Learning Plan-Satisficing Motion Policies from Demonstrations

Yanwei Wang*, Nadia Figueroa†, Shen Li*, Ankit Shah‡, Julie Shah*
*MIT CSAIL, †University of Pennsylvania, ‡Brown University

***Abstract*—Learning from demonstration (LfD) methods have shown promise for solving multi-step tasks; however, these approaches do not guarantee successful reproduction of the task given perturbations. In this work, we identify the roots of such a challenge as the failure of the learned continuous policy to satisfy the discrete plan implicit in the demonstration. By utilizing modes (rather than subgoals) as the discrete abstraction and motion policies with both mode invariance and goal reachability properties, we show our learned continuous policy can simulate any given discrete plan. Consequently, the imitator is robust to both task- and motion-level perturbations and guaranteed to achieve task success. Project page: https://yanweiw.github.io/tli/**

I. Introduction

In prior work, learning from demonstration (LfD) [1] has successfully enabled robots to accomplish multi-step tasks by segmenting demonstrations into subtasks/subgoals [2], phases [3], keyframes [4], and primitives [5]. Most of these abstractions assume reaching subgoals sequentially will deliver the desired outcomes. However, successful imitation of many manipulation tasks with spatial/temporal constraints cannot be reduced to imitation at the motion level unless the learned motion policy also satisfies these constraints. For example, transferring a spoonful of soup without restricting the orientation of the spoon can fail due to spilling even when the spoon reaches the target successfully.

We show that successful goal-reaching does not imply successful task execution in Fig. 1 with a 2D task, where task success depends on whether a continuous trajectory simulates a discrete plan of transitioning through the white, yellow, pink and green regions consecutively. Human demonstrations, shown in Fig. 1**(a)**, are employed to learn a dynamical system (DS) policy [6] depicted by the streamlines in Fig. 1**(b)**. Of all sampled trajectories, only the blue ones succeed in the task. The red ones fail as they simulate at least one discrete transition not physically realizable (e.g., white ⇒ pink). The issue is not mitigated by further segmenting the demonstrations into three subgoals and learning a DS for each subgoal, as seen in Fig. 1 **(c-f)**. While one can frame this problem as covariate shift and solve it by asking humans for more demonstrations [7], we frame it as the mismatch between a learned continuous policy and a discrete task plan and solve it by asking humans for a task specification.

Specifically, the core challenges illustrated by this example are two-fold: 1) Subgoals only impose point constraints that are insufficient to represent the boundary of a discrete abstraction. 2) The continuous policy may deviate from a demonstrated discrete plan and in such cases cannot replan to ensure all discrete transitions are valid. Instead, our approach employs "modes" as the discrete abstraction. We define a *mode* as a set of robot and environment configurations that share the same sensor reading [8]. For example, in Fig. 1 each colored region is a unique mode and has a boundary that imposes path constraints on motion policies. Additionally, we use a task automaton as a receding horizon controller that replans when a perturbation causes the system to travel outside a mode boundary. For example, detecting a transition of yellow ⇒ white instead of the desired yellow ⇒ pink will result in a new plan: white ⇒ yellow ⇒ pink ⇒ green.

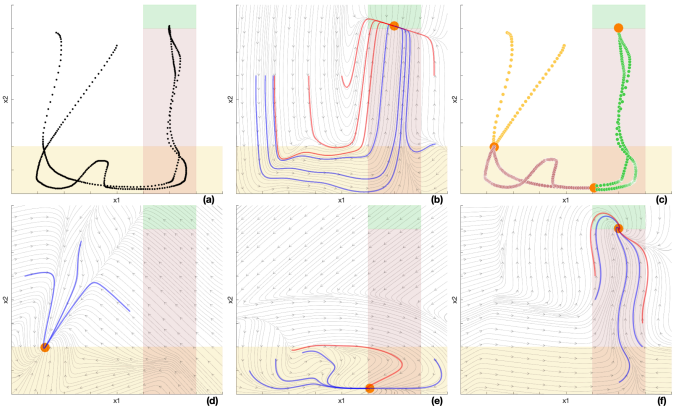

Fig. 1. A mode abstraction of a soup-scooping task. x_1 denotes spoon orientation and x_2 represents spoon distance from the soup. **(a)** Task: White region (all spoon configurations without soup on it) ⇒ yellow region (spoon in contact with soup) ⇒ pink region (spoon holding soup) ⇒ green region (soup at target). Note transitions (white ⇒ pink) and (white ⇒ green) are not physically realizable. The black curves denote two successful demonstrations. **(b)** Learning a dynamical system (DS) policy [6] over unsegmented data can result in successful task replay (blue trajectories), but lacks a guarantee due to invalid transitions (red trajectories). **(c)** Trajectories segmented into three colored regions (modes), with orange circles denoting attractors. **(d-f)** Learning individual DS over segmented trajectories still results in *invariance failures* (i.e., traveling outside of modes) for some starting configurations (red trajectories) as we perturb the spoon during task replay.

In this work, we assume the task automaton is given in the form of a Linear Temporal Logic formula. We denote the challenge of learning a continuous policy that can realize any discrete plan of valid mode transitions specified by the automaton as *temporal logic imitation* (TLI). In contrast to temporal logic planning (TLP) [9], where the workspace is typically partitioned into connected convex cells with known boundaries, TLI does not assume knowing mode boundaries. Consequently, the learned policy might prematurely exit a mode if the robot is perturbed to out-of-distribution states. To guarantee any discrete plan is feasible during execution at the continuous level, we show a learned policy with a global stability property can be refined to satisfy the bisimulation criteria [9] through human perturbations. By studying TLP in the setting of LfD, we are able to address covariate shift through learning motion policies that always obey discrete

plans without additional online data collection.

II. Temporal Logic Imitation Formulation

A. Robot Model and Sensor Model

We use a first-order dynamical system $\dot{x} = f(x)$ to represent the desired evolution of a robot end-effector, where $x = [x_1, ..., x_n]^T \in \mathbb{R}^n$ describes an n-dimensional continuous robot state. Let discrete sensor state $\alpha = [\alpha_1, ..., \alpha_m]^T \in \{0, 1\}^m$ be an m-dimensional sensor variable. We convert all sensor signals into Booleans to identify different modes. We define a system state as a tuple $s = (x, \alpha) \in \mathbb{R}^n \times \{0, 1\}^m$ and its corresponding mode $\sigma \in \Sigma$ as $\sigma = \mathcal{L}(\alpha)$. Overloading the notation, we use σ to represents the set of all system states in the same mode, i.e., $\sigma_i = \{s = (x, \alpha) \mid \mathcal{L}(\alpha) = \sigma_i\}$. In contrast, $\delta_i = \{x | s = (x, \alpha) \in \sigma_i\}$ represent the corresponding set of robot states. In this work, we only consider task-space end-effector pose control of a manipulator.

B. Demonstrations and perturbations

Demonstrations are in the form $\{\{x^{t,k}, \dot{x}^{t,k}, \alpha^{t,k}\}_{t=1}^{T_k}\}_{k=1}^{K}$, where $x^{t,k}$, $\dot{x}^{t,k}$, $\alpha^{t,k}$ are robot state, velocity, and sensor state at time t in demonstration k. T_k is the length of each k-th trajectory. Given l number of unique sensor states, we can segment the K demonstrations into l clusters of modes. We learn a policy for each mode, which we stress test with either (1) motion-level perturbations that displace the continuous motion within the same mode, or (2) task-level perturbations that drive the system outside of the current mode.

C. Problem Statement

Given (1) a task automaton ϕ specifying valid mode transitions to achieve a task, and (2) successful demonstrations $\{\{x^{t,k}, \dot{x}^{t,k}, \alpha^{t,k}\}_{t=1}^{T_k}\}_{k=1}^{K}$, we want to imitate a continuous policy that can realize any discrete mode sequence planned by the automaton despite arbitrary perturbations.

III. Method

A. Bisimulation between Discrete Plan and Continuous Policy

To realize any discrete plan, every mode's associated continuous policy must satisfy the bisimulation conditions: [9].

Condition 1 (Invariance). *Every continuous motion starting in a mode must remain within the same mode while following the current mode's policy; i.e.,* $\forall i \ \forall t \ (s^0 \in \sigma_i \rightarrow s^t \in \sigma_i)$

Condition 2 (Reachability). *Every continuous motion starting in a mode must reach the next mode in the demonstration while following the current mode's policy; i.e.,* $\forall i \ \exists T \ (s^0 \in \sigma_i \rightarrow s^T \in \sigma_j)$

B. Ensuring Goal Reachability by DS

We associate each mode σ_i with a state-based policy $\dot{x} = f_i(x)$. While any BC variant with a stability guarantee can satisfy reachability, we focus on DS in this work. DS f_i has domain $\mathbb{R}^n$ and goal set $g_i = \{x^*_{\sigma_i}\}$ with $x^*_{\sigma_i}$ being the attractor. DS is globally asymptotically stable (G.A.S) and every $x \in \mathbb{R}^n$ is guaranteed to reach $x^*_{\sigma_i}$. By putting $x^*_{\sigma_i}$ in the boundary set of δ_j for a mode σ_j, we ensure mode σ_j is reachable from every s in mode σ_i. Next, we follow [6] to represent a non-linear DS as a weighted sum of linear systems: $\dot{x} = f(x) = \sum_{k=1}^{K} \theta_k(x)(A^k x + b^k)$ where $A^k \in \mathbb{R}^{n \times n}$ and $b^k \in \mathbb{R}^n$ are the k-th linear system parameters, and $\theta_k(x) : \mathbb{R}^n \rightarrow \mathbb{R}^+$ is the mixing function. To certify stability, we use Lyapunov function $V(x) = (x - x^*)^T P (x - x^*)$. The nonlinear DS is G.A.S at an attractor x^* if $\exists P = P^T$ and $P \succ 0$ such that:

$$\begin{cases} (A^k)^T P + P A^k = Q^k, Q^k = (Q^k)^T \prec 0 \\ b^k = -A^k x^* \end{cases} \quad \forall k \qquad (1)$$

Minimizing the fitting error with respect to demonstrations subject to constraints in Eq. 1 finds us a non-linear DS with a stability guarantee.

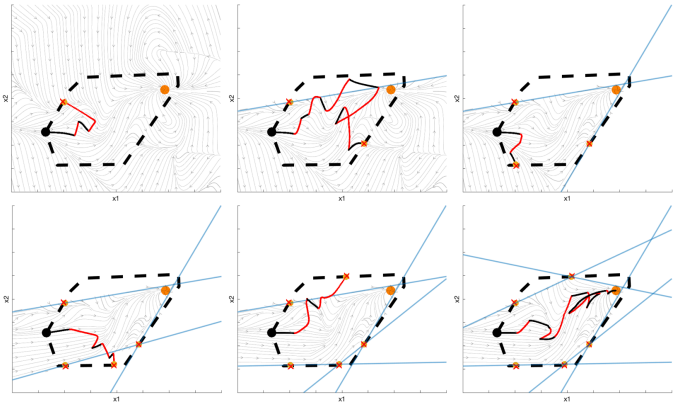

Fig. 2. Illustration of iterative estimation of mode boundary with cutting planes. A system enters a mode with unknown boundary (dashed line) at the black circle and is attracted to the goal at the orange circle. Its trajectory in black shows the original policy rollout, and its trajectory in red is driven by perturbation. When the system exits the mode, a cut is placed at the last in-mode state (yellow circle) to bound the mode from the failure state (red cross), and the system is reset to the entry state. When the system is inside the cuts, it experiences modulated DS that never leaves the mode (flows entering the mode are not modulated). When the system is outside the cuts but inside the mode, it follows the original DS. Notice only mode exits in black are invariance failures in need of modulation. Mode exits in red are driven by perturbation to illustrate that more cuts lead to better boundary approximation.

C. Ensuring Mode Invariance by Modulating DS

First, we estimate unknown mode boundaries by leveraging sparse events of mode exits detected by sensors. Specifically, for each invariance failure we construct a cut that separates the failure state x^{T_f} from the mode-entry state, the last in-mode state $x^{T_f - 1}$, and the mode attractor x^*. We ensure this separation constraint with a quadratically constrained quadratic program (QCQP) and search for the normal direction (pointing away from the mode) of a hyper-plane that goes through $x^{T_f - 1}$ such that the plane's distance to x^* is minimized. The intersection of half-spaces cut by hyper-planes inner approximates a convex mode boundary as seen in Fig. 2. Second, the cuts, which deflect DS flows as collision objects [10], together form an implicit function $\Gamma(x) : \mathbb{R}^n \rightarrow \mathbb{R}^+$ with $\Gamma(x) < 1, = 1, > 1$ denoting the current estimated

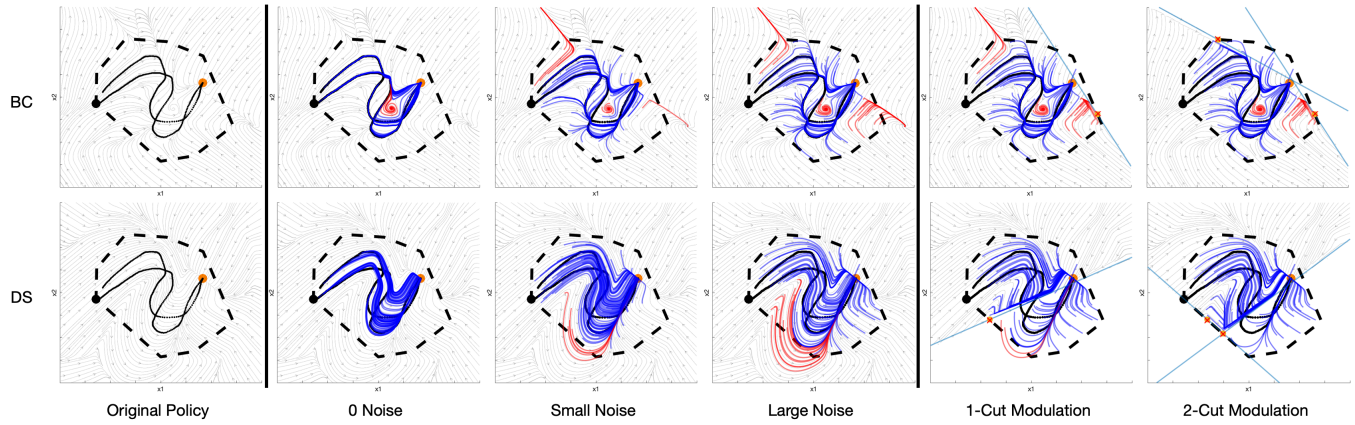

Fig. 3. Policy rollouts from different starting states for a randomly generated convex mode. The top row shows BC results and the bottom row shows DS results. The left column visualizes the original policies learned from 2 demonstrations (black trajectories). The orange circle indicates an attractor. The middle columns add different levels of Gaussian noise to the initial states sampled from the demonstration distribution. Blue trajectories successfully reach the attractor while red trajectories fail either due to invariance failures or reachability failures. Notice these failures only happen at places without data coverage. Right columns show cutting planes (blue lines) separate failures (red cross) from last visited in-mode states (yellow circles) and consequently can modulate policies to be mode invariant. Applying cutting planes on BC policies cannot correct reachability failures within the mode. More results are on the website.

interior, boundary and exterior of a mode. $\Gamma(x)$ monotonically increases from 0 as x moves away from a reference point x^r inside the mode to infinity. We modulate $f(x)$:

$$\begin{cases} \dot{x} = M(x)f(x) \quad \text{with} \quad M(x) = E(x)D(x)E(x)^{-1} \\ E(x) = [\mathbf{r}(x)\ \mathbf{e}_1(x)\ ...\ \mathbf{e}_{d-1}(x)] \quad \mathbf{r}(x) = \frac{x-x^r}{\|x-x^r\|} \\ D(x) = \mathbf{diag}(\lambda_r(x), \lambda_e(x), \lambda_e(x), ..., \lambda_e(x)) \\ \lambda_r(x) = 1 - \Gamma(x) \quad \lambda_e(x) = 1 \end{cases} \tag{2}$$

where $M(x)$ is constructed through eigenvalue decomposition. The full-rank basis consists of a reference direction $\mathbf{r}(x)$ stemming from x^r towards x, and $d-1$ directions that span the hyperplane orthogonal to $\nabla\Gamma(x)$. In other words, all directions $\mathbf{e}_1(x)...\mathbf{e}_{d-1}(x)$ are tangent to the closest cut except $\mathbf{r}(x)$. By modulating only the diagonal component $\lambda_r(x)$ with $\Gamma(x)$, we have $\lambda_r(x) \to 0$ as x approaches the closest cut, effectively zeroing out the velocity penetrating the cut while preserving velocity tangent to the cut.

Policy	Reachability	Invariance	No Noise	Small Noise	Large Noise
BC	✗	✗	88.9	72.4	58.6
BC+mod	✗	✓	91.9	83.6	76.0
DS	✓	✗	100	97.0	86.9
DS+mod	✓	✓	**100**	**100**	**100**

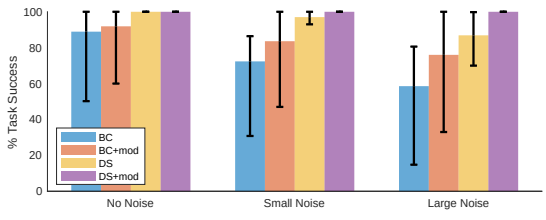

Fig. 4. Single mode reaching task success rate (%). For each randomly generated convex mode, we sample 100 starting states and compute the average success rate for 50 trials. As we start to sample out of distribution by adding more noise, the BC's success rate degrades more rapidly than the DS policy's. After modulation DS (+mod) maintains a success guarantee which BC (+mod) lacks due to the base policy's lack of a stability guarantee.

IV. Experiments

A. Single Mode Invariance and Reachability

We show quantitatively both reachability and invariance are necessary to achieve 100% task success rate in figure 4. We compare DS and a NN-based BC policy (denoted as BC) to represent policies with and without a stability guarantee. Figure 3 shows that policy rollouts start to fail (turn red) as increasingly larger perturbations are applied to the starting states; however, DS only suffers from invariance failures, while BC suffers from both invariance and reachability failures (due to diverging flows and spurious attractors). Figure 3 (right) shows that all flows are bounded within the mode for both DS and BC after two cuts. In the case of DS, flows originally leaving the mode are now redirected to the attractor by the cuts; in the case of BC, while no flows leave the mode after modulation, spurious attractors are created, leading to reachability failures.

B. Multi-Modal Reactivity

We now show empirically having a reactive discrete plan is insufficient to guarantee task success without mode invariance for tasks with multiple modes. Consider the task introduced in Fig. 1: scooping and transporting soup. Formally, we define four modes: (a) starting empty spoon, (b) sensing the spoon is in contact with the soup, (c) sensing the spoon has soup on it, and (d) sensing the spoon has arrived at a target location. During successful demonstrations, we observe the following discrete transitions a (reaching) $\Rightarrow$ b (scooping) $\Rightarrow$ c (transporting) $\Rightarrow$ d (done) . Invariance of mode b enforces contact during scooping and invariance of mode c constrains the spoon orientation to avoid spilling. One might assume having a task automaton is sufficient to guarantee task success without modulation, as it only needs to replan a finite number of times assuming a finite number of perturbations; however, not enforcing mode invariance can

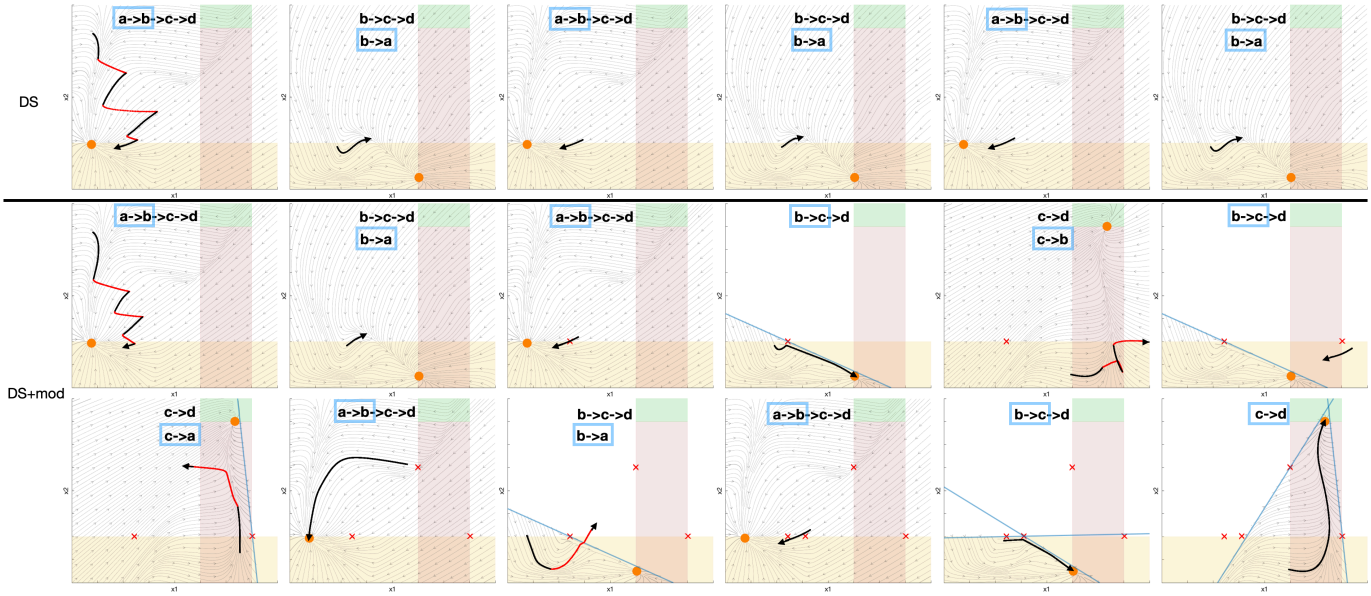

Fig. 5. Rollouts of multi-step scooping task under perturbations. The orange circle indicates the attractor for the current mode. We show the mode sequence planned by the automaton at the top of each sub-figure with the blue bounding box indicating the current mode transition actually detected. The trajectory in black is the rollout following the original policy, and the trajectory in red is driven by perturbations. The first row shows DS policies sequenced by an automaton but without boundary estimation can lead to looping. The second and third rows show modulation can prevent looping and eventually allow the system to reach the goal mode despite repeated perturbations. Please check the video in the project page.

lead to looping at the discrete level, and ultimately renders the goal unreachable, as depicted in the top row of Fig. 5. In contrast, looping is prevented when modulation is enabled, as the system experiences each invariance failure only once.

V. Conclusion

In this paper, we introduce *temporal logic imitation* as the problem of learning plan-satisficing motion policies. We identify one challenge of applying LfD methods to multi-step tasks as being that the learned controllers do not necessarily satisfy the bisimulation criteria. To address this issue, we propose a DS-based approach that can iteratively estimate mode boundaries to ensure invariance and reachability. Combining the task-level reactivity of a task automaton and the motion-level reactivity of DS, we arrive at an imitation learning system that can robustly perform a multi-step scooping task under arbitrary perturbations given only a few demonstrations.

References

[1] H. Ravichandar, A. S. Polydoros, S. Chernova, and A. Billard, "Recent advances in robot learning from demonstration," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 3, pp. 297–330, 2020.

[2] A. Mandlekar, D. Xu, R. Martín-Martín, S. Savarese, and L. Fei-Fei, "Learning to generalize across long-horizon tasks from human demonstrations," *arXiv preprint arXiv:2003.06085*, 2020.

[3] O. Kroemer, C. Daniel, G. Neumann, H. Van Hoof, and J. Peters, "Towards learning hierarchical skills for multi-phase manipulation tasks," in *2015 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2015, pp. 1503–1510.

[4] B. Akgun, M. Cakmak, J. W. Yoo, and A. L. Thomaz, "Trajectories and keyframes for kinesthetic teaching: A human-robot interaction perspective," in *Proceedings of the seventh annual ACM/IEEE international conference on Human-Robot Interaction*, 2012, pp. 391–398.

[5] S. Niekum, S. Chitta, A. G. Barto, B. Marthi, and S. Osentoski, "Incremental semantically grounded learning from demonstration." in *Robotics: Science and Systems*, vol. 9. Berlin, Germany, 2013, pp. 10–15 607.

[6] N. Figueroa and A. Billard, "A physically-consistent bayesian non-parametric mixture model for dynamical system learning." in *CoRL*, 2018, pp. 927–946.

[7] S. Ross, G. Gordon, and D. Bagnell, "A reduction of imitation learning and structured prediction to no-regret online learning," in *Proceedings of the fourteenth international conference on artificial intelligence and statistics*. JMLR Workshop and Conference Proceedings, 2011, pp. 627–635.

[8] 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.

[9] H. Kress-Gazit, M. Lahijanian, and V. Raman, "Synthesis for robots: Guarantees and feedback for robot behavior," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 1, pp. 211–236, 2018.

[10] S. M. Khansari-Zadeh and A. Billard, "A dynamical system approach to realtime obstacle avoidance," *Autonomous Robots*, vol. 32, no. 4, pp. 433–454, 2012.