SAFE-GIL: SAFEty Guided Imitation Learning for Robotic Systems

Author Names Omitted for Anonymous Review. Paper-ID [add your ID here]

***Abstract*—Behavior cloning (BC) is a widely used approach in imitation learning, where a robot learns a control policy by observing an expert supervisor. However errors in the learned policy can lead to safety violations, especially in safety-critical settings. While prior works have tried improving a BC policy via additional real or synthetic action labels, adversarial training, or runtime filtering, none of them explicitly focus on reducing the BC policy's safety violations during training time. We propose SAFE-GIL, a design-time method that improves safety in BC by injecting adversarial disturbances during data collection. These disturbances guide the expert into safety-critical states, better preparing the policy for similar situations at test time. We use a reachability-based formulation to compute these disturbances. Evaluations in ground navigation, aircraft taxiing, and quadrotor flight show that SAFE-GIL significantly reduces safety failures, especially in low-data regimes.**

I. Introduction

Imitation learning enables robots to learn complex behaviors without manually specified objective functions. A common approach is Behavior Cloning (BC), where policies are trained to imitate expert actions via supervised learning [1], [2]. BC has been widely applied in manipulation [3], navigation [4], and autonomous driving [5], but learned policies often make mistakes due to distribution shifts or limited data, leading to safety violations during deployment.

Existing methods for improving safety in BC can be grouped into deployment-time (e.g., safety filters [6], [7]) and design-time approaches. Safety filters intervene at runtime using tools such as control barrier functions (CBFs) [8] or reachability analysis [9], but can be hard to construct, rely on sufficient environment information, and may utilize an underlying policy that is unsafe. Thus, designing BC policies that proactively account for safety becomes a desirable alternative.

Design-time methods improve BC by modifying the expert policy [10] or augmenting the dataset to reduce covariate shift [11], [12]. Some approaches simulate errors through adversarial training [13], [14] or define safety in terms of expert deviation [15]. However, few explicitly address safety constraint satisfaction (e.g. collision avoidance) during training. To overcome these challenges, we propose SAFE-GIL (SAFEty Guided Imitation Learning) – a novel *design-time* algorithm that injects adversarial disturbances into the system during data collection. These disturbances simulate worst-case policy errors and guide the expert into safety-critical states, enabling the policy to learn corrective actions. We compute disturbances using offline reachability analysis, ensuring safety-critical scenarios are covered during training.

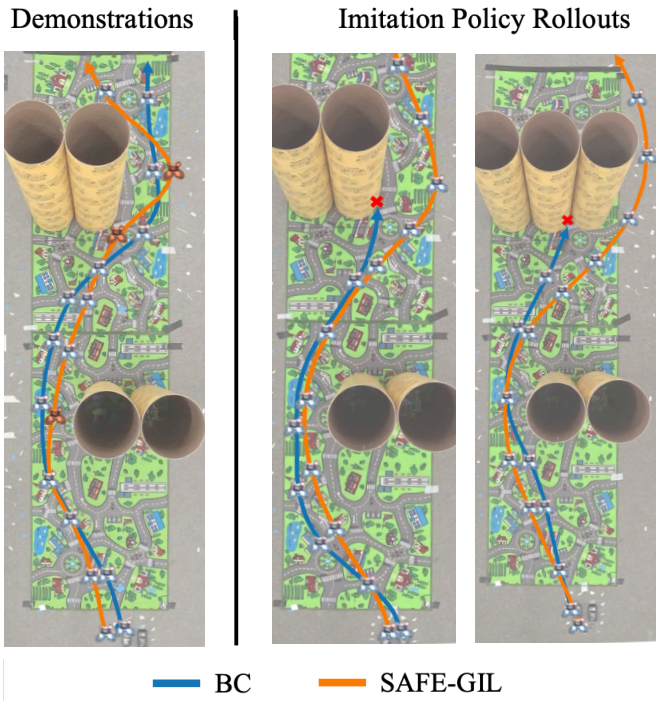

Fig. 1: Left: Human controlled quadrotor demonstration trajectories with (SAFE-GIL) and without adversarial safety guidance. With SAFE-GIL, the robot observes more safety-critical states during training time (illustrated by red quadrotor icons). Right: BC and SAFE-GIL policy rollouts. The red cross denotes a collision. SAFE-GIL results in a significant improvement in robot safety during the test time.

SAFE-GIL produces safer policies by exposing the learner to potential failure modes during training—complementing and enhancing both BC and online safety mechanisms.

To summarize, our key contributions are: (a) we propose a new algorithm, SAFE-GIL, to learn safety-aware imitation policies to reduce test-time safety failures; (b) through simulation studies and hardware experiments on a quadrotor testbed, we demonstrate that SAFE-GIL results in significantly safer imitation policies, while maintaining system's performance; (c) we demonstrate that SAFE-GIL can be seamlessly integrated with existing imitation learning approaches, such as DAgger, to reduce covariate shift while mitigating safety-critical failures. Additionally, it can be combined with online safety mechanisms, such as safety filtering, to further reduce test-time failures.

II. Problem Formulation

Consider a robot with state $x \in X \subseteq \mathbb{R}^n$ and control $u \in \mathcal{U}$. We assume access to an approximate dynamics model of the

robot $\dot{x} = f(x,u)$ during training. Let $\pi^*(\cdot)$ denote an expert policy that we want to replicate via an imitation policy $\pi_\theta(\cdot)$, where $\theta \in \Theta$ is to be learned. Finally, let $L \subseteq \mathbb{R}^n$ be the set of states that represent failure for the robot. For example, L might represent obstacles for a navigation robot or off-runway positions for an autonomous aircraft.

Objective. Our goal in this work is to learn an imitation policy that minimizes safety violations (i.e., stay outside L) during the test time, while minimally sacrificing the policy performance.

III. Background: Hamilton-Jacobi Reachability

We now provide a brief overview of Hamilton-Jacobi (HJ) reachability analysis which we will use to compute adversarial disturbances to guide the expert towards safety-critical states. HJ reachability analysis is a formal verification method for characterizing the safety-critical regions of a dynamical system [16], [17]. For reachability analysis, we will consider a more general form of dynamics: $\dot{x} = f(x,u,d)$, where $d \in [-\bar{d},\bar{d}]$ represents the disturbance or uncertainty in the dynamics. Later on, in our work, we will use d to model potential policy errors. Given the dynamics f and a failure set L, we are interested in computing the backward reachable tube (BRT) of the system – the set of all states from which the system will eventually enter the failure set regardless of the control policy $u(\cdot)$, given the system is subject to optimum disturbance policy $d(\cdot)$ [16].

To compute the BRT, we represent the failure set L implicitly with a target function $l(x)$, i.e., $L = \{x : l(x) \leq 0\}$. Next, we define the cost function as the minimum value of the target function along the robot trajectory, starting from a state x_0 at time t:

$$J(x_0,t,u(\cdot),d(\cdot)) = \min_{\tau\in[t,T]} l(x(\tau)). \tag{1}$$

Thus, the sign of the cost function tells us whether a robot trajectory ever entered the failure set. Finally, the optimal $u(\cdot)$ drives the system away from the unsafe states (i.e., it maximizes the above cost function) and the optimal $d(\cdot)$ drives the system towards L (i.e., minimizes the cost function). The robust value function can be obtained using the principle of dynamic programming, which results in solving the following final value Hamilton-Jacobi-Isaacs Variational Inequality (HJI-VI).

$$\min\{D_t V(x,t) + H(x,t), l(x) - V(x,t)\} = 0; \quad V(x,T) = l(x), \tag{2}$$

where D_t and ∇ represent the time and spatial gradients of the value function and H is the Hamiltonian given as:

$$H(x,t) = \max_u \min_d \langle \nabla V(x,t), f(x,u,d)\rangle. \tag{3}$$

For a detailed derivation and discussion of the HJI-VI in (2), we refer the interested readers to [17] and [16]. Intuitively, the value function at x represents the closest the robot will ever get to the failure set under the optimal safety policy starting from state x. Thus, a lower value denotes a more safety-critical state, and a value below 0 denotes an unrecoverable state, i.e., the state is in the BRT.

The value function typically converges after some time horizon, beyond which the system has enough time to maneuver to avoid the failure set despite the worst case disturbance. Consequently, we can use the converged value function, $V^*(x) = \lim_{t\to\infty} V(x,t)$. Given V^*, the BRT is given as the sub-zero level set of the value function: $\mathcal{V} = \{x : V^*(x) \leq 0\}$. Importantly, the value function can be used to synthesize the optimal disturbance that at any state x maximally steers the robot toward unsafe states:

$$d^*(x) = \max_u \arg\min_d \langle \nabla V^*(x), f(x,u,d)\rangle, \tag{4}$$

i.e., towards lower values. We will use d^* to guide the expert towards safety-critical states.

HJ Reachability Computation. Several methods have been proposed to solve the HJI-VI in (2) and compute the value function (see [16] for a survey). One method is to numerically solve the HJI-VI over a discretized grid in the state space [18], [19], which is what we use in this work to compute the value function for the simulation case studies. Other learning-based methods also exist that are more scalable to high-dimensional systems, such as DeepReach [20], which we use for the quadrotor hardware experiment.

IV. SAFE-GIL: SAFEty Guided Imitation Learning

We aim to learn safe imitation policies by intentionally guiding the expert toward safety-critical states during data collection. Our key idea is to abstract potential test-time policy errors as an *adversarial disturbance* that attempts to steer the agent towards unsafe states over time, and inject this disturbance during data collection. Specifically, the expert is guided towards safety-critical states by the application of a perturbed input, $\pi^G(x)$, to the system, instead of $\pi^*(x)$:

$$\pi^G(x) = \pi^*(x) + d(x), \tag{5}$$

where $d(x)$ is the injected disturbance. Intuitively, $d(x)$ can be thought of as simulating potential learning error at state x during the test time. However, since the learning error is not known beforehand, we inject $d(x)$ so as to maximally steer the system towards the failure set. Overall, by guiding the expert with π^G, the system is likely to visit safety-critical states more often. The corrective examples from these states will allow the robot to learn to recover from them, should they be encountered during the test time, enhancing the overall system safety.

Computation of $d(x)$. In this work, we use HJ reachability analysis to compute $d(x)$. Specifically, given the disturbance injection scheme in (5), the robot state evolution can approximately be described as $\dot{x} = f(x,u+d) := g(x,u,d)$, where $d \in [-\bar{d},\bar{d}]$. $\bar{d}$ is a hyperparameter that can be used to control the amount of injected disturbance. Given the disturbance-injected dynamics, g, and the failure set L, we compute the corresponding safety value function $V^*(x)$ by solving the HJI-VI in (2). To compute the value function for different disturbance bounds, we further condition the value function on $\bar{d}$ to obtain a parameter-conditioned value function $V^*(x;\bar{d})$.

Algorithm 1: Safety Guided Data Collection

Input: Number of demonstrations K; Trajectory length T;
Output: Guided state trajectory and corresponding expert actions $(x, \pi^*(x))$;
for i=1:K **do**
  Sample initial state x_1
  for j=1:T **do**
    $\bar{d} \sim U(0, \bar{d}_{max})$
    Get optimal disturbance $d^*(x_i; \bar{d})$ using (4)
    $\pi^G(x_i) \leftarrow \pi^*(x_i) + d^*(x_i; \bar{d})$,
    Store $(x_i, \pi^*(x_i))$ pair
    Apply $\pi^G(x_i)$ on the system; obtain x_{i+1}
  end for
end for
return State and expert action pairs

The parameter-conditioned value function can be obtained by simply adding a dummy state $\bar{d}$ to the system with zero dynamics [21]. Given the value function, the optimal adversarial disturbance $d^*(x; \bar{d})$ is obtained using (4), which steers the system towards the smallest V^* at any state, i.e., towards more safety-critical states.

Finally, at each time step during data collection, the upper bound on disturbance, $\bar{d}$, is uniformly sampled between $[0, \bar{d}_{max}]$, i.e., $\bar{d} \sim U(0, \bar{d}_{max})$. Correspondingly, we obtain the optimal adversarial disturbance $d^*(x; \bar{d})$. Scaling of the disturbance bound helps in various ways: first, injecting disturbance with the maximum bound at each time step can steer the system too close to the failure set during data collection, which is not desirable for safety-critical systems. Second, since we do not know the policy error distribution beforehand, randomizing the disturbance magnitude results in more diverse demonstrations without being overly conservative. Our overall data collection procedure is described in Algorithm 1. To collect a demonstration trajectory, we randomly sample an initial state. Next, at each time step along the trajectory, we uniformly sample a $\bar{d}$ and compute the optimal disturbance using $V^*(x; \bar{d})$ and (4). The disturbance is then injected into the expert action and applied to the system, and the entire process is repeated until the end of the trajectory. Finally, we train a behavior cloning policy $\pi_\theta(x)$ on the collected data.

V. Experiments

We demonstrate the robustness of the learned policy under SAFE-GIL on two simulation case studies (state-based autonomous navigation and camera-based aircraft taxiing) and on a hardware testbed (aerial navigation). Each study varies in dynamics, observation space, and compute resources, with the intention of demonstrating safety enhancement.

A. Autonomous Navigation Using a State-Based Policy

We consider a wheeled robot navigating in a 2D space to reach a goal position without colliding with obstacles in the environment. The navigation task is to be performed autonomously during the test time, starting from various initial states. The robot state is given by $x := (p_x, p_y, \theta)$, where (p_x, p_y) is the robot position and θ is its heading. The control input is angular velocity $u := \omega$, bounded by $|\omega| \leq \bar{\omega}$. We model the robot as a unicycle with dynamics: $\dot{p}_x = v\cos(\theta), \quad \dot{p}_y = v\sin(\theta), \quad \dot{\theta} = \omega$. In our example, we use $\bar{\omega} = 1\,rad/s$ and $v = 1\,m/s$.

The failure set L is given by the gray obstacles (Fig. 2) that the robot must avoid to reach the goal (green area). The expert is a Model Predictive Control (MPC)-based controller that minimizes a cost function penalizing the robot's distance to the goal, obstacle penetration, and control energy. The imitation agent takes as input the current robot state (p_x, p_y, θ) and outputs the angular speed, and is modeled as an MLP. Each imitator is trained for 10 different seeds to capture the performance variance.

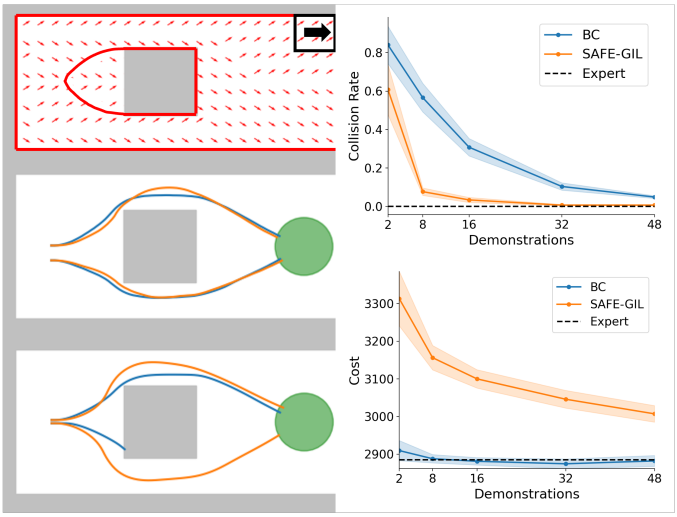

Fig. 2: Top row: Computed BRT and disturbance for $\theta = 0$. Middle row: Demonstration trajectories with (Orange) and without (Blue) disturbance injection. Bottom row: BC and SAFE-GIL policy rollouts. Right column (Top): Mean collision rate and (Bottom) cost of safe trajectories vs number of demonstrations. SAFE-GIL results in a significant safety improvement.

For SAFE-GIL, we use $\bar{d}_{max} = 0.6\ \bar{\omega}$, which corresponds to a maximum of 60% policy error in imitating the agent. The value function is computed over a $101 \times 101 \times 101$ grid using the Level Set Toolbox [22] by solving the HJI-VI numerically (converging within $\approx 17s$). The converged value function is then used to compute d^* for guidance. In Fig. 2 (top), the red region indicates the BRT – if the robot enters this set, it doesn't have enough control authority (angular velocity) to avoid a collision with the obstacle. The red arrows indicate the direction of d^* at each state, which pushes the robot towards the obstacles.

Safety. We compare SAFE-GIL against vanilla Behavior Cloning (BC) for different number of expert demonstrations. Fig. 2 compares the collision rates (percentage of trajectories that collide before reaching the goal location) of the resulting imitation policies by rolling out from 100 randomly sampled initial states. SAFE-GIL is able to achieve substantially lower collision rates compared to BC, especially in low data regimes

where policy errors are more likely and the safety problem is more pronounced. To understand this behavior, we illustrate the expert demonstrations for BC and SAFE-GIL (middle row), as well as the corresponding rollouts of the imitation policies (bottom row) in Fig. 2. Due to disturbance injection, the expert under SAFE-GIL is guided closer to the obstacles; consequently, expert recovery maneuvers from these states are present in the demonstration data, allowing the learned imitation policy to better avoid the obstacles. On the other hand, without the proposed disturbance guidance, some of these safety-critical states are never encountered by the expert, even as the number of collected expert trajectories increases, resulting in a slower safety improvement.

Performance tradeoff. Under SAFE-GIL, the training distribution shifts towards more safety-critical states. This might result in having fewer samples from the states that maximize the expert's reward function, ultimately leading to performance degradation compared to BC. To distinguish between safety and task performance, we plot the mean cost accrued per safe trajectory of policy rollouts in Fig. 2, using the cost function expert is minimizing for. Note that although SAFE-GIL results in more number of safe trajectories compared to BC, the performance of the safe trajectories is lower.

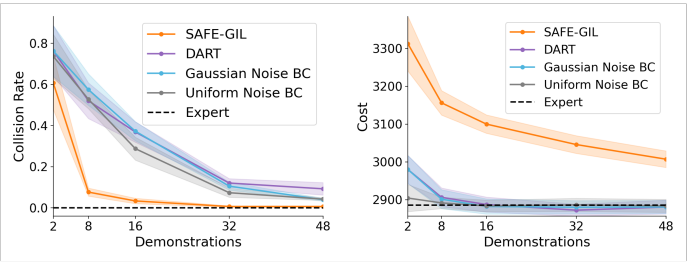

Fig. 3: Mean collision rate (Left) and cost of safe rollouts (Right) vs number of demonstrations. Adversarial noise injection leads to a significant safety improvement over random noise.

Effect of adversarial disturbance. SAFE-GIL can also be thought of as a data augmentation scheme that adds noise to the expert data. To understand the importance of using *adversarial* noise during data augmentation, we compare against other schemes that inject Gaussian noise (**Gaussian Noise BC**), uniform random noise (**Uniform Noise BC**), and estimate the noise covariance online (**DART** [23]). Results in Fig. 3 show that alternative variants fail to improve the system safety, highlighting the importance of accounting for the safety criticality of a state during data augmentation. By using HJ reachability, SAFE-GIL is *targetedly* guiding the agent towards safety-critical states, resulting in safer policies.

Effect of disturbance bound. The guidance and learning of the safety-critical behavior in SAFE-GIL is dependent on the choice of the disturbance bound. Thus, we perform an ablation study to understand the effect of the disturbance bound on the safety of the resultant policy (Table I). As the disturbance bound increases, SAFE-GIL is able to expose the demonstrator to higher potential learning errors (modeled as disturbance) and guide it towards more unsafe states. Thus, a higher disturbance bound results in a more robust imitation

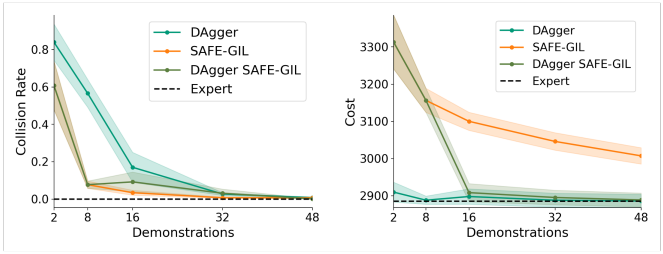

Fig. 4: SAFE-GIL can be combined with other imitation learning approaches to have complementary safety and performance advantages.

policy. However, increasing the disturbance bound beyond a particular point leads the system to states that are too risky, which the expert might fail to recover from. Moreover, as the training data shifts towards more unsafe states, the task performance of the learned policy decreases.

$\bar{d}_{max}$	0.2	0.4	0.6
Collision Rate	0.19 ± 0.02	0.11 ± 0.05	0.07 ± 0.01
Cost	2975 ± 2	3089 ± 3	3156 ± 3

TABLE I: Mean collision rates and cost of safe trajectories of agents trained with SAFE-GIL for different disturbance bounds.

Mitigating Covariate Shift. We demonstrate that SAFE-GIL can easily be integrated with existing approaches to reduce covariate shift and performance degradation, while mitigating safety-critical failures. As an example, we consider **DAgger** [24] that mitigates covariate shift by iteratively aggregating expert actions on the states visited by the learned policy. We combine DAgger with SAFE-GIL by injecting adversarial disturbance during expert rollouts. We see significant improvement in safety compared to DAgger alone and in performance compared to SAFE-GIL alone (Fig. 4).

B. Autonomous Aircraft Taxiing Using a Vision-Based Policy

We now consider an aircraft taxiing task, used as a benchmark for robust perception and verification [25]–[27]. The agent is an aircraft simulated in an X-Plane flight simulator. The system dynamics are modeled as: $\dot{p}_x = v\sin(\theta)$, $\dot{p}_y = v\cos(\theta)$, $\dot{\theta} = (v/h)\tan(\omega)$, where (p_x, p_y, θ) denote the crosstrack error (CTE), the downtrack position (DTP), and the the heading error (HE) with respect to the runway centerline, respectively. The aircraft controls its steering ω, bounded by $|\omega| \leq \bar{\omega}$. In our example, we use $\bar{\omega} = 1\,rad/s$, $v = 5\,m/s$, and $h = 5\,m$. The task is to reach the end of the runway ($p_y = 200$) without leaving the runway, which corresponds to $L = \{x : |p_x| \geq 10\}$. The expert is a PID controller designed to keep the aircraft on the runway and steer towards the centerline. In this case, the expert has privileged access to the system state during the data collection procedure, which is *not* available to the imitator during test time. Instead, it needs to imitate the expert based on the RGB image observations obtained through a camera mounted on the plane's right wing.

We compute the value function and optimal disturbance using the Level Set Toolbox [22]. We train each imitator for 5 different seeds to capture the performance variance and compare their *excursion rates* (the percentage of trajectories that go

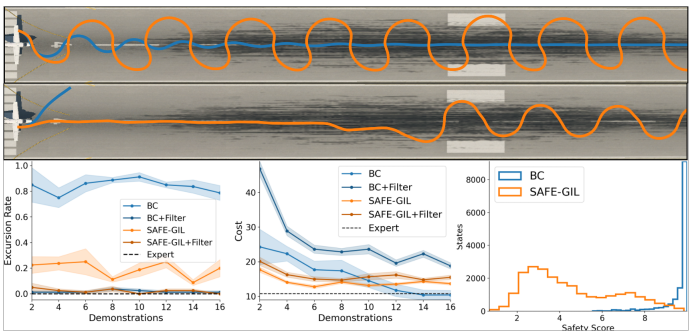

Fig. 5: Top: Expert demonstration with and without disturbance injection. Middle: BC and SAFE-GIL rollouts from the same initial state. BC fails to keep the aircraft on the runway. Bottom: Mean excursion rate (Left) and Mean squared distance from the centerline (Middle) vs number of demonstrations. Safety value distribution of the collected demonstrations (Right) is shifted towards lower values for SAFE-GIL.

out of the runway) starting from 16 randomly sampled initial states. We use mean squared distance from the centerline of the safe trajectories as the performance metric. As evident from Fig. 5, SAFE-GIL achieves a substantially lower excursion rate compared to BC. This highlights the effectiveness of the safety guidance for learning robust imitation policies. During demonstrations, the disturbance guides the expert towards the runway boundaries. In turn, the agent is able to recover from policy errors that may steer the system towards the runway boundaries, whereas BC fails to do so. This is also evident from the shift towards lower regimes in the safety value ($V^*(x)$) of the dataset collected with SAFE-GIL (Fig 5). Overall, this case study demonstrates that even though SAFE-GIL relies on state-based guidance during data collection, it does not require any such privileged information during the test time and can readily be applied to observation-based policies.

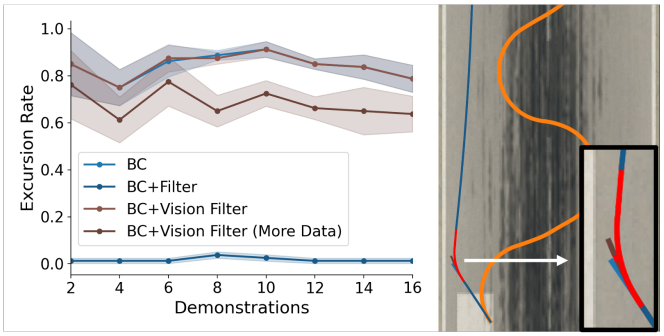

Fig. 6: Left: Mean excursion rates vs number of demonstrations. Right: SAFE-GIL, BC+Filter, BC+Vision Filter (More Data) trajectories from the same initial state. States where safety filter engages are denoted with red.

Safety Filtering. We now compare SAFE-GIL against a safety filter-based approach, a popular mechanism to ensure safety during test time [28]. We implement a least restrictive safety filter that overrides the imitation policy with a safe controller whenever the system approaches the BRT boundary [29]. One of the key challenges in this example is the computation of the BRT for filtering since the state information is not available during the test time. Thus, we implement three different filters: for **BC+Filter**, privileged state information is used during test time. For **BC+Vision Filter** we train a state estimator from the images captured from states visited during expert demonstrations. For **BC+Vision Filter (More Data)** we sampled 10000 (image, state) pairs uniformly over the state space to train the state estimator. During testing, a least restrictive filter is applied based on the estimated state of the agent.

The results are shown in Fig. 6. While BC+Filter is able to maintain system safety, the overall performance of the system is still limited by the underlying policy. This can be seen in the trajectory shown in Fig. 6. Since the underlying policy doesn't have a good notion of recovery behavior, even though the filter keeps the agent on the runway, the agent isn't able to return to the centerline, whereas SAFE-GIL recovers much earlier and returns to the centerline. This highlights the need for *design-time methods* that proactively reason about system safety during the training time itself. Moreover, the performance of the safety filter degrades as we move away from the perfect state estimation assumption. BC+Vision Filter (More Data) with additional access to the environment data has only slightly lower excursion rates than BC, whereas the vision filter trained without any additional data performs similarly to BC. In Fig. 6 (Right), we see that the vision filter with additional data engages safety control too little and too late due to errors in state estimation, highlighting the challenges associated with safety filtering when a system operates on raw sensory data. Finally, we note that safety filtering, being a test-time method, is complementary to SAFE-GIL. For instance, **SAFE-GIL+Filter** uses a safety filter on SAFE-GIL policies. Fig. 5 shows that this further reduces the test-time failures without much degradation in task performance since the underlying policy is better able to reason about the safety of the system and recovery.

Effect of adversarial disturbance. Similar to the autonomous navigation case study in Section V-A, we compare SAFE-GIL against alternative noise injection methods (Gaussian Noise BC, Uniform Noise BC, and DART). We notice a similar pattern as before, highlighting the importance of the adversarial nature of injected disturbance for safety improvement. We omit those results due to space constraints.

$\bar{d}_{max}$	0.1	0.3	0.5
Excursion Rate	0.66 ± 0.06	0.63 ± 0.08	0.11 ± 0.02
Cost	9.52 ± 0.7	12.6 ± 0.6	14.1 ± 0.4

TABLE II: Mean excursion rates and squared distance from the centerline for the agents trained with SAFE-GIL for different disturbance bounds.

Effect of disturbance bound. As the disturbance bound increases, SAFE-GIL is able to expose the demonstrator to more unsafe states. As before, this results in better learning of the safety critical behavior at the expense of some task performance as seen in Table II.

C. Quadrotor Navigation: Hardware experiment

We now demonstrate our method on a Crazyflie 2.1 quadrotor testbed where the robot needs to reach a goal location with-

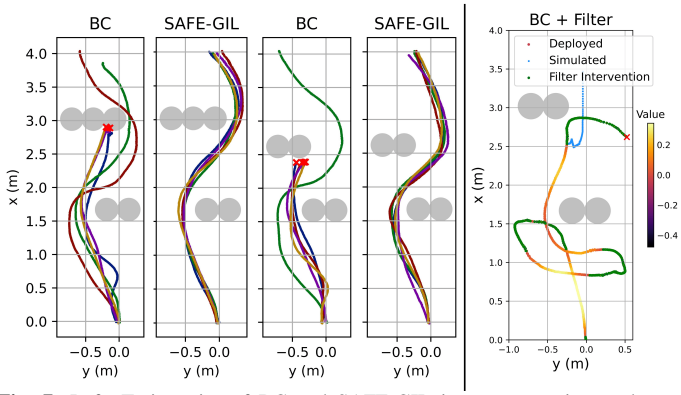

Fig. 7: Left: Trajectories of BC and SAFE-GIL in unseen settings, where X denotes a collision. SAFE-GIL leads to a significantly more robust performance compared to BC. Right: BC+Filtering trajectory.

out collisions. In this case, we conduct experiments comparing BC, BC+Filter, and SAFE-GIL. To collect the demonstrations, a *human expert* uses a game controller to fly the Crazyflie and collect the applied roll and pitch commands (which are then tracked internally by a PID controller) along with estimated states $p_x, p_y, p_z, v_x, v_y, v_z$ (xyz position and velocities, computed fully onboard using an optical flow camera), and an 8-pixel row of depth measurements for obstacles. A simplified quadrotor dynamics model is used for the disturbance computation in SAFE-GIL and safety filtering: $\dot{p}_x = v_x, \quad \dot{p}_y = v_y, \quad \dot{p}_z = v_z, \quad \dot{v}_x = a_g \tan u_\theta, \quad \dot{v}_y = -a_g \tan u_\phi, \quad \dot{v}_z = u_T - a_g$. The control inputs are the net upwards thrust u_T and desired roll u_ϕ and pitch u_θ angles. The failure set includes all the cylinder obstacles (gray circles) and the x-y position outside the grid shown in Fig. 7. The optimal disturbance is computed by solving the HJI-VI through a DeepReach [20] neural network, which is then distilled for onboard computation.

Obstacle Configuration	$(1,1)$	$(2,2)$	$(2,3)$	$(2,2_c)$
BC	20%	60%	60%	80%
SAFE-GIL	**0%**	**0%**	**0%**	**0%**

TABLE III: Collision rates for different obstacle configurations. $(2,3)$ denotes two obstacles in the first row and three obstacles in the second row. Whereas, 2_c marks that the two obstacles rows are moved closer.

We collect 5 demonstrations on 2 different obstacle settings, first with singular obstacle pairs and another with dual obstacle pairs. The trained policies are tested five times each across four obstacle settings, two of which are not observed during the training time (Table III). Trajectories for unseen obstacle settings are shown in Figure 7, where one has an additional third obstacle, and the other has the second set of obstacles brought closer to the first to make the passage between them narrower. SAFE-GIL outperforms BC across all configurations, achieving 0% collision rate. We find that the approximate dynamics model used is still useful in providing safety guidance through SAFE-GIL, highlighting that an exact model of the system may not be necessary.

Safety Filtering. We implement a least restrictive filter for BC by querying the value function (one for each obstacle setting) used during guidance. BC+Filter achieves a 0% collision rate on obstacle setting $(1,1)$ and 40% in $(2,2)$. Ideally the safety filter should keep the system safe at all times. However, the approximate dynamics model used for reachability computations assumes instantaneous direct control over the roll and pitch of the quadrotor, whereas delays between the desired and realized control during deployment result in failures. To further test this, we propagate simulated approximate dynamics with no control delay in blue at the moment the safety filter intervenes (Fig. 7 (Right)). Indeed, the simulated filter maintains safety and the underlying policy is able to complete the task, highlighting the dependence of the safety filter on an accurate dynamics model. Moreover, even when the safety filter keeps the agent safe, the performance of the filtered policy is still limited by the underlying BC policy. This is seen in Fig. 7 (Right). The filter protects the BC policy from a collision with the first set of obstacles early on. However, the BC policy isn't able to bring the agent back to the task when the safety filter disengages and the quadrotor swings from one side of the task space to the other until the safety filter engages again to keep the agent safe.

VI. Discussion and Future Work

We propose SAFE-GIL, an algorithm to learn safety-aware imitation policies. Our experiments show that SAFE-GIL results in significantly safer imitation policies, especially in low data regimes where the prediction error might be high. However, the proposed framework has a few limitations: *1.) Scalability of $d(x)$ computation*. Our approach requires a known dynamics model to solve for HJI-VI in order to compute $d(x)$. Furthermore, HJI-VI is known to scale poorly for high-dimensional systems. To overcome these challenges, we will explore learning-based reachability methods [20], [30]. These methods are shown to be scalable to complex high-dimensional systems, and to black-box dynamics models. *2.) Knowledge of the safety constraints*. The proposed approach relies on the known definitions of safety. An interesting future direction would be to learn the safety constraints from data, e.g., using inverse RL [31]. *3.) Practicality of disturbance scale tuning*. The current approach is limited by an informed selection of disturbance bounds to maintain a reasonable safety-performance tradeoff. In our experiments, we have found 20-30% disturbance compared to the control bounds results in the best safety to performance ratio. A potential extension could be to estimate the disturbance bounds online based on the policy error. *4.) Compensating for added disturbance during data collection.* The expert needs to compensate for the disturbance added during data collection, which can be challenging for high disturbance bounds. One way to overcome this challenge could be to use a safety filter *during training time* along with the proposed approach to compensate for any last-resort safety failures that the expert policy can't handle.

REFERENCES

[1] D. A. Pomerleau, "Alvinn, an autonomous land vehicle in a neural network," 2015.

[2] M. Zare, P. M. Kebria, A. Khosravi, and S. Nahavandi, "A survey of imitation learning: Algorithms, recent developments, and challenges," *ArXiv*, vol. abs/2309.02473, 2023.

[3] C. Finn, T. Yu, T. Zhang, P. Abbeel, and S. Levine, "One-shot visual imitation learning via meta-learning," *ArXiv*, vol. abs/1709.04905, 2017.

[4] S. Bansal, V. Tolani, S. Gupta, J. Malik, and C. Tomlin, "Combining optimal control and learning for visual navigation in novel environments," in *Conference on Robot Learning (CoRL)*, 2019.

[5] Y. Pan, C.-A. Cheng, K. Saigol, K. Lee, X. Yan, E. A. Theodorou, and B. Boots, "Agile autonomous driving using end-to-end deep imitation learning," *Robotics: Science and Systems XIV*, 2017.

[6] A. Reichlin, G. L. Marchetti, H. Yin, A. Ghadirzadeh, and D. Kragic, "Back to the manifold: Recovering from out-of-distribution states," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2022, pp. 8660–8666.

[7] K.-C. Hsu, H. Hu, and J. F. Fisac, "The safety filter: A unified view of safety-critical control in autonomous systems," *arXiv preprint arXiv:2309.05837*, 2023.

[8] A. D. Ames, X. Xu, J. W. Grizzle, and P. Tabuada, "Control barrier function based quadratic programs for safety critical systems," *IEEE Transactions on Automatic Control*, vol. 62, no. 8, pp. 3861–3876, 2017.

[9] A. Bajcsy, S. Bansal, E. Bronstein, V. Tolani, and C. J. Tomlin, "An efficient reachability-based framework for provably safe autonomous navigation in unknown environments," in *IEEE Conference on Decision and Control (CDC)*, 2019.

[10] R. K. Cosner, Y. Yue, and A. D. Ames, "End-to-end imitation learning with safety guarantees using control barrier functions," in *2022 IEEE 61st Conference on Decision and Control (CDC)*. IEEE, 2022, pp. 5316–5322.

[11] M. Laskey, J. Lee, R. Fox, A. Dragan, and K. Goldberg, "DART: Noise injection for robust imitation learning," in *Conference on Robot Learning*, 2017.

[12] L. Ke, J. Wang, T. Bhattacharjee, B. Boots, and S. Srinivasa, "Grasping with chopsticks: Combating covariate shift in model-free imitation learning for fine manipulation," in *IEEE International Conference on Robotics and Automation*, 2021, pp. 6185–6191.

[13] J. Ho and S. Ermon, "Generative adversarial imitation learning," in *Neural Information Processing Systems*, 2016. [Online]. Available: https://api.semanticscholar.org/CorpusID:16153365

[14] S. Ross, G. Gordon, and D. Bagnell, "A reduction of imitation learning and structured prediction to no-regret online learning," in *International Conference on Artificial Intelligence and Statistics*, 2011.

[15] J. Zhang and K. Cho, "Query-efficient imitation learning for end-to-end autonomous driving," *arXiv preprint arXiv:1605.06450*, 2016.

[16] S. Bansal, M. Chen, S. Herbert, and C. J. Tomlin, "Hamilton-Jacobi Reachability: A brief overview and recent advances," in *IEEE Conference on Decision and Control (CDC)*, 2017.

[17] I. Mitchell, A. Bayen, and C. J. Tomlin, "A time-dependent Hamilton-Jacobi formulation of reachable sets for continuous dynamic games," *IEEE Transactions on Automatic Control (TAC)*, vol. 50, no. 7, pp. 947–957, 2005.

[18] I. Mitchell, "A toolbox of level set methods," *http://www. cs. ubc. ca/mitchell/ToolboxLS/toolboxLS. pdf, Tech. Rep. TR-2004-09*, 2004.

[19] S. Bansal, M. Chen, K. Tanabe, and C. J. Tomlin, "Provably safe and scalable multivehicle trajectory planning," *IEEE Transactions on Control Systems Technology (TCST)*, vol. 29, no. 6, pp. 2473–2489, 2020.

[20] S. Bansal and C. J. Tomlin, "Deepreach: A deep learning approach to high-dimensional reachability," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, 2021, pp. 1817–1824.

[21] J. Borquez, K. Nakamura, and S. Bansal, "Parameter-conditioned reachable sets for updating safety assurances online," *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 10 553–10 559, 2022. [Online]. Available: https://api.semanticscholar.org/CorpusID:252596075

[22] I. M. Mitchell, "A toolbox of level set methods," 2005.

[23] M. Laskey, J. Lee, R. Fox, A. D. Dragan, and K. Goldberg, "Dart: Noise injection for robust imitation learning," in *Conference on Robot Learning*, 2017.

[24] 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*, ser. Proceedings of Machine Learning Research, G. Gordon, D. Dunson, and M. Dudík, Eds., vol. 15. Fort Lauderdale, FL, USA: PMLR, 11–13 Apr 2011, pp. 627–635. [Online]. Available: https://proceedings.mlr.press/v15/ross11a.html

[25] S. M. Katz, A. Corso, C. A. Strong, and M. J. Kochenderfer, "Verification of image-based neural network controllers using generative models," *2021 IEEE/AIAA 40th Digital Avionics Systems Conference (DASC)*, pp. 1–10, 2021.

[26] K. D. Julian, R. Lee, and M. J. Kochenderfer, "Validation of image-based neural network controllers through adaptive stress testing," *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*, pp. 1–7, 2020.

[27] T. Byun and S. Rayadurgam, "Manifold for machine learning assurance," *2020 IEEE/ACM 42nd International Conference on Software Engineering: New Ideas and Emerging Results (ICSE-NIER)*, pp. 97–100, 2020.

[28] 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*, vol. 43, no. 5, pp. 137–177, 2023.

[29] J. Borquez, K. Chakraborty, H. Wang, and S. Bansal, "On safety and liveness filtering using hamilton-jacobi reachability analysis," *arXiv preprint arXiv:2312.15347*, 2023.

[30] K.-C. Hsu, D. P. Nguyen, and J. F. Fisac, "Isaacs: Iterative soft adversarial actor-critic for safety," in *Learning for Dynamics and Control Conference*. PMLR, 2023, pp. 90–103.

[31] K. Kim, G. Swamy, Z. Liu, D. Zhao, S. Choudhury, and S. Z. Wu, "Learning shared safety constraints from multi-task demonstrations," *Advances in Neural Information Processing Systems*, vol. 36, 2024.