RAIL: Reachability-Aided Imitation Learning for Safe Policy Execution

Wonsuhk Jung, Dennis Anthony, Utkarsh A. Mishra, Nadun Ranawaka Arachchige, Matthew Bronars, Danfei Xu*, Shreyas Kousik*
Georgia Institute of Technology, Carnegie Mellon University
Email: {wonsuhk.jung, danthony33, umishra31, nadun.ranawaka, mbronars, danfei}@gatech.edu, shreyas.kousik@me.gatech.edu

Abstract: Imitation learning (IL) has shown great success in learning complex robot manipulation tasks. However, there remains a need for practical safety methods to justify widespread deployment. In particular, it is important to certify that a system obeys hard constraints on unsafe behavior in settings when it is unacceptable to design a tradeoff between performance and safety via tuning the policy (i.e. soft constraints). This leads to the question, how does enforcing hard constraints impact the performance (meaning safely completing tasks) of an IL policy? To answer this question, this paper builds a reachability-based safety filter to enforce hard constraints on IL, which we call Reachability-Aided Imitation Learning (RAIL). Through evaluations with state-of-the-art IL policies in mobile robots and manipulation tasks, we make two key findings. First, the highest-performing policies are sometimes only so because they frequently violate constraints, and significantly lose performance under hard constraints. Second, surprisingly, hard constraints on the lower-performing policies can occasionally increase their ability to perform tasks safely. Finally, hardware evaluation confirms the method can operate in real time.

Keywords: Robot Safety, Imitation Learning

1 Introduction

Model-free learning has shown great success in the robotics domain, enabling robots to acquire complex behaviors without explicit modeling of task dynamics. Among these approaches, offline imitation learning (IL) has emerged as a powerful paradigm, allowing robots to learn from demonstrations without the need for extensive environmental interactions or carefully designed reward functions. Recent advancements in IL have demonstrated impressive generalization [1], performance [2], and the ability to generate temporally consistent plans [3]. In this work, we aim to build on top of effective IL frameworks.

That said, IL models still face significant challenges when deployed in real-world scenarios, particularly in safety-critical applications where performance must consider both task success and adherence to safety specifications. These challenges stem from several factors. The absence of physical interaction during training can lead to compounding errors when executing planned actions [4]. IL models tend to be brittle when encountering scenarios beyond their training data [5, 4]. Pre-trained models may struggle to adapt to dynamic changes in the environment during inference. In the worst case, these factors can cause a robot to act unsafely by colliding with its environment and causing damage. However, if we were to enforce hard safety constraints on a policy, it stands to reason that we may simultaneously sacrifice performance.

This paper seeks to understand the potential impact of imposing hard constraints on an IL policy. Doing so first requires a strategy for enforcing hard constraints on IL. To this end, we adapt tech-

Workshop on Safe and Robust Robot Learning for Operation in the Real World (SAFE-ROL) at CoRL 2024.

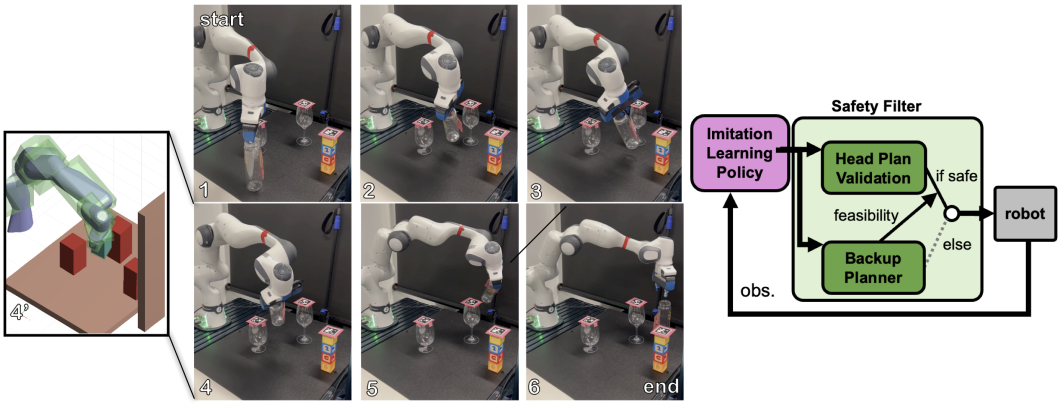

Figure 1: Our RAIL framework applied to a real-world robotic manipulation task. **Left**: A Franka robot arm safely executes a pick-and-place task among delicate obstacles. **Right**: System diagram illustrating how RAIL integrates an imitation learning policy with a safety filter, using plan validation and a failsafe planner to enforce hard constraints.

niques from model-based robot motion planning to propose **R**eachability-**A**ided **IL** (**RAIL**). Our approach introduces a safety filter that acts as a supervisory layer over the learned policy by leveraging recent advancements in continuous-time collision checking and reachability analysis to provide robust safety guarantees in a three-dimensional workspace. To achieve this in a robotics manipulation task, one critical challenge is efficiently computing the continuous-time swept volume of a multi-link manipulator arm given the planned action sequence output by the motion policy, such that the policy can run in real-time. We address this challenge with a novel extension of prior work on polynomial zonotopes [6, 7] to overapproximate the robot's swept volume for fast, accurate collision checking in continuous time.

By integrating this safety filter with IL policies, we find that our method allows for the exploitation of the strengths of learned policies in generating complex behaviors while enforcing hard constraints over an infinite time horizon. Importantly, RAIL is composable with different state-of-the-art IL frameworks, including Diffusion Policy [3] and ACT [8], suggesting broad applicability.

We evaluate the impact of hard constraints on IL via a series of increasingly complex tasks: navigating a 2D maze in simulation, manipulating objects in clutter in simulation, and maneuvering Franka arm hardware to transport a bottle while avoiding collisions with wine glasses and stacked blocks. As expected, RAIL significantly improves policy safety across all tasks, achieving 0% collision rates compared to 5-35% for baseline policies, while maintaining high task success rates. We find that the highest-performing seeds from these baseline policies also suffer a performance drop under hard constraints. However, RAIL, in some cases excitingly, aids IL policies in discovering new behavioral modalities that safely complete the task, even when the IL policies alone fail to do so. This demonstrates that incorporating hard safety constraints can lead to *increased performance*, challenging the notion that they are inherently in opposition.

2 Related Work

Offline Imitation Learning. Offline imitation learning (IL) algorithms typically learn from demonstrations through supervised regression [9, 10], offline reinforcement learning [11, 12, 13, 14], or generative models [15, 16, 17, 18, 19, 20, 21, 22, 3]. Notably, Diffusion Policy [3] introduces a powerful diffusion-based framework for offline IL. It learns the conditional distribution of action trajectory given the history of observations and executes the planned action sequence in a receding horizon fashion. While such policies can learn complex behaviors effectively, their practical deployment presents several challenges: (1) the absence of physical interaction results in compounding errors while planning [4], and (2) these methods tend to be fragile when encountering scenarios beyond their training data [5, 4]. Consequently, such policies can be unsafe. To address these issues,

our method proposes a safety filter [23, 24] for offline IL policies to exploit the strengths of these policies while guaranteeing safety.

Safety for Learned Policies. A wide variety of approaches seek to enforce safety with learned policies. Many approximately enforce collision avoidance constraints (e.g., [25, 26, 27, 28, 29]) or more complex constraints (e.g., staying close to contact [30]); broadly speaking, the unifying concept is to incorporate a safety specification as a penalty or loss in training or test-time inference. Other methods focus on probabilistic safety constraints [31, 32], ensuring that the likelihood of safety violations remains below a specified threshold. In contrast, our method focuses on enforcing hard constraints as seen in [33, 34, 35, 36, 37, 38], including our prior work [39, 40]. The common trend across these is to analyze *invariant sets* [34, 35, 36] or *reachable sets* [38, 37, 39, 40] of the trajectories of an uncertain dynamical system, and develop strict fallback or failsafe policies that can replace a learned policy's actions (i.e., a safety filter). The key challenge across all such safety methods is to balance safety against performance, which requires careful design of the safety filter. While some hard constraint methods for reinforcement learning see a drop or no change in performance [37, 38], others see an increase [40], suggesting the same may be true for IL.

Safety for Manipulators. Manipulator collision avoidance is a classical problem that is challenging due to the nonlinear, set-valued map between state space and workspace [41], and due to the need to represent collision avoidance in continuous time [42]. Recent progress in numerical representation has enabled new ways to compute and differentiate through continuous-time collision checks with polynomial zonotopes [6, 7, 43], capsules [44, 45], or learned collision functions [46, 47]. A further challenge is to consider uncertain dynamics, which can be handled with polynomial zonotopes [7] or control barrier functions [48]. In this space, we extend our recent work [6, 7] from parameterized trajectories to trajectories planned by an IL policy.

3 Problem Statement and Preliminaries

To understand the impact of hard constraints on IL, we must first address the following:

Problem 1. *Plan and control the motion of a robot to complete tasks while guaranteeing the robot avoids collision with obstacles in continuous time as a hard constraint.*

Robot, Environment, and Safety. We consider a robot with state $x(t) \in \mathcal{X} \subseteq \mathbb{R}^n$ at time t and action $a(t) \in \mathcal{A}$ (e.g., desired end effector poses), executed by the low-level controller (e.g., Operational Space Controller [49].) The robot's dynamics are represented as $\dot{x}(t) = f(x(t), a(t))$. The robot occupies a workspace $\mathcal{W} \subset \mathbb{R}^3$ via a forward occupancy map $\text{FO}(x(t)) \subset \mathcal{W}$. We denote workspace obstacles $\mathcal{W}_{\text{obs}} \subset \mathcal{W}$. The robot receives observations $o(t) \in \mathcal{O}$ (e.g., joint states or RGB-D images) at a fixed rate. We assume full observability, where observations $o(t)$ include privileged information for safety verification, such as the true robot state, task-relevant object poses, and object physics. Given a trajectory $x : [0, \infty) \to \mathcal{Q}$, safety means $\text{FO}(x(t)) \cap \mathcal{W}_{\text{obs}} = \emptyset$ for all time, plus other task-dependent constraints (e.g., max velocity).

Set Operations. We use the following operations to compute swept volumes (i.e., forward occupancy) to enforce hard constraints. For $p_1, p_2 \in \mathbb{R}^n$, we denote the line segment between them $\overline{p_1 p_2}$. For a pair of set $\mathcal{X}, \mathcal{Y} \subset \mathbb{R}^n$, we make use of the Minkowski sum $\mathcal{X} \oplus \mathcal{Y} = \{x + y \mid x \in \mathcal{X}, y \in \mathcal{Y}\}$, Cartesian product $\mathcal{X} \times \mathcal{Y}$, set cross product $\mathcal{X}^{\times}\mathcal{Y} = \{x \times y \mid x \in \mathcal{X}, y \in \mathcal{Y}\}$. For a matrix set $\mathcal{X} \subset \mathbb{R}^{n \times k}$ and a vector set $\mathcal{Y} \subset \mathbb{R}^k$, we define set product as $\mathcal{X}\mathcal{Y} = \{XY \in \mathbb{R}^n \mid X \in \mathcal{X}, Y \in \mathcal{Y}\}$. For a function f, we denote $f(\mathcal{X}) = \{f(x) \mid x \in \mathcal{X}\}$. We apply linear operations on the left or right as appropriate (e.g., $A\mathcal{X}$ or $\mathcal{X}A$). We numerically implement all set operations with polynomial zonotopes [50] as per [7, §III-C]. These operations are detailed in Appendix A.1.

4 Framework Overview

We aim to incorporate hard safety guarantees in state-of-the-art imitation learning frameworks directly at inference. Our method adopts reachability-based trajectory design and continuous time

collision checking to guarantee safety while leveraging the multi-modal behavior learning of a baseline motion policy. We provide the details for each design segment of our framework below.

4.1 Motion Policy

We consider the paradigm of offline IL where we have a dataset $\mathcal{D} = \{(o,a)\}$ containing pre-recorded demonstration trajectories (sequences of observation-action pairs), of successfully completing the task objective. We seek to train an imitation learning policy that maps observations to action motion plans $\pi : \mathcal{O} \to \mathcal{A}^{T_p}$ where $T_p \in \mathbb{N}$ is the plan length. Finally, we assume the obstacles $\mathcal{W}_{\text{obs}}$ may not be present in the training data, or may change from training to test time, but can be observed at test time. In this paper, we mainly consider Diffusion Policy [3] as our motion policy and learn the target conditional distribution $p_\pi(a_{t:t+T_p}|o_t)$. We note that our method can generalize to any similar offline IL algorithms and include additional base algorithms in experiments.

4.2 Reachability-Assisted Imitation Learning

This section presents our first contribution: a framework that interfaces an imitation learning policy with a model-based backup planner using reachability analysis.

We adopt receding-horizon planning, where the robot plans over a finite horizon T_p, but only executes the first T_a steps before re-planning.

The key idea is to maintain a safe failsafe [37] backup plan using reachability analysis while verifying the safety of the proposed plan by imitation learning policy. Towards that, we build upon the safety filter concept from [23, 24], where a safe policy intervenes only when the nominal policy is verified to be unsafe. We especially override the definition of safety with not only the safety of the plan proposed by the imitation learning policy but also verifying whether the backup plan can be attached at the end of the plans to execute (i.e., $a_{t:t+T_a}$.) The framework comprises two key safety verification algorithms for the proposed plan $a_{t:t+T_p}$.

Head Plan Verification. First, we introduce the head-plan safety verifier, which checks the continuous-time safety of arbitrary plans from the imitation learning policy using reachability analysis. We verify the safety of the plan $a_{t:t+T_a}$, which we call as *head-plan*. We detail this verifier, particularly for manipulators, in Section 5.

Backup Plan Creation. Second, we verify the existence of a backup plan, using a reachability-based model-based planner similar to [51, 6]. This planner generates failsafe plans, parameterized by low-dimensional (e.g., number of joints for a manipulator) parameter k, where a failsafe maneuver is a stopping action. Note that the failsafe is not concerned with task completion, only safety, so using a low-dimensional parameter is sufficient.

At each iteration, the planner solves the following nonlinear optimization problem to find a safe plan parameter that satisfies the safety requirements:

$$k^* = \underset{k}{\operatorname{argmin}}\, f(x_0,k) \text{ s.t. } g(x_0,k) < 0 \tag{1}$$

where f is an objective function and g is the reachability-based safety constraint. The objective function includes, but is not restricted to the projection objective, meaning minimizing the difference between the proposed plan from the imitation learning policy and the parameterized backup plan. The solution of (1) compactly represents the desired trajectory $x(t)$ whose initial state is x_0 and safe throughout the horizon. By discretizing this trajectory and taking the desired state as the action representation, and using a low-level controller to track this trajectory, we formulate a backup policy as $\pi_{\text{back}} : \mathcal{X} \to \mathcal{A}^{T_b}$, where T_b is the horizon of the parameterized failsafe trajectory. The detailed algorithm and pseudo-code are provided in Appendix B.

5 Continuous-Time Collision Checking

We now detail our approach to verifying head plan safety to enable hard collision-avoidance constraints for IL. This addresses two key challenges. First, most IL policies output discrete states/actions, but collisions occur in continuous time. Second, unlike in our past work on safe RL for mobile robots [40, 39], we cannot simply replace the IL output with parameterized plans (i.e., backup plans), because this would collapse the richness of the IL policy.

To solve these challenges, we propose a novel approach extension of our prior work [7, Sec. VIII-A] to compute continuous-time swept volume for the head plan output by the IL motion policy. First, we review forward occupancy, then explain our approach and implementation details.

Manipulator Forward Occupancy. We compute FO in the following well-known way (c.f., [41, Ch. 3.3]). Suppose the robot has n revolute joints and configuration $q \in \mathcal{Q} \subseteq (\mathrm{SO}(1))^n$. Suppose that the robot's j^{th} link has a local reference frame at the center of its joint with link $j-1$ and rotates about the local vector w_j, and $j=0$ is the baselink/inertial frame. For $q \in \mathcal{Q}$, let q_j denote its j^{th} element. Then $R_j^{j-1}(q_j, w_j) = I_{3\times 3} + \sin(q_j) w_j^{\times} + (1-\cos(q_j))(w_j^{\times})^2$ is the rotation from frame $j-1$ to frame j per Rodrigues' rotation formula. Suppose t_j^{j-1} is the origin of frame j in frame $j-1$, and let $\mathcal{L}_j \subset \mathbb{R}^3$ represent the volume of the j^{th} link in the j^{th} reference frame. Then link j's forward occupancy is

$$\mathrm{FO}_j(q) = \{t_j(q)\} \oplus (R_j(q)\mathcal{L}_j) \subset \mathcal{W}, \tag{2}$$

where $R_j(q) = \prod_{i=1}^{j} R_j^{i-1}(q_j)$ and $t_j(q) = \sum_{i=1}^{j} R_i(q) t_i^{i-1}$. Letting $n = \dim(\mathcal{Q})$, the forward occupancy of the robot is $\mathrm{FO}(q) = \bigcup_{j=1}^{n} \mathrm{FO}_j(q)$.

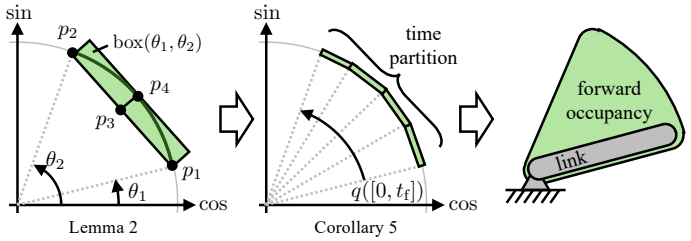

Figure 2: We overapproximate the sines and cosines of the robot's joint angles along a trajectory to enable overapproximating the robot's swept volume.

Method Overview. To proceed to overapproximate the rotation of a single joint using a box (Lem. 2), which we use to overapproximate the joint's rotation matrices (Lem. 3), and then the forward occupancy (Thm. 4 and Cor. 5).

Lemma 2. *Suppose a 1-D revolute joint travels counterclockwise (CCW) from an angle θ_1 to $\theta_2 > \theta_1$. Map this motion to the unit circle* $\mathrm{SO}(1)$ *as $p_1 = (\cos(\theta_1), \sin(\theta_1))$ and $p_2 = (\cos(\theta_2), \sin(\theta_2))$. Define $p_3 = \frac{1}{2}(p_1 + p_2)$ and $p_4 = (\cos(\frac{1}{2}(\theta_1+\theta_2)), \sin(\frac{1}{2}(\theta_1+\theta_2))))$. Let $p_5 = \frac{1}{2}(p_3 + p_4)$. Then, for every $\theta \in [\theta_1, \theta_2]$, we have*

$$(\cos(\theta), \sin(\theta)) \in \mathrm{box}(\theta_1, \theta_2) := \overline{p_1 p_2} \oplus \overline{(p_3 - p_5)(p_4 - p_5)}.$$

Proof. See Appendix A.2 for proof. □

Next, we overapproximate the set of rotation matrices corresponding to a single revolute joint. To enable this, we consider Rodrigues' rotation formula as a function

$$\mathrm{RO}(p, w) = I_{3\times 3} + p_2 w^{\times} + (1-p_1)(w^{\times})^2 \tag{3}$$

for a point $p = (p_1, p_2) \in \mathbb{R}^2$ and a rotation axis $w \in \mathbb{R}^3$.

Lemma 3. *Consider a revolute joint with local rotation axis $w \in \mathbb{R}^3$ rotating through an interval $[\theta^-, \theta^+]$. Then, for any rotation matrix R corresponding to this rotation, we have $R \in \mathrm{RO}(\mathrm{box}(\theta^-, \theta^+)), w)$*

Proof. This follows from Lemma 2, since Rodrigues' formula is applied to every element of Θ. □

Now we can overapproximate the swept volume of each link of the robot given bounded rotations of each joint:

Theorem 4. *Suppose each joint rotates through an interval $\Theta_j = [\theta_j^-, \theta_j^+]$. Let $\Theta = \Theta_1 \times \cdots \times \Theta_n$, and define $\mathcal{R}_j(\Theta) = \prod_{i=1}^{j} \mathrm{RO}(\mathrm{box}(\theta_j^-, \theta_j^+), w_j)$ and $\mathcal{T}_j(\Theta) = \sum_{i=1}^{j} \mathcal{R}_j(\Theta) t_i^{i-1}$. Then, for any $q \in \Theta$, $\mathrm{FO}_j(q) \subset \{\mathcal{T}_j(\Theta)\} \oplus (\mathcal{R}_j(\Theta, w_j)\mathcal{L}_j)$, where $\mathcal{R}_j(\Theta, w_j)\mathcal{L}_j$ is a set product.*

Proof. This follows directly from Lemma 3 and the forward occupancy definition (2). □

Thm. 4 would be very conservative if applied for all joints along an entire trajectory, since the joints are treated independent of time. We mitigate this by partitioning time, which reduces the overapproximation as shown in Fig. 2:

Corollary 5. *Consider a trajectory $q : [0, t_\mathrm{f}] \to \mathcal{Q}$. Partition time into $[0, t_\mathrm{f}] = \bigcup_{i=1}^{m-1} [i\delta, (i+1)\delta]$ with $\delta = \frac{t_\mathrm{f}}{m}$ and $m \in \mathbb{N}$. For each $i = 1, \cdots, m-1$, suppose for each j^{th} joint angle, we know its min and max values: $q_j([i\delta, (i+1)\delta]) \in \Theta_{j,i} = [q_{j,i}^-, q_{j,i}^+]$. Then we construct $\Theta_i = \Theta_{1,i} \times \cdots \times \Theta_{n,i}$ and have*

$$\mathrm{FO}(q([0, t_\mathrm{f}])) \subset \bigcup_{j=1}^{n} \bigcup_{i=1}^{m-1} \{\mathcal{T}_j(\Theta_i)\} \oplus (\mathcal{R}_j(\Theta_i, w_j)\mathcal{L}_j). \tag{4}$$

Proof. This follows directly from Thm. 4. □

The challenge with using Cor. 5 is bounding $q(t)$ in each partition interval. For the head plan, we know $\dot{q}(t)$ at every t and assume a maximum acceleration $\overline{\ddot{q}}_j$, so we use the standard kinematic equation: $q_{j,i}^- \geq q_j(i\delta) - \delta \dot{q}_j(i\delta) - \frac{1}{2}\overline{\ddot{q}}_j \delta^2$, and $q_{j,i}^+$ similarly. The bound $\overline{\ddot{q}}_j$ is found per joint from demonstration data and augmented by a small $\varepsilon > 0$.

6 Experimental Results

In this section, we seek to (1) confirm that RAIL does indeed enforce hard constraints on IL policies, and (2) understand the impact of these constraints. We hypothesize that RAIL allows learning policies that leverage the long-horizon reasoning capability inherited from a base imitation learning policy while being robust to unseen danger using the short-horizon model-based reasoning of the backup planner. It is worth noting, that our experiment does not assume that all safety definitions are present in the offline training dataset. In addition, we consider that our model-based safety filter is privileged with the safety specification at inference runtime.

Baselines. We consider two powerful imitation learning policies: Diffusion Policy (**DP**) [3] and Action Chunking with Transformers (**ACT**) [8]. These policies are trained using offline datasets. To understand the impact of *soft constraints*, we also formulate a Safety-guided Diffusion Policy (**SGDP**) leveraging classifier-guided reverse sampling [52] (See Appendix C.2 for detail). Finally, we also evaluate the performance of a reachability-based backup planner [53, 6] (**BP**) as a model-based baseline with hard constraints.

Evaluation metrics. We evaluate the success rate as reaching the goal in the target environment (Succ %). We measure the number of transitions that led to a collision between the agent and the obstacles and report it as a percentage of all the transitions (Col %). Most importantly, we report the percentage of successful episodes where none of the safety conditions were violated throughout the trajectory (SSucc %). Lastly, to understand how efficient each agent is at completing tasks, we report the average number of timesteps of the trajectories (Horizon).

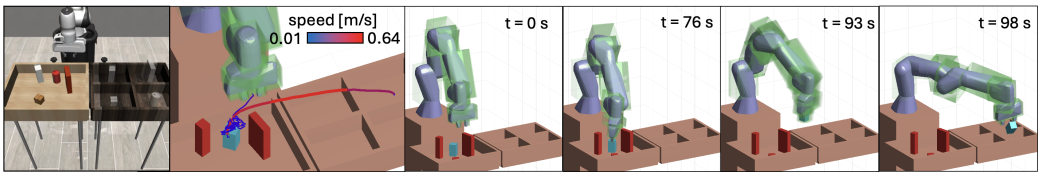

Figure 3: **(Leftmost)** Illustration of Can Pick-Place task: A can must be picked from a table and placed in a target bin. **(Rightward to Rightmost)** Timelapse of an episode involving navigating through a tight gap. A diffusion policy repeatedly proposed unsafe plans, until eventually RAIL validated a safe path to picking. The IL policy again proposed unsafe plans to move the picked object, until RAIL validated a safe path outwards to placing.

Table 1: Results on Maze Medium and Maze Large environments: We run each of the policies for 100 episodes and repeat for 3 seeds. We report the average values below.

(a) Maze Medium environment

Method	Succ.	Col.	SSucc.	Horizon
DP	**100.00**	5.17	15.00	**278.39**
BP	58.00	**0.00**	58.00	691.62
SGDP	**100.00**	4.64	18.00	284.84
RAIL (Ours)	97.00	**0.00**	**97.00**	309.77

(b) Maze Large environment

Method	Succ.	Col.	SSucc.	Horizon
DP	**100.00**	5.66	15.00	470.48
BP	49.00	**0.00**	49.00	921.12
SGDP	**100.00**	4.85	18.00	**447.5**
RAIL (Ours)	95.00	**0.00**	**95.00**	532.37

6.1 Simulation: Safe Planning in Maze

Setup. We evaluate our framework in the PointMaze environment [54], where the objective is for a double integrator mobile robot to navigate a maze to a target position. The robot's state is 2-D position and velocity and its action is 2-D force. Safety means avoiding collision with the maze walls and limiting velocity. We consider a diffusion policy trained on the offline D4RL dataset [54] as our baseline and [53] as our backup planner. We run experiments on Medium and Large mazes to see how RAIL performs as task complexity increases. Details are found in Appendix C.3.

Results and Discussion. We see similar results in both the Medium (Table 1a) and Large mazes (Table 1b). It is evident that even if DP can solve the task, it collides with the walls. In this experiment, we specifically try to make this more obvious by deliberately sabotaging our policy with a new safety specification at runtime (i.e., the offline dataset includes the trajectory colliding with obstacles). On the other hand, BP [53] guarantees safety but shows a low success rate due to lack of long-horizon reasoning capability. SGDP implicitly informs and guides the action sampling from diffusion policy and hence leads to decrease the number of collisions, but cannot guarantee constraint satisfaction. Our proposed method, RAIL leverages the conservative backup planner only when diffusion policy suggests unsafe actions, enabling it to achieve the highest safe success rate despite hard constraints.

6.2 Simulation: Safe Planning in Pick-and-place

Table 2: Pick-Place Results and Increase of SSucc Rate Comparison

(a) Pick-Place environment results

Method	Succ.	Col.	SSucc.	Horizon
DP	**78.00**	27.20	58.00	**723.08**
ACT	74.00	35.39	62.00	1434.98
RAIL + DP	68.00	**0.00**	**68.00**	1385.34
RAIL + ACT	58.00	**0.00**	58.00	1294.18

(b) Increase of SSucc Rate per Policy Class

Method	Min.	Max.	Mean.
DP ($T_a = 8$)	**4.00**	**16.00**	**9.25**
DP ($T_a = 16$)	-14.00	10.00	2.25
ACT ($T_a = 8$)	-4.00	8.00	2.00
ACT ($T_a = 16$)	-18.00	4.00	-7.20

Setup. We now consider a more complex task derived from robomimic [55] that involves a single-arm manipulator picking up a can and placing it in a target bin. Safety means avoiding collisions between the robot and the environment while obeying joint position and velocity constraints. To enable picking, the collision constraint between the gripper and the can is ignored. Additionally,

the robot's behavior can be broken down into reaching, picking, and placing; so, we impose phase-specific constraints, especially in placing, where collisions between the can and the environment are unsafe. We represent all objects as bounding boxes (see Figure 3). We consider Diffusion Policy [3] and ACT [8] trained on 184 safe, successful demonstrations collected via robomimic, while we consider ARMTD [6] as our backup planner. Detailed setup is provided in Appendix C.4.

Results and Discussion. We summarize results in Table 2a. We see that the diffusion policy and ACT without RAIL have high success. This is because the task requires picking in clutter, so the policy learns to push the clutter aside, leading to a high collision rate; the SSucc numbers confirm this. On the other hand, with RAIL, the diffusion policy is able to perform many more tasks both safely and successfully, indicating that many of the scenarios were possible to achieve without collisions. We also see that RAIL incurs nearly twice the horizon of the unfiltered diffusion policy, indicating the policy and safety layer often oppose each other (see Figure 3, Table 2a).

We make a key surprising finding by comparing the safe-and-success rate across seeds of diffusion policy and ACT, shown in Table 2b. All of the policies considered had at least one seed (typically not the highest-performing) that significantly improved its SSucc when integrated with RAIL. In other words, *enforcing hard constraints can sometimes increase the performance of an IL policy*. As suggested by Figure 3, this is likely because (a) the safety filter forces the policy to behave more cautiously to carry out fine-grained tasks in clutter, where the constraints are always close to being active, and (b) lower-performing seeds may oppose the safety filter less often.

7 Real World Evaluation

We demonstrate RAIL with a diffusion policy in the real world on a Franka Panda robot arm shown in Figure 1. This indicates RAIL enforces hard constraints in real time and under sensing uncertainty.

Setup. The goal is to pick and place the target object at a goal while avoiding delicate wine glasses and stacked blocks (represented for the safety filter as bounding boxes and detected with AprilTags [56] via an Azure Kinect DK RGBD camera.) Other safety specifications are as in simulation. We train a diffusion policy on 75 teleoperated, safe demonstrations (collected with Gello [57]), and use ARMTD [6] serves as the backup planner. We assume all obstacles stay static. Details are provided in Appendix C.5.

Results and Discussion. We found that the diffusion policy with RAIL solves the task safely as expected (see Figure 1), while the vanilla diffusion policy failed to comply with the safety specifications. On average, RAIL's computation time was 0.42 ± 0.05 seconds per plan, based on 500 planning runs, demonstrating that it operates in real time. However, it is important to note that the primary focus of this paper is not on achieving maximally efficient implementation, which we leave to future work.

8 Conclusion and Limitations

This paper presented Reachability Assisted Imitation Learning (RAIL), a novel framework that combines deep imitation learning with a reachability-based safety filter to enforce hard constraints. Simulated and real-world evaluations show that RAIL can indeed enforce hard constraints (0% collision rate) on state-of-the-art IL policies in real-time. Excitingly, in some cases RAIL *increases* policy performance, indicating hard constraints are not always a negative tradeoff.

Limitations. RAIL has two main limitations. First, it assumes access to predicted observations after action execution, which can be a bottleneck for scaling to policies conditioned on high-dimensional observations (e.g., images, languages) where making accurate predictions is challenging. Second, applying RAIL can result in longer task completion times due to potential conflicts between IL policy and the safety filters. A possible future direction is to incorporate safety filters directly into the IL policy planning process to generate inherently safer plans and minimize these conflicts.

Acknowledgments

We thank Georgia Tech AMPF for supporting this work. We also express our gratitude to Chuizheng Kong for valuable discussions on the hardware experiment, Shuo Cheng for contributing pose estimation code for the hardware experiment, and Vaibhav Saxena for assistance with setting up Gello.

References

[1] O. X.-E. Collaboration, A. O'Neill, A. Rehman, A. Gupta, A. Maddukuri, A. Gupta, A. Padalkar, A. Lee, A. Pooley, A. Gupta, A. Mandlekar, A. Jain, A. Tung, A. Bewley, A. Herzog, A. Irpan, A. Khazatsky, A. Rai, A. Gupta, A. Wang, A. Kolobov, A. Singh, A. Garg, A. Kembhavi, A. Xie, A. Brohan, A. Raffin, A. Sharma, A. Yavary, A. Jain, A. Balakrishna, A. Wahid, B. Burgess-Limerick, B. Kim, B. Schölkopf, B. Wulfe, B. Ichter, C. Lu, C. Xu, C. Le, C. Finn, C. Wang, C. Xu, C. Chi, C. Huang, C. Chan, C. Agia, C. Pan, C. Fu, C. Devin, D. Xu, D. Morton, D. Driess, D. Chen, D. Pathak, D. Shah, D. Büchler, D. Jayaraman, D. Kalashnikov, D. Sadigh, E. Johns, E. Foster, F. Liu, F. Ceola, F. Xia, F. Zhao, F. V. Frujeri, F. Stulp, G. Zhou, G. S. Sukhatme, G. Salhotra, G. Yan, G. Feng, G. Schiavi, G. Berseth, G. Kahn, G. Yang, G. Wang, H. Su, H.-S. Fang, H. Shi, H. Bao, H. B. Amor, H. I. Christensen, H. Furuta, H. Bharadhwaj, H. Walke, H. Fang, H. Ha, I. Mordatch, I. Radosavovic, I. Leal, J. Liang, J. Abou-Chakra, J. Kim, J. Drake, J. Peters, J. Schneider, J. Hsu, J. Vakil, J. Bohg, J. Bingham, J. Wu, J. Gao, J. Hu, J. Wu, J. Wu, J. Sun, J. Luo, J. Gu, J. Tan, J. Oh, J. Wu, J. Lu, J. Yang, J. Malik, J. Silvério, J. Hejna, J. Booher, J. Tompson, J. Yang, J. Salvador, J. J. Lim, J. Han, K. Wang, K. Rao, K. Pertsch, K. Hausman, K. Go, K. Gopalakrishnan, K. Goldberg, K. Byrne, K. Oslund, K. Kawaharazuka, K. Black, K. Lin, K. Zhang, K. Ehsani, K. Lekkala, K. Ellis, K. Rana, K. Srinivasan, K. Fang, K. P. Singh, K.-H. Zeng, K. Hatch, K. Hsu, L. Itti, L. Y. Chen, L. Pinto, L. Fei-Fei, L. Tan, L. J. Fan, L. Ott, L. Lee, L. Weihs, M. Chen, M. Lepert, M. Memmel, M. Tomizuka, M. Itkina, M. G. Castro, M. Spero, M. Du, M. Ahn, M. C. Yip, M. Zhang, M. Ding, M. Heo, M. K. Srirama, M. Sharma, M. J. Kim, N. Kanazawa, N. Hansen, N. Heess, N. J. Joshi, N. Suenderhauf, N. Liu, N. D. Palo, N. M. M. Shafiullah, O. Mees, O. Kroemer, O. Bastani, P. R. Sanketi, P. T. Miller, P. Yin, P. Wohlhart, P. Xu, P. D. Fagan, P. Mitrano, P. Sermanet, P. Abbeel, P. Sundaresan, Q. Chen, Q. Vuong, R. Rafailov, R. Tian, R. Doshi, R. Mart'in-Mart'in, R. Baijal, R. Scalise, R. Hendrix, R. Lin, R. Qian, R. Zhang, R. Mendonca, R. Shah, R. Hoque, R. Julian, S. Bustamante, S. Kirmani, S. Levine, S. Lin, S. Moore, S. Bahl, S. Dass, S. Sonawani, S. Tulsiani, S. Song, S. Xu, S. Haldar, S. Karamcheti, S. Adebola, S. Guist, S. Nasiriany, S. Schaal, S. Welker, S. Tian, S. Ramamoorthy, S. Dasari, S. Belkhale, S. Park, S. Nair, S. Mirchandani, T. Osa, T. Gupta, T. Harada, T. Matsushima, T. Xiao, T. Kollar, T. Yu, T. Ding, T. Davchev, T. Z. Zhao, T. Armstrong, T. Darrell, T. Chung, V. Jain, V. Kumar, V. Vanhoucke, W. Zhan, W. Zhou, W. Burgard, X. Chen, X. Chen, X. Wang, X. Zhu, X. Geng, X. Liu, X. Liangwei, X. Li, Y. Pang, Y. Lu, Y. J. Ma, Y. Kim, Y. Chebotar, Y. Zhou, Y. Zhu, Y. Wu, Y. Xu, Y. Wang, Y. Bisk, Y. Dou, Y. Cho, Y. Lee, Y. Cui, Y. Cao, Y.-H. Wu, Y. Tang, Y. Zhu, Y. Zhang, Y. Jiang, Y. Li, Y. Li, Y. Iwasawa, Y. Matsuo, Z. Ma, Z. Xu, Z. J. Cui, Z. Zhang, Z. Fu, and Z. Lin. Open X-Embodiment: Robotic learning datasets and RT-X models. https://arxiv.org/abs/2310.08864, 2023.

[2] N. M. Shafiullah, Z. Cui, A. A. Altanzaya, and L. Pinto. Behavior transformers: Cloning k modes with one stone. *Advances in neural information processing systems*, 35:22955–22968, 2022.

[3] C. Chi, S. Feng, Y. Du, Z. Xu, E. Cousineau, B. Burchfiel, and S. Song. Diffusion policy: Visuomotor policy learning via action diffusion. *arXiv preprint arXiv:2303.04137*, 2023.

[4] J. A. D. Bagnell. An invitation to imitation. Technical Report CMU-RI-TR-15-08, Carnegie Mellon University, Pittsburgh, PA, March 2015.

[5] B. D. Argall, S. Chernova, M. Veloso, and B. Browning. A survey of robot learning from demonstration. *Robotics and autonomous systems*, 57(5):469–483, 2009.

[6] P. Holmes, S. Kousik, B. Zhang, D. Raz, C. Barbalata, M. J. Roberson, and R. Vasudevan. Reachable Sets for Safe, Real-Time Manipulator Trajectory Design. In *Proceedings of Robotics: Science and Systems*, Corvallis, Oregon, USA, 7 2020. doi:10.15607/RSS.2020.XVI.100.

[7] J. Michaux, P. Holmes, B. Zhang, C. Chen, B. Wang, S. Sahgal, T. Zhang, S. Dey, S. Kousik, and R. Vasudevan. Can't touch this: Real-time, safe motion planning and control for manipulators under uncertainty. *arXiv preprint arXiv:2301.13308*, 2023.

[8] T. Z. Zhao, V. Kumar, S. Levine, and C. Finn. Learning fine-grained bimanual manipulation with low-cost hardware. *arXiv preprint arXiv:2304.13705*, 2023.

[9] D. A. Pomerleau. Alvinn: An autonomous land vehicle in a neural network. *Advances in neural information processing systems*, 1, 1988.

[10] T. Zhang, Z. McCarthy, O. Jow, D. Lee, X. Chen, K. Goldberg, and P. Abbeel. Deep imitation learning for complex manipulation tasks from virtual reality teleoperation. In *2018 IEEE international conference on robotics and automation (ICRA)*, pages 5628–5635. IEEE, 2018.

[11] S. Levine, A. Kumar, G. Tucker, and J. Fu. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *arXiv preprint arXiv:2005.01643*, 2020.

[12] A. Kumar, A. Zhou, G. Tucker, and S. Levine. Conservative q-learning for offline reinforcement learning. *Advances in Neural Information Processing Systems*, 33:1179–1191, 2020.

[13] M. Janner, Q. Li, and S. Levine. Offline reinforcement learning as one big sequence modeling problem. *Advances in neural information processing systems*, 34:1273–1286, 2021.

[14] Y. Chebotar, Q. Vuong, K. Hausman, F. Xia, Y. Lu, A. Irpan, A. Kumar, T. Yu, A. Herzog, K. Pertsch, et al. Q-transformer: Scalable offline reinforcement learning via autoregressive q-functions. In *Conference on Robot Learning*, pages 3909–3928. PMLR, 2023.

[15] T. Pearce, T. Rashid, A. Kanervisto, D. Bignell, M. Sun, R. Georgescu, S. V. Macua, S. Z. Tan, I. Momennejad, K. Hofmann, et al. Imitating human behaviour with diffusion models. *arXiv preprint arXiv:2301.10677*, 2023.

[16] U. A. Mishra and Y. Chen. Reorientdiff: Diffusion model based reorientation for object manipulation. *arXiv preprint arXiv:2303.12700*, 2023.

[17] U. A. Mishra, S. Xue, Y. Chen, and D. Xu. Generative skill chaining: Long-horizon skill planning with diffusion models. In *Conference on Robot Learning*, pages 2905–2925. PMLR, 2023.

[18] U. A. Mishra, Y. Chen, and D. Xu. Generative factor chaining: Coordinated manipulation with diffusion-based factor graph. In *8th Annual Conference on Robot Learning*, 2024. URL https://openreview.net/forum?id=p6Wq6TjjHH.

[19] M. Reuss, M. Li, X. Jia, and R. Lioutikov. Goal-conditioned imitation learning using score-based diffusion policies. *arXiv preprint arXiv:2304.02532*, 2023.

[20] M. Reuss and R. Lioutikov. Multimodal diffusion transformer for learning from play. In *2nd Workshop on Language and Robot Learning: Language as Grounding*, 2023. URL https://openreview.net/forum?id=nvtxqMGpn1.

[21] M. Janner, Y. Du, J. B. Tenenbaum, and S. Levine. Planning with diffusion for flexible behavior synthesis. *arXiv preprint arXiv:2205.09991*, 2022.

[22] A. Ajay, Y. Du, A. Gupta, J. Tenenbaum, T. Jaakkola, and P. Agrawal. Is conditional generative modeling all you need for decision-making? *arXiv preprint arXiv:2211.15657*, 2022.

[23] K. P. Wabersich, A. J. Taylor, J. J. Choi, K. Sreenath, C. J. Tomlin, A. D. Ames, and M. N. Zeilinger. Data-driven safety filters: Hamilton-jacobi reachability, control barrier functions, and predictive methods for uncertain systems. *IEEE Control Systems Magazine*, 43(5):137–177, 2023.

[24] K.-C. Hsu, H. Hu, and J. F. Fisac. The safety filter: A unified view of safety-critical control in autonomous systems. *Annual Review of Control, Robotics, and Autonomous Systems*, 7, 2023.

[25] C. Dawson, Z. Qin, S. Gao, and C. Fan. Safe nonlinear control using robust neural lyapunov-barrier functions. In *Conference on Robot Learning*, pages 1724–1735. PMLR, 2022.

[26] Z. Qin, D. Sun, and C. Fan. Sablas: Learning safe control for black-box dynamical systems. *IEEE Robotics and Automation Letters*, 7(2):1928–1935, 2022.

[27] J. Carvalho, M. Baierl, J. Urain, and J. Peters. Conditioned score-based models for learning collision-free trajectory generation. In *NeurIPS 2022 Workshop on Score-Based Methods*, 2022.

[28] J. Carvalho, A. T. Le, M. Baierl, D. Koert, and J. Peters. Motion planning diffusion: Learning and planning of robot motions with diffusion models. In *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1916–1923. IEEE, 2023.

[29] Y. U. Ciftci, Z. Feng, and S. Bansal. Safe-gil: Safety guided imitation learning. *arXiv preprint arXiv:2404.05249*, 2024.

[30] P. Kicki, P. Liu, D. Tateo, H. Bou-Ammar, K. Walas, P. Skrzypczyński, and J. Peters. Fast kinodynamic planning on the constraint manifold with deep neural networks. *IEEE Transactions on Robotics*, 2023.

[31] K. Menda, K. Driggs-Campbell, and M. J. Kochenderfer. Ensembledagger: A bayesian approach to safe imitation learning. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5041–5048. IEEE, 2019.

[32] F. Castaneda, J. J. Choi, W. Jung, B. Zhang, C. J. Tomlin, and K. Sreenath. Probabilistic safe online learning with control barrier functions. *arXiv preprint arXiv:2208.10733*, 2022.

[33] H. Yin, P. Seiler, M. Jin, and M. Arcak. Imitation learning with stability and safety guarantees. *IEEE Control Systems Letters*, 6:409–414, 2021.

[34] J. Chen, B. Yuan, and M. Tomizuka. Deep imitation learning for autonomous driving in generic urban scenarios with enhanced safety. In *2019 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pages 2884–2890. IEEE, 2019.

[35] W. Xiao, T.-H. Wang, C. Gan, and D. Rus. Safediffuser: Safe planning with diffusion probabilistic models. *arXiv preprint arXiv:2306.00148*, 2023.

[36] J.-B. Bouvier, K. Nagpal, and N. Mehr. POLICEd RL: Learning Closed-Loop Robot Control Policies with Provable Satisfaction of Hard Constraints. In *Proceedings of Robotics: Science and Systems*, Delft, Netherlands, 7 2024. doi:10.15607/RSS.2024.XX.104.

[37] J. Thumm and M. Althoff. Provably safe deep reinforcement learning for robotic manipulation in human environments. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 6344–6350. IEEE, 2022.

[38] H. Krasowski, X. Wang, and M. Althoff. Safe reinforcement learning for autonomous lane changing using set-based prediction. In *2020 IEEE 23rd international conference on Intelligent Transportation Systems (ITSC)*, pages 1–7. IEEE, 2020.

[39] M. Selim, A. Alanwar, S. Kousik, G. Gao, M. Pavone, and K. H. Johansson. Safe reinforcement learning using black-box reachability analysis. *IEEE Robotics and Automation Letters*, 7(4): 10665–10672, 2022.

[40] Y. S. Shao, C. Chen, S. Kousik, and R. Vasudevan. Reachability-based trajectory safeguard (rts): A safe and fast reinforcement learning safety layer for continuous control. *IEEE Robotics and Automation Letters*, 6(2):3663–3670, 2021.

[41] S. M. LaValle. *Planning algorithms*. Cambridge university press, 2006.

[42] S. Redon, M. C. Lin, D. Manocha, and Y. J. Kim. Fast Continuous Collision Detection for Articulated Models. *Journal of Computing and Information Science in Engineering*, 5(2): 126–137, 02 2005. ISSN 1530-9827. doi:10.1115/1.1884133. URL https://doi.org/10.1115/1.1884133.

[43] J. Michaux, A. Li, Q. Chen, C. Chen, and R. Vasudevan. Safe Planning for Articulated Robots Using Reachability-based Obstacle Avoidance With Spheres. In *Proceedings of Robotics: Science and Systems*, Delft, Netherlands, 7 2024. doi:10.15607/RSS.2024.XX.035.

[44] S. R. Schepp, J. Thumm, S. B. Liu, and M. Althoff. Sara: A tool for safe human-robot coexistence and collaboration through reachability analysis. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 4312–4317. IEEE, 2022.

[45] A. Pereira and M. Althoff. Calculating human reachable occupancy for guaranteed collision-free planning. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4473–4480. IEEE, 2017.

[46] J. B. Michaux, Y. S. Kwon, Q. Chen, and R. Vasudevan. Reachability-based Trajectory Design with Neural Implicit Safety Constraints. In *Proceedings of Robotics: Science and Systems*, Daegu, Republic of Korea, 7 2023. doi:10.15607/RSS.2023.XIX.062.

[47] H.-T. L. Chiang, J. E. Baxter, S. Sugaya, M. R. Yousefi, A. Faust, and L. Tapia. Fast deep swept volume estimator. *The International Journal of Robotics Research*, 40(10-11):1068–1086, 2021.

[48] A. Singletary, W. Guffey, T. G. Molnar, R. Sinnet, and A. D. Ames. Safety-critical manipulation for collision-free food preparation. *IEEE Robotics and Automation Letters*, 7(4):10954–10961, 2022.

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

[50] N. Kochdumper and M. Althoff. Sparse polynomial zonotopes: A novel set representation for reachability analysis. *IEEE Transactions on Automatic Control*, 66(9):4043–4058, 2020.

[51] S. Kousik, S. Vaskov, M. Johnson-Roberson, and R. Vasudevan. Safe trajectory synthesis for autonomous driving in unforeseen environments. In *Dynamic systems and control conference*, volume 58271, page V001T44A005. American Society of Mechanical Engineers, 2017.

[52] P. Dhariwal and A. Nichol. Diffusion models beat gans on image synthesis. *Advances in neural information processing systems*, 34:8780–8794, 2021.

[53] S. Kousik, P. Holmes, and R. Vasudevan. Safe, aggressive quadrotor flight via reachability-based trajectory design. In *Dynamic Systems and Control Conference*, volume 59162, page V003T19A010. American Society of Mechanical Engineers, 2019.

[54] J. Fu, A. Kumar, O. Nachum, G. Tucker, and S. Levine. D4rl: Datasets for deep data-driven reinforcement learning. *arXiv preprint arXiv:2004.07219*, 2020.

[55] A. Mandlekar, D. Xu, J. Wong, S. Nasiriany, C. Wang, R. Kulkarni, L. Fei-Fei, S. Savarese, Y. Zhu, and R. Martín-Martín. What matters in learning from offline human demonstrations for robot manipulation. In *arXiv preprint arXiv:2108.03298*, 2021.

[56] J. Wang and E. Olson. Apriltag 2: Efficient and robust fiducial detection. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4193–4198, 2016. doi:10.1109/IROS.2016.7759617.

[57] P. Wu, Y. Shentu, Z. Yi, X. Lin, and P. Abbeel. Gello: A general, low-cost, and intuitive teleoperation framework for robot manipulators. *arXiv preprint arXiv:2309.13037*, 2023.

[58] J. Song, C. Meng, and S. Ermon. Denoising diffusion implicit models. *arXiv preprint arXiv:2010.02502*, 2020.

A Notations and Proofs

A.1 Set Operations

For $p_1, p_2 \in \mathbb{R}^n$, the line segment between them is

$$\overline{p_1 p_2} = \left\{ p \in \mathbb{R}^n \mid p = \alpha p_1 + (1-\alpha) p_2,\ \alpha \in [0,1] \right\}.$$

Let $\mathcal{X}, \mathcal{Y} \subset \mathbb{R}^n$. We use the Minkowski sum, Cartesian product, set product, and cross product (for $n = 3$):

$$\begin{aligned}
\mathcal{X} \oplus \mathcal{Y} &= \{x + y \mid x \in \mathcal{X}, y \in \mathcal{Y}\}, \\
\mathcal{X} \times \mathcal{Y} &= \{(x,y) \in \mathbb{R}^{n+m} \mid x \in \mathcal{X}, y \in \mathcal{Y}\}, \\
\mathcal{X}\mathcal{Y} &= \left\{ XY \in \mathbb{R}^{n \times k} \mid X \in \mathcal{X}, Y \in \mathcal{Y} \right\}, \text{ and} \\
\mathcal{X}^{\times}\mathcal{Y} &= \left\{ \begin{bmatrix} 0 & -x_3 & x_2 \\ x_3 & 0 & -x_1 \\ -x_2 & x_1 & 0 \end{bmatrix} y \in \mathbb{R}^3 \mid \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \in \mathcal{X},\ y \in \mathcal{Y} \right\}.
\end{aligned}$$

For a function f, we denote $f(\mathcal{X}) = \{f(x) \mid x \in \mathcal{X}\}$. We apply linear operations on the left or right as appropriate (e.g., $A\mathcal{X}$ or $\mathcal{X}A$). We numerically implement all set operations with polynomial zonotopes [50] as per [7, §III-C].

A.2 Proof for Lemma 2

Proof. Notice that p_3 is halfway along the line segment between p_1 and p_2. Similarly, p_4 is the projection of p_3 onto the unit circle CCW between p_1 and p_2. Thus, by construction and CCW motion, $\mathscr{Z}(c,G)$ is a rectangle containing all four points and the arc from p_1 to p_2, as in Fig. 2. □

B Summary of RAIL Algorithm

Algorithm 1: Reachability Aided IL (RAIL)

Input: Observation o_0, IL policy π, Backup Policy π_{back}

1 // initialize plan and backup plan
2 $(t, x_0) \leftarrow (0,$ **stateEstimate**$(o_0))$
3 $\hat{a}_{0:T_b} \leftarrow \pi_{\text{back}}(x_0)$
4 $(a_{0:T_p}, {}^{\text{back}}a) \leftarrow (\hat{a}_{0:T_p}, \hat{a}_{T_a:T_b})$
5 **while** *task is not done* **do**
6 Apply $a_{t:t+T_a}$ to the robot for horizon of t to $t+T_a$
7 // execute lines below while applying actions
8 $o_{t+T_a} \leftarrow$ **predictObservation**$(o_t, a_{t:t+T_a})$
9 $x_{t+T_a} \leftarrow$ **stateEstimate**(o_{t+T_a})
10 $a_{t+T_a:t+T_a+T_p} \leftarrow \pi(o_{t+T_a})$
11 $(a_{t+T_a:t+2T_a}, {}^{\text{back}}a) \leftarrow \text{Filter}(x_{t+T_a}, a_{t+T_a:t+T_a+T_p}, {}^{\text{back}}a)$
12 **if** ${}^{\text{back}}a$ *is empty* **then**
13 // robot stopped safely
14 Set up a backup plan as Line 2-4

Algorithm 2: Filter

Input: State x_t, Nominal plan $a_{t:t+T_p}$ Backup plan ${}^{\text{back}}a$

Output: Filtered plan $a_{t:t+T_a}$, Updated Backup plan ${}^{\text{back}}a$

1 $x_{t:t+T_p} \leftarrow$ **predictState**$(x_t, a_{t:t+T_p})$
2 **if** $\text{FO}(x_{t:t+T_a}) \cap \mathcal{W}_{\text{obs}} = \emptyset$ **then**
3 $\hat{a}_{t+T_a:t+T_a+T_b} \leftarrow \pi_{\text{back}}(x_{t+T_a})$
4 **if** $\hat{a}_{t+T_a:t+T_a+T_b}$ *is feasible* **then**
5 **return** $a_{t:t+T_a}$, $\hat{a}_{t+T_a:t+T_a+T_b}$
6 **else**
7 **return** ${}^{\text{back}}a_{t:t+T_a}, {}^{\text{back}}a_{t+T_a:\text{end}}$

C Implementation Details

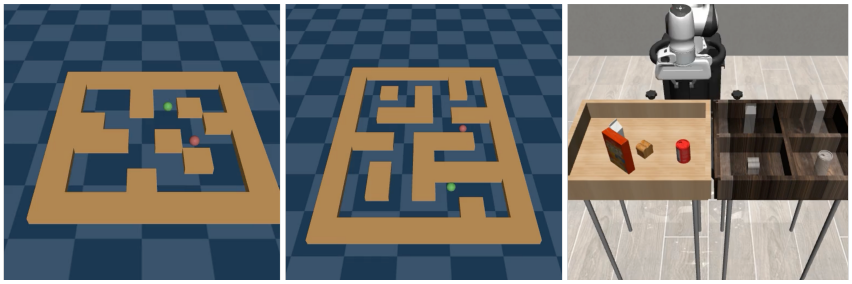

Figure 4: Evaluation tasks (left-to-right): **(a) Maze-Medium:** A point mass has to be taken to a goal in a maze without hitting the walls. **(b) Maze-Large:** Similar to Maze-Medium, but larger and more difficult to traverse. **(c) Can Pick-Place:** A can must be picked from a table and placed in a target bin.

C.1 Algorithm: Continuous-Time Collision Checking

In this section, we provide the implementation details of the continuous-time collision checking algorithm, first introduced in Section 5.

Per Cor. 5, we can overapproximate the forward occupancy of the robot over an entire trajectory by considering a union over all links and all time intervals. Let $\widetilde{\mathrm{FO}}(q([0,t_\mathrm{f}]))$ denote the right-hand side of (4), so at runtime we check $\widetilde{\mathrm{FO}}(q([0,t_\mathrm{f}])) \cap \mathcal{W}_\mathrm{obs} = \emptyset$. We implement this with polynomial zonotopes [50]. All functions used in the approximation are analytic and thus overbounded per [50] (see [7, §III-C.5]), and polynomial zonotopes are closed under intersections. We represent the unions in (4) by storing a polynomial zonotope for each time interval and arm link separately.

C.2 Algorithm: Safety-guided Diffusion Policy

We implement a Safety Guided Diffusion Policy as follows: (a) annotating the offline dataset with safe and unsafe labels based on whether the resulting state is safe after executing the action or not (b) training a classifier network to predict safety probability (c) following classifier-guided reverse sampling [52] with pre-trained diffusion policy network.

C.3 Experiments Setup: Safe Maze Navigation

As our baseline IL policy, we train a Diffusion Policy [3] on the offline D4RL dataset [54] with an action trajectory prediction horizon of $T_p = 32$; at test time we execute a horizon of $T_a = 16$. The observation space is the full state of the robot (i.e., robot position and velocity). The action space is a 2D force input applied to the point-mass robot where the low-level controller directly outputs the action. Additionally, we implement a diffusion policy guided by a safety probability prediction classifier, following the method described in Appendix C.2.

To implement the backup planner, we adapt Reachability-based Trajectory Design (RTD) from [53], where the desired trajectory defines the target position of the robot. A Proportional-Integral-Derivative (PID) controller is then used to track this trajectory.

We access the predictions of state and observations by rolling out the model.

C.4 Experiments Setup: Safe Pick-and-Place

As our baseline IL policy, we train Diffusion Policy [3] and ACT [8] on 184 safe, successful demonstrations collected via a robomimic [55]. For all policies, we sample action sequences with a horizon of $T_a = (8, 16)$, $T_p = 32$. For diffusion policy, we use a DDIM [58] scheduler with 100 reverse denoising timesteps for sampling. The observation space is end effector pose, gripper position, and

objects' poses. We choose the delta end effector pose as an action abstraction, where an Operational Space Controller [49] is employed as a low-level controller.

To implement the backup planner, we adapt ARMTD [6], where the desired trajectory defines the target join position of the robot, and a joint position controller is used to track the trajectory.

Similar to the Maze example, we predict states and observations by rolling out the model.

C.5 Real World Evaluation Setup

As our baseline IL policy, we train Diffusion Policy on 75 safe, successful demonstrations collected via a gello [57]. For all policies, we sample action sequences with a horizon of $T_a = 16$, $T_p = 32$. The observation space is arm's joint angles, gripper position, and objects' poses. We choose the desired joint angle as action representation, and joint impedance controller as low-level controller.

ARMTD [6] serves as the backup planner as in Appendix C.4.

To predict states and observations, we directly use the desired joint angles from the policy with an empirically determined latency horizon to minimize the prediction error.

D Additional Experiments

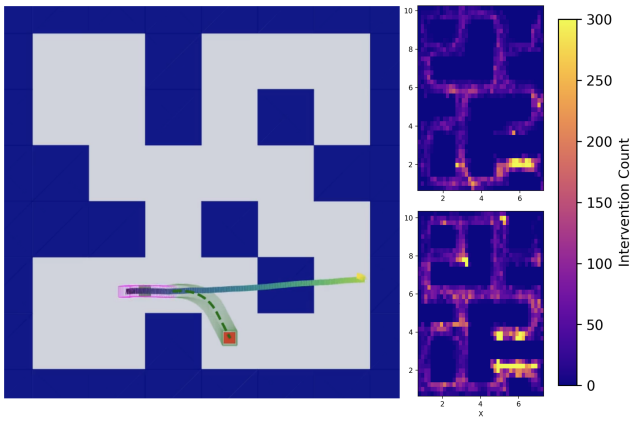

Figure 5: **(Left)** Visual representation of RAIL in the Maze-Medium task. The diffusion policy's proposed plan (gradient line) is validated by checking the reachable set of head plans (magenta) and the existence of a backup plan (green dotted line), with its reachable set (green tube). **(Right)** Heatmap comparing the number of safety interventions by RAIL in the Maze-Large task over 100 episodes with random initializations, comparing diffusion policies at Epoch 1900 **(Top)** and Epoch 50 **(Bottom)**.

We further analyze how IL policy performance affects integration with RAIL by evaluating trained models across different epochs. The number of safety interventions notably decreases in later epochs *despite no explicit safety training signal* (see Figure 5).