sampling-based methods, especially in long-horizon manipulation tasks with sparse rewards. Planning with effective abstractions is a promising approach to address these issues. However, identifying an effective abstraction of the geometric world that can facilitate motion-level planning and control remains an open question. In this work, we propose learning such abstractions through tensor factorization, which enables efficient planning and control in contact-rich scenarios. We validate the proposed method across three manipulation domains, encompassing both prehensile and non-prehensile primitives. The results demonstrate its ability to find the optimal solution over the full logic and geometric path. Real-robot experiments further showcase the effectiveness of our approach in handling contact uncertainty and external disturbances in the real world.**

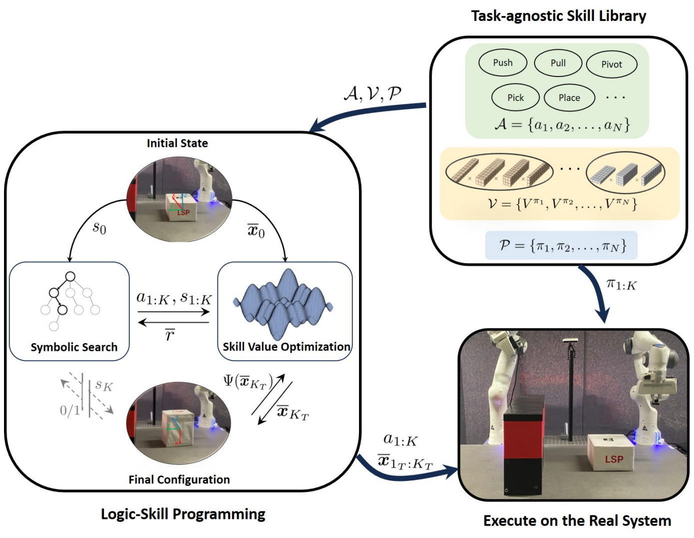

Task-agnostic Skill Library
Push
Pull
Pivot
Pick
Place
$\cdots$
$\mathcal{A} = \{a_1, a_2, \dots, a_N\}$
$\cdots$
$\mathcal{V} = \{V^{\pi_1}, V^{\pi_2}, \dots, V^{\pi_N}\}$
$\mathcal{P} = \{\pi_1, \pi_2, \dots, \pi_N\}$
$\mathcal{A}, \mathcal{V}, \mathcal{P}$
Initial State
LSP
s_0
$\overline{\boldsymbol{x}}_0$
Symbolic Search
$a_{1:K}, s_{1:K}$
$\overline{r}$
Skill Value Optimization
s_K
0/1
$\Psi(\overline{\boldsymbol{x}}_{K_T})$
$\overline{\boldsymbol{x}}_{K_T}$
Final Configuration
Logic-Skill Programming
$\pi_{1:K}$
$a_{1:K}$
$\overline{\boldsymbol{x}}_{1_T:K_T}$
LSP
Execute on the Real System

Fig. 1: Overview of the proposed approach: Given the evaluation function Ψ of the final configuration, along with the initial symbolic state s_0 and geometric state $\overline{\boldsymbol{x}}_0$, the objective of LSP is to find a solution that can accomplish the task with minimal control costs. A task-agnostic skill library is pretrained, consisting of N skill operators $\mathcal{A} = \{a_{1:N}\}$, along with corresponding value functions $\mathcal{V} = \{V^{\pi_{1:N}}\}$ and policies $\mathcal{P} = \{\pi_{1:N}\}$ in Tensor Train format. LSP finds the logic-geometric path by alternating between symbolic search and skill value optimization. Symbols $s_{1:K}$ are used as constraints for skill optimization, while skill optimization is used to check skeleton feasibility and final configuration performance, with a feedback reward $\overline{r}$ informing the symbolic search. This results in the appropriate skill skeleton $a_{1:K}$ and subgoal sequence $\overline{\boldsymbol{x}}_{1_T:K_T}$, which are then combined with the skill policies $\pi_{1:K}$ to actuate the real robot. Notably, the gray channel with symbolic final state s_K is interrupted because our framework eliminates the need for a symbolic target goal s_T, while such information is typically required in existing sampling-based sequential skill planning methods.

I. Introduction

Consider the following task: ”A large box is positioned on the table, next to a wall. The objective is to reorient the box to a new 6D pose using a single robot manipulator with minimal control efforts (or maximal rewards). The robot is allowed to have any interactions with the surroundings.” A potential solution involves pushing the box until it reaches the wall, pivoting, and then pulling it toward the target.

Such tasks are common in sequential manipulation scenarios, typically involving the sequencing of multiple contact-rich manipulation skills, such as pushing, pivoting, and pulling. Solving these tasks requires reasoning about the appropriate order of skills and the corresponding motion trajectories. This hybrid structure leads to Mixed-Integer Nonlinear Programming (MINLP), which combines the challenge of handling nonlinearities with the combinatorial complexity introduced by discrete contact modes. Introducing abstractions is promising to mitigate the computational burden of MINLP. One approach is to use convex relaxation as a geometric abstraction of the non-convex problem, resulting in Mixed-Integer Convex Programming (MICP) [12], which can be efficiently solved using modern solvers. Such methods have been applied to planar pushing ([11, 10]), pivoting [1], and locomotion [2] tasks. Recently, a method called Graph of Convex Sets (GCS) [12] was proposed, which formulates the planning problem as a shortest-path problem over graph, using a tight relaxation to approximate the original problem. [8] further demonstrates its effectiveness for planar pushing task. Although MICP methods have shown great performance in both manipulation and locomotion tasks, finding the convex relaxation of optimal control problems involving contact is not easy. Another approach is to use the Planning Domain Definition Language (PDDL) [3] as a symbolic abstraction of skill operators and geometric constraints [21, 7], where first-order logic facilitates high-level task planning and constrains low-level motion planning. However, the resulting constrained mathematical program is generally non-convex, and feasible solutions typically lie on low-dimensional contact manifolds, making both gradient-based and sampling-based methods struggle.

Therefore, having an effective abstraction that is more informative than symbols (or languages) and can also facilitate motion-level planning and control is essential. This aligns with the objective of value functions in dynamic programming [5]. Consequently, our paper aims to find an effective abstraction of the value function for each contact-rich skill. Reinforcement Learning (RL) is the most popular approach for this, but RL is typically time-consuming, and the learned value functions are not informative enough for sequencing in long-horizon manipulation planning. Recently, [6, 19] proposed a novel way to approximate arbitrary functions in low-rank arrays, which helps in efficiently finding global optimal solutions. The basic idea is to find the low-rank embedding of the original function, specifically in Tensor Train (TT) format, through cross-approximation [13, 17]. In this format, the marginal distribution of each dimension can be efficiently computed, enabling fast iterative sampling to find the solution. [20] further proposed using TT for approximate dynamic programming and demonstrated impressive results in the hybrid control of planar pushing tasks. In this work, we aim to demonstrate the effectiveness of this approach in learning value functions for contact-rich manipulation skills.

Using tensor approximation, we can construct a library with symbolic skill operators and their corresponding value functions in TT format, as shown in the Task-agnostic Skill library in Fig. 1. These value functions implicitly represent control policies, where control commands can be obtained through an $\arg\max$ operation. This differs from other learning-based sequential planning works [22, 4], which rely on explicit RL policies to output control commands directly. Logic-Skill Programming (LSP) [23] is further used to sequence task-agnostic skills from the library. Fig. 1 provides an overview of our proposed approach. Given the initial state s_0 and $\overline{\boldsymbol{x}}_0$, along with the evaluation function Ψ for the final geometric configuration, it generates the full logic-geometric path.

II. Methodology

A. Problem Formulation

Our objective is to address long-horizon manipulation tasks by sequentially executing a series of skills included in the library. To ensure that the skills can be sequenced in an arbitrary manner and generalized to any long-horizon tasks, the learning process should be task-agnostic. Each skill domain can be modeled by a Markov Decision Process (MDP)

$$\mathcal{M}_n = (\mathcal{X}_n, \mathcal{U}_n, \mathcal{T}_n, \mathcal{R}_n), \tag{1}$$

where $\mathcal{X}_n$ is the state space, $\mathcal{U}_n$ is the action space, $\mathcal{T}_n(x'_n|x_n, u_n)$ is the transition model, $\mathcal{R}_n(x_n, u_n)$ is the reward function given current state and action.

The full K-length long-horizon domain is the union of $\{\mathcal{M}_1, \mathcal{M}_2, \dots, \mathcal{M}_K\}$, where each time segment corresponds to a single skill execution. It is specified as

$$\overline{\mathcal{M}} = (\mathcal{M}_{1:K}, \overline{\mathcal{X}}, \Phi_{1:K}, \Gamma_{1:K}), \tag{2}$$

where $\overline{\mathcal{X}}$ is the full state space of the long-horizon domain, $\Phi_{1:K} : \mathcal{X}_k \to \overline{\mathcal{X}}$ is a function that maps the skill-specific state space to the full state space, and $\Gamma_{1:K} : \overline{\mathcal{X}} \to \mathcal{X}_k$ is a function that extracts the skill-specific state from the full state. Note that the state spaces of different skills usually have different dimensions and structures. Taking `push` and `pivot` as examples, they are planar manipulation primitives defined in the horizontal and vertical planes, respectively, affecting different degrees of freedom of the object. The long-horizon state space acts as a bridge, transmitting the local state changes from one skill to the next.

After acquiring these task-agnostic value functions, our goal is to determine the optimal skill skeleton and subgoal sequence that maximize the overall cumulative reward for the complete skill sequence $a_{1:K}$ and the evaluation of the final configuration $\Psi(\overline{\boldsymbol{x}}_{K_T})$:

$$\max_{\boldsymbol{x}, a_{1:K}} \sum_{k=1}^{K} \mathbb{E}_{\pi_k} \left[\sum_{t=0}^{\infty} \gamma^t R_{k_t} \right] + \Psi(\overline{\boldsymbol{x}}_{K_T}). \tag{3}$$

Here, R_{k_t} denotes the reward of skill a_k at time t, which is approximated by the value function V^{π_k}. Therefore, sequential skill planning problem can be formulated as maximizing the sum of value functions, along with the evaluation function Ψ of the final configuration, incorporating first-order logic as constraints:

$$\begin{aligned} \max_{\overline{\boldsymbol{x}}, a_{1:K}, s_{1:K}} \quad & \sum_{k=1}^{K} V^{\pi_k}(\boldsymbol{x}_{k_0}) + \Psi(\overline{\boldsymbol{x}}_{K_T}) \\ \text{s.t.} \quad & s_k \in \mathit{succ}(s_{k-1}, a_k), \overline{\boldsymbol{x}}_{1_0} = \overline{\boldsymbol{x}}_0, \\ & h_{\text{path}}(\boldsymbol{x}_{k_t}, \pi_k(\boldsymbol{x}_{k_t})|s_k) = 0, \\ & g_{\text{path}}(\boldsymbol{x}_{k_t}, \pi_k(\boldsymbol{x}_{k_t})|s_k) \leq 0, \\ & h_{\text{switch}}(\overline{\boldsymbol{x}}_{k_T}|a_{k+1}, s_k) = 0, \\ & g_{\text{switch}}(\overline{\boldsymbol{x}}_{k_T}|a_{k+1}, s_k) \leq 0, \\ & \overline{\boldsymbol{x}}_{k_t} = \Phi_k(\boldsymbol{x}_{k_t}|a_k), \\ & \boldsymbol{x}_{k_0} = \Gamma_k(\overline{\boldsymbol{x}}_{k_0}, \overline{\boldsymbol{x}}_{k_T}|a_k), \end{aligned} \tag{4}$$

where $\overline{\boldsymbol{x}}_{k_0}$ and $\overline{\boldsymbol{x}}_{k_T}$ are the initial and final configuration of skill operator a_k in the long-horizon domain $\overline{\mathcal{X}}$. $\boldsymbol{x}_{k_0}$ is the initial state of a_k within the skill domain $\mathcal{X}_k$. Φ_k and Γ_k are the mapping functions between long-horizon domain $\overline{\mathcal{X}}$ and skill domain $\mathcal{X}_k$. s_0 and $\overline{\boldsymbol{x}}_0$ denote the initial symbolic state and geometric configuration, respectively. h_{switch} and g_{switch} express the transition consistency of configuration $\boldsymbol{x}_k$ with the skill operator a_k. h_{path} and g_{path} indicate the constraints on the path $\boldsymbol{x}_{k_t}$ given current symbolic state s_k. These constraints specify the degrees of freedom in the system configurations that can be actuated under the symbolic state s_k. For instance, an object marked as `non-graspable` and `onTable` can only be manipulated through `push` or `pull`, with actuation in the dimensions of `x`, `y` and `yall`. Conversely, if the object is marked as `atWall`, it can be actuated by `pivot`, with changes in `roll` or `pitch`.

We assume the initial geometric configuration $\overline{\boldsymbol{x}}_0$ is given, along with an initial symbolic state $s_0 \in L$. The symbolic states can be transited through the skill operator a_k as $s_k = \mathit{succ}(a_k, s_{k-1})$. The existing skill sequencing frameworks typically require an explicit symbolic goal target $s_T \models \mathcal{G}$, while our method eliminates this requirement by considering only the final geometric configuration $\boldsymbol{x}_{K_T}$.

To solve Eq. (4), we propose an approach that alternates between searching for symbolic skill skeleton and optimizing the value functions. The role of symbolic search is to define the optimization constraints, while skill value optimization checks whether the given skill skeleton can reach the final target.

*1) **Symbolic Search**:* We use PDDL to describe the task domain and then utilize Monte Carlo Tree Search (MCTS) to search for the appropriate skill sequence $a_{1:K}$ from the skill library. The preconditions and effects of each skill form a rule-based representation of symbolic transitions $s_k = succ(a_k, s_{k-1})$, serving as the forward model for symbolic search. In each iteration, MCTS relies on the Upper Confidence Bound (UCB) to select the node, balancing exploitation and exploration. If no child node is found in this branch, the branch will expand and simulate until reaching the target or exceeding the maximum length. A reward or penalty will be backpropagated through the branch, depending on the simulation result. This iterative process refines the search tree, focusing on promising branches and ultimately converging towards an optimal decision. The sequence length (K in Eq. (4)) is purely unknown before symbolic search, allowing diverse sequence lengths for the same target configuration. By applying a higher exploration parameter in the UCB formula, MCTS can return multiple solutions.

*2) **Skill Value Optimization**:* The symbolic skill skeleton can only express the symbolic feasibility. It has to be verified by optimizing over Eq. (4) to determine whether it can reach the final configuration while satisfying the constraints. The value functions in Eq. (4) are obtained through Generalized Policy Iteration using Tensor Train (TTPI) [20].

After obtaining the library of optimal value functions, we can solve Eq. (4) conditioning on the skill skeleton returned from symbolic search. Note that both discrete and continuous variables may be involved in this problem. For instance, in the task mentioned at the beginning, we have to rely on pivoting against the wall to change the `roll` or `pitch` angle of the box, requiring one face of the box to be parallel to the wall. Moreover, the objective functions are arbitrary, depending on the value functions of selected skills. All of these factors pose significant challenges for optimization techniques. In this work, we employ the Cross-Entropy Method (CEM) [16] with mixed distribution, namely CEM-MD, as the optimization technique. It can handle mixed-integer programming by using Gaussian distribution and Categorical distribution for continuous and discrete variables, respectively. The distributions are iteratively updated towards the fraction of the population with higher objective scores until converging to the best solution.

III. Experiments

In this section, we compare our method with RL for value function learning and demonstrate its effectiveness in both simulation and real-world environments. Demos can be found at https://sites.google.com/view/lsp4plan [1].

A. Evaluation on Value Function Abstractions

Five manipulation skills, pushing, pivoting, pulling, picking, and placing, are used to evaluate the performance of the learned value functions through TTPI and two RL methods: Soft Actor-Critic (SAC) [9] and Proximal Policy Optimization (PPO) [18]. Each skill is characterized by unique state and action spaces.

1) **Push**: The state is characterized by $(\boldsymbol{p}_o, \theta_o, \boldsymbol{p}_r, f_c)$, while the action is denoted by $(\boldsymbol{v}_r, f_n)$. Here, $(\boldsymbol{p}_o, \theta_o) \in$ SE(2) denotes the object's pose in the world frame. $\boldsymbol{p}_r$ and $\boldsymbol{v}_r$ represent the position and velocity of the robot end-effector in the object frame. We assume that the object has a rectangular shape (common in industry), where $f_c \in 0, 1, 2, 3$ represents the current contact surface and f_n denotes the next contact surface. Thus, the system encompasses a total of 6 states and 3 control variables, comprising both continuous and discrete variables.
2) **Pivot**: The pivoting domain features a goal-augmented state $(\beta, \tilde{\beta})$, where β is the current rotation angle of the object in the gravity plane, and $\tilde{\beta}$ is the desired angle. The control input is $\dot{\beta}$, which denotes the angular velocity of the robot end-effector..
3) **Pull**: The state is defined as $(\boldsymbol{p}_o, \theta_o) \in SE(2)$, representing the object's position and orientation in a planar plane. The control input is $(\dot{\boldsymbol{p}}_o, \dot{\theta}_o)$, denoting the translational and angular velocities of the robot end-effector.
4) **Pick**: In this domain, the state is defined as $(x, y, z, \alpha, \beta, \theta) \in SE(3)$, representing the pose of the robot end-effector, while the control input is the Cartesian velocity of the end-effector, $(\dot{x}, \dot{y}, \dot{z}, \dot{\alpha}, \dot{\beta}, \dot{\theta}) \in SE(3)$. Without loss of generality, we assume that the object to be picked is located at $(0, 0, 0, 0, 0, 0)$.
5) **Place**: This domain shares the same state and action spaces as the **Pick** domain.

We define the general reward for skill learning as:

$$r = -1 \times (c_p + \rho \times c_o + 0.01 \times c_a + 0.1 \times c_f), \quad (5)$$

with

$$\begin{aligned} c_p &= \|\boldsymbol{x_p} - \boldsymbol{x_p}^{\text{des}}\|/l_p, \quad c_o = \|\boldsymbol{x_o} - \boldsymbol{x_o}^{\text{des}}\|/l_o, \\ c_a &= \|\boldsymbol{u}\|, \quad c_f = 1 - \delta(f_c - f_n), \end{aligned} \quad (6)$$

where $\boldsymbol{x}_p$ and $\boldsymbol{x}_o$ represent the current position and orientation of the system in each skill domain, while $\boldsymbol{x}_p^{\text{des}}$ and $\boldsymbol{x}_o^{\text{des}}$ denote the target position and orientation. We set the state space to be within $[-0.5m, 0.5m]$ for position elements, and $[-\pi, \pi]$ for orientation elements. l_p and l_o are therefore set to 0.5 and π respectively to normalize the position error and orientation error. ρ is a hyperparameter used to balance position and orientation errors. Due to the underactuated nature of pushing dynamics, we set $\rho = 0.5$ in practice. For other skills, ρ is set to 1. $\boldsymbol{u}$ represents the control inputs of each skill domain.

[1]This website is for our original paper, from which this workshop abstract is derived.

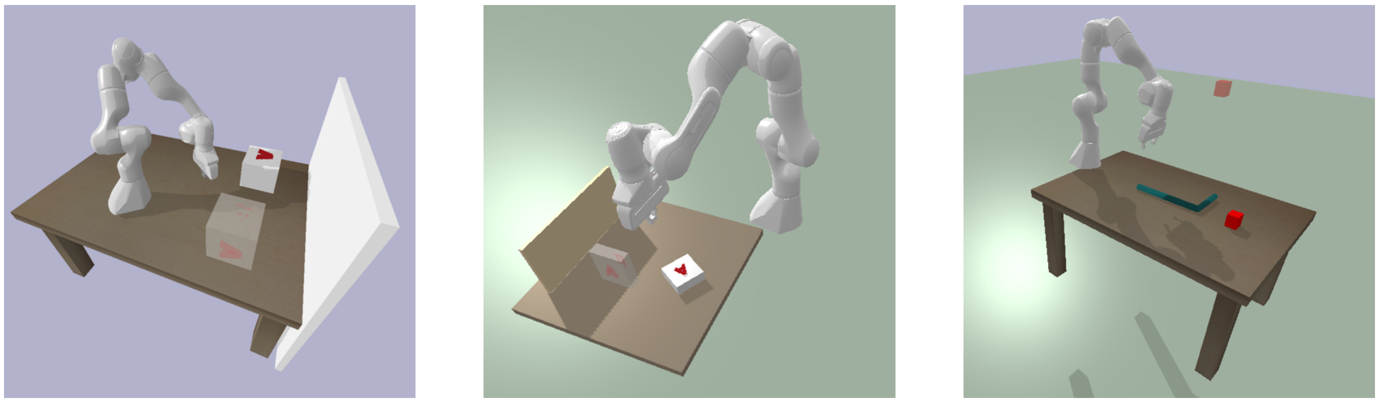

(a) Non-Prehensile Manipulation domain **(b)** Partly-Prehensile Manipulation domain **(c)** Prehensile Manipulation domain

Fig. 2: Three sequential manipulation domains, including both prehensile and non-prehensile manipulation primitives. The transparent object represents the final target configuration in each domain.

c_f is a specialized term unique to the pushing domain, used to penalize face switching during pushing. Here, $\delta(f_c - f_n)$ returns 1 when $f_c = f_n$ (indicating no face switching), otherwise, it returns 0.

Notably, considering that SAC and PPO are not able to handle hybrid action space, we exclude f_c and f_n for a fair comparison in this section. For both PPO and SAC, our primary implementation relies on Stable-Baselines3 [15], utilizing a Multilayer Perceptron (MLP) architecture with dimensions of 32×32 as the policy network. We set the discount factor to 0.99 and the learning rate to 0.001, while configuring the task horizon to 10^4. In TTPI, we establish the accuracy threshold for TT-cross as $\epsilon = 10^{-3}$ and set the maximum rank $r_{\max}$ to 10^2.

The obtained policies are then evaluated by comparing the success rates across 1000 initial states in skill domains. We define a successful task as achieving a position error of less than 0.03cm and an orientation error of less than 15°. Table I reveals that all the policies can nearly reach the final targets, demonstrating the proficiency of the learned policies in accomplishing individual manipulation tasks. Moreover, we evaluate the accuracy of obtained value functions by checking whether they can offer the same guidance as the cumulative rewards given two states, shown as *value prediction* in Table I. The metric is to compare whether $V^{\pi_k}(\boldsymbol{x}_1) - V^{\pi_k}(\boldsymbol{x}_2)$ has the same direction as $\mathcal{R}_c^{\pi_k}(\boldsymbol{x}_1) - \mathcal{R}_c^{\pi_k}(\boldsymbol{x}_2)$, where $\mathcal{R}_c^{\pi_k}(\boldsymbol{x})$ is the cumulative reward, defined as

$$\mathcal{R}_c^{\pi_k}(\boldsymbol{x}) = \sum_{t=0}^{\infty} \gamma^t R_{k_t}(\boldsymbol{x}_{k_t}, \pi_k(\boldsymbol{x}_{k_t})), \text{ s.t. } \quad \boldsymbol{x}_{k_0} = \boldsymbol{x}. \tag{7}$$

We randomly selected 1000 state pairs in the state domain. The value function of TTPI demonstrates superior prediction performance compared to SAC and PPO. This indicates that value functions in TT format offer better accuracy in approximating the cumulative reward, which can inform which state in the state domain is more dynamically optimal given current policy. This observation aligns with our motivation, emphasizing that typical RL methods approximate the value function primarily within a local space and the policy retrieval is often sub-optimal due to gradient-based optimization. In contrast, TTPI, aims to approximate the value function across the entire state space. This feature is advantageous in our sequential skill planning framework for identifying the subgoal for each skill, which can be anywhere within the skill-specific state space.

B. Simulation and real-world experiments

Three different long-horizon domains are used to validate our method, involving both non-prehensile and prehensile manipulation primitives, as shown in Fig. 2.

a) Non-Prehensile Manipulation (NPM): As depicted in Fig. 2a, this domain involves objects that cannot be grasped. The objective is to manipulate the box within the 3D world by leveraging multiple non-prehensile planar manipulation primitives and establishing contacts with the surroundings to achieve the final 6D pose. This task can also be seen as a special case of in-hand manipulation, where the robot and the wall act as active and passive "fingers", respectively.

b) Partly-Prehensile Manipulation (PPM): As shown in Fig. 2b, this domain involves objects that can only be grasped in specific directions. The goal in this domain is to manipulate the 6D pose of the cube, including the z direction. Therefore, the robot must strategically decide how to pick up the object in the end. This domain requires the robot to reason about physical contact and object geometry to unify both prehensile and non-prehensile primitives.

c) Prehensile Manipulation (PM): As illustrated in Fig. 2c, this domain involves objects that can be directly grasped. The goal is to alter the 6D pose of a block placed on the table, beyond the robot's reachability. To achieve this, the robot must pick up the block in the end. This requires the robot to figure out using a hook to extend the kinematic chain and pull the block back into the reachability region, and then grasp it. However, the way in which the robot picks up the hook will affect whether the block can be reached, and where to pull will also affect the entire trajectory and the total energy cost.

Table I: Comparative Analysis of Skill Policy Performance and Value Function Precision

	TTPI			SAC			PPO		
	success rate	value prediction	time (min)	success rate	value prediction	time (min)	success rate	value prediction	time (min)
pushing	1.0	0.85	5.6	0.83	0.63	36.3	0.95	0.61	44.8
pivoting	1.0	0.94	0.9	1.0	0.51	16.0	1.0	0.68	8.7
pulling	1.0	0.97	1.1	1.0	0.84	16.5	1.0	0.72	10.7
pick/place	1.0	0.93	1.67	0.98	0.64	33.6	1.0	0.66	24.6

In this task, we want to show that our method can find an optimal trajectory which has the minimum energy cost, while successfully finishing the task.

We define the evaluation function Ψ of the final configuration $\overline{\boldsymbol{x}}_{K_T}$ as follows:

$$\Psi(\overline{\boldsymbol{x}}_{K_T}) = \lambda \|\overline{\boldsymbol{x}}_{K_T} - \overline{\boldsymbol{x}}_T\|, \tag{8}$$

where $\overline{\boldsymbol{x}}_T$ is the target configuration in each domain, and $\lambda = 10^2$. The initial configuration $\overline{\boldsymbol{x}}_0$ and target configuration $\overline{\boldsymbol{x}}_T$ are randomly sampled from the long-horizon domain $\overline{\mathcal{X}}$, which is the configuration space of the entire environment. In NPM and PPM, $\overline{\mathcal{X}} = \mathcal{S}_R \times \mathcal{S}_O$, while in PM, $\overline{\mathcal{X}} = \mathcal{S}_R \times \mathcal{S}_O \times \mathcal{S}_T$. Here, $\mathcal{S}_O = SE(3)$ and $\mathcal{S}_R = SE(3)$ denote the pose of the object and robot end-effector, respectively, while $\mathcal{S}_T = SE(2)$ represents the pose of a tool positioned on the table.

We then run LSP for these three domains to find the full logic-geometric path. Given the same initial state s_0 and $\overline{\boldsymbol{x}}_0$ in each domain, we randomly sample 10 different target configurations $\overline{\boldsymbol{x}}_T$ that require multi-step manipulation. Fig. 3 shows that LSP can actively find multiple solutions. For the Non-Prehensile domain, two solutions found by LSP are `push-pivot-pull` and `pull-pivot-pull`. For the Partly-Prehensile domain, four different skeletons are found: `push-pick`, `pull-pick`, `push-pivot-pull-pick` and `pull-pivot-pull-pick`. For the Prehensile domain, the solved skill skeleton is `pick-place-pull-place-pick`.

We further conducted real-robot experiments in the NPM domain, employing a 7-axis Franka Emika robot and a RealSense D435 camera. A large box (21cm x 21cm x 16cm) was positioned on a flat plywood surface. Fig. 4 displays the keyframes of the robot experiments. Given the initial and final configurations, the robot adeptly manipulated the box by utilizing three planar manipulation primitives, establishing and breaking contact with the surroundings. It is worth noting that real-world contact-rich manipulation is quite challenging due to friction uncertainty and external disturbances [14]. This highlights the importance of introducing effective geometric abstractions for online real-time control. Through skill sequencing, we demonstrate that the robot can actively engage with the physical world, accomplishing a much more complex long-horizon task.

IV. Conclusion and Future Work

In this work, we introduce the use of tensor factorization to approximate value functions, which serve as geometric abstractions of contact-rich skills to facilitate motion-level planning and control. A first-order extended mathematical program is used to find the skill skeleton and subgoal sequence that maximize the overall cumulative reward and optimize the performance of the final configuration.

We demonstrated that the value functions in TT format provide a better approximation of cumulative reward compared to state-of-the-art RL methods. Furthermore, the proposed LSP framework can generate multiple skill skeletons and their corresponding subgoal sequences, given only an evaluation function of the final geometric configuration. We validated this approach across three manipulation domains, highlighting its robust performance in sequencing both prehensile and non-prehensile manipulation primitives.

In this work, the skill learning method, TTPI, assumes a low-rank structure of the value functions for approximation. This works well for skill domains with low-to-medium dimensionality but struggles with image-based policy learning. To address this issue, combining neural networks with TT decomposition could be an interesting direction.

Moreover, the Cross-Entropy Method with Mixed Distribution (CEM-MD) is used for mixed-integer optimization. While it demonstrates good performance in the current domain settings, its efficiency may decrease when dealing with much longer skill skeletons and additional objects. Given that TT allows for efficient optimization, and that the mode-switching configurations typically lie on low-dimensional manifolds, a potential future direction would be to exploit the low-rank structure for full path optimization.

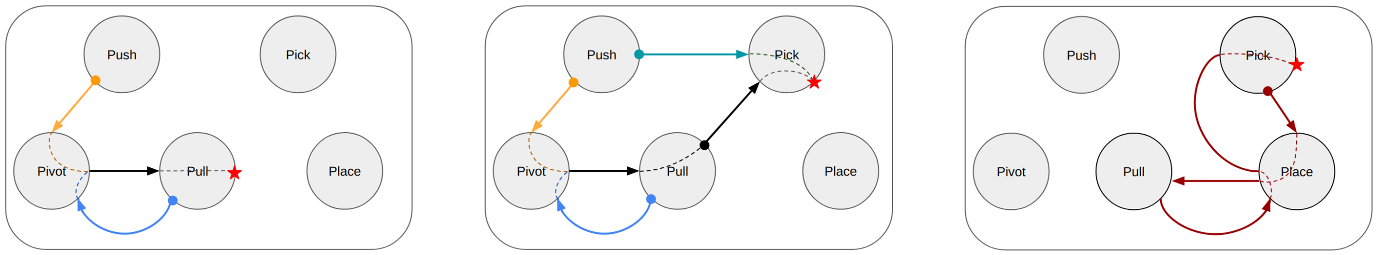
Furthermore, while Large Language Models (LLMs) have demonstrated impressive results in abstracting the geometric world as language, we believe our method can play an important role in enriching these language abstractions with more geometric information to facilitate low-level motion planning and control.

References

[1] Bernardo Aceituno-Cabezas and Alberto Rodriguez. A global quasi-dynamic model for contact-trajectory optimization in manipulation. 2020.

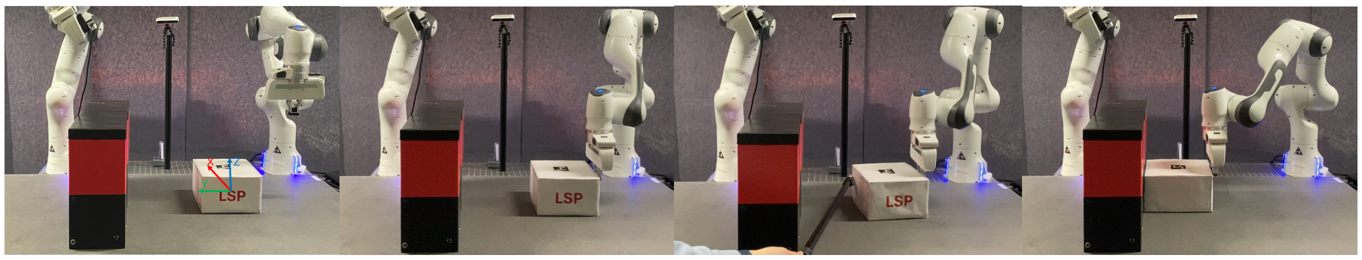
[2] Bernardo Aceituno-Cabezas, Carlos Mastalli, Hongkai Dai, Michele Focchi, Andreea Radulescu, Darwin G Caldwell, José Cappelletto, Juan C Grieco, Gerardo Fernández-López, and Claudio Semini. Simultaneous contact, gait, and motion planning for robust multilegged locomotion via mixed-integer convex optimization. *IEEE Robotics and Automation Letters*, 3(3):2531–2538, 2017.

[3] Constructions Aeronautiques, Adele Howe, Craig Knoblock, ISI Drew McDermott, Ashwin Ram, Manuela Veloso, Daniel Weld, David Wilkins SRI, Anthony

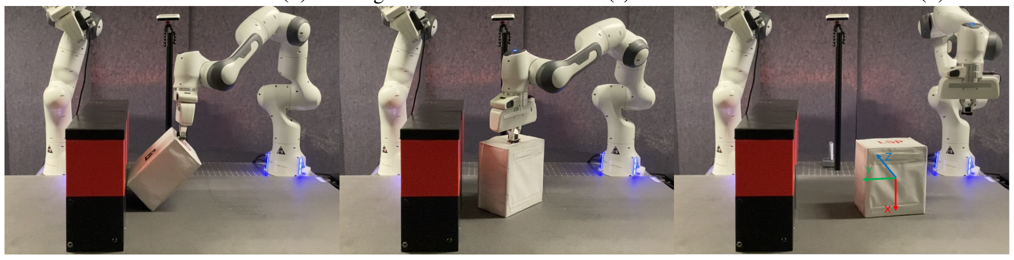


(a) Non-Prehensile Manipulation **(b)** Partly-Prehensile Manipulation **(c)** Prehensile Manipulation

Fig. 3: The action skeletons obtained by LSP for three domains. The dot point denotes the start of the skill sequence. Each color represents one solution, with black lines indicating the common shared tunnel. The red star illustrates the end of the skill skeleton.



(a) Initialization **(b)** Pushing **(c)** Disturbance **(d)** Push-Pivot Switch



(e) Pivoting **(f)** Pulling **(g)** Target Reaching

Fig. 4: Non-prehensile manipulation domain task. The system is initialized as (a), and the objective is to manipulate the box to achieve the target configuration as (g). The first stage involves pushing the box towards the wall with a 90° rotation. Additionally, we apply an external disturbance to test the skill policy (c). After the pushing stage, the robot switches to the pivoting skill (d, e), followed by pulling (f), until reaching the final geometric configuration.

Barrett, Dave Christianson, et al. Pddl— the planning domain definition language. *Technical Report, Tech. Rep.*, 1998.

[4] Christopher Agia, Toki Migimatsu, Jiajun Wu, and Jeannette Bohg. Stap: Sequencing task-agnostic policies. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 7951–7958. IEEE, 2023.

[5] Richard Bellman. Dynamic programming. *Science*, 153 (3731):34–37, 1966.

[6] Andrei Chertkov, Gleb Ryzhakov, Georgii Novikov, and Ivan Oseledets. Optimization of functions given in the tensor train format. *arXiv preprint arXiv:2209.14808*, 2022.

[7] Caelan Reed Garrett, Tomás Lozano-Pérez, and Leslie Pack Kaelbling. Pddlstream: Integrating symbolic planners and blackbox samplers via optimistic adaptive planning. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 30, pages 440–448, 2020.

[8] Bernhard P Graesdal, Shao YC Chia, Tobia Marcucci, Savva Morozov, Alexandre Amice, Pablo A Parrilo, and Russ Tedrake. Towards Tight Convex Relaxations for Contact-Rich Manipulation. 2024.

[9] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pages 1861–1870. PMLR, 2018.

[10] Francois R Hogan and Alberto Rodriguez. Reactive planar non-prehensile manipulation with hybrid model predictive control. *International Journal of Robotics Research (IJRR)*, 39(7):755–773, 2020.

[11] François Robert Hogan and Alberto Rodriguez. Feedback control of the pusher-slider system: A story of hybrid and underactuated contact dynamics. In *Algorithmic Foundations of Robotics XII: Proceedings of the Twelfth Workshop on the Algorithmic Foundations of Robotics*, pages 800–815. Springer, 2020.

[12] Tobia Marcucci and Russ Tedrake. Mixed-integer formulations for optimal control of piecewise-affine systems. In *Proceedings of the 22nd ACM International Conference on Hybrid Systems: Computation and Control*, pages

230–239, 2019.

[13] Ivan Oseledets and Eugene Tyrtyshnikov. TT-cross approximation for multidimensional arrays. *Linear Algebra and its Applications*, 432(1):70–88, 2010.

[14] Tao Pang, HJ Terry Suh, Lujie Yang, and Russ Tedrake. Global planning for contact-rich manipulation via local smoothing of quasi-dynamic contact models. *IEEE Transactions on Robotics*, 2023.

[15] Antonin Raffin, Ashley Hill, Adam Gleave, Anssi Kanervisto, Maximilian Ernestus, and Noah Dormann. Stable-Baselines3: Reliable Reinforcement Learning Implementations. *Journal of Machine Learning Research*, 22(268):1–8, 2021. URL http://jmlr.org/papers/v22/20-1364.html.

[16] Reuven Rubinstein. The cross-entropy method for combinatorial and continuous optimization. *Methodology and computing in applied probability*, 1:127–190, 1999.

[17] Dmitry V. Savostyanov and Ivan V. Oseledets. Fast adaptive interpolation of multi-dimensional arrays in tensor train format. *The 2011 International Workshop on Multidimensional (nD) Systems*, pages 1–8, 2011.

[18] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[19] S. Shetty, T. Lembono, T. Löw, and S. Calinon. Tensor Train for Global Optimization Problems in Robotics. *International Journal of Robotics Research (IJRR)*, 43 (6):811–839, 2024. doi: 10.1177/02783649231217527.

[20] S. Shetty, T. Xue, and S. Calinon. Generalized Policy Iteration using Tensor Approximation for Hybrid Control. In *Proc. Intl Conf. on Learning Representations (ICLR)*, 2024.

[21] Marc Toussaint. Logic-geometric programming: an optimization-based approach to combined task and motion planning. In *Proceedings of the 24th International Conference on Artificial Intelligence*, pages 1930–1936, 2015.

[22] Danfei Xu, Ajay Mandlekar, Roberto Martín-Martín, Yuke Zhu, Silvio Savarese, and Li Fei-Fei. Deep affordance foresight: Planning through what can be done in the future. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6206–6213. IEEE, 2021.

[23] T. Xue, A. Razmjoo, S. Shetty, and S. Calinon. Logic-Skill Programming: An Optimization-based Approach to Sequential Skill Planning. In *Proc. Robotics: Science and Systems (R:SS)*, 2024.