A Framework for Online Updates to Safe Sets for Uncertain Dynamics

Jennifer C. Shih, Franziska Meier, Akshara Rai

***Abstract*— Safety is crucial for deploying robots in the real world. One way of reasoning about safety of robots is by building safe sets through Hamilton-Jacobi (HJ) reachability. However, safe sets are often computed offline, assuming perfect knowledge of the dynamics, due to high compute time. In the presence of uncertainty, the safe set computed offline becomes inaccurate online, potentially leading to dangerous situations on the robot. We propose a novel framework to learn a safe control policy in simulation, and use it to generate online safe sets under uncertain dynamics. We start with a conservative safe set and update it online as we gather more information about our dynamics. We also show an application of our framework to a model-based reinforcement learning problem, proposing a safe model-based RL setup. Our framework enables robots to simultaneously learn about their dynamics, accomplish tasks, and update their safe sets. It also generalizes to complex high-dimensional dynamical systems, like 3-link manipulators and quadrotors, and reliably avoids obstacles, while achieving a task, even in the presence of unmodeled noise.**

I. Introduction

Machine learning can help robots adapt to unseen scenarios, but the fear of damaging the robot or its environment hinders its deployment in the real world. Combining learning algorithms with control theoretic tools, developed to ensure the safety of dynamical systems, can help with this. In our work, we build on Hamilton-Jacobi (HJ) reachability [1], a control-theoretic framework that offers safety guarantees for dynamical systems. Intuitively, HJ reachability computes the set of safe states and an action policy that can together ensure that the system does not enter a dangerous zone. Once safe sets are computed, they can be used in combination with learning algorithms to build frameworks that can be guaranteed to be safe, while achieving a given task.

However, computation of safe sets suffers from the curse of dimensionality, due to discretization of the state space. [2] reported taking 3 days for reachability computations on a 4-dimensional system and exact computation of reachability is intractable for systems with more than 5 dimensions. As a result, the optimal safe policy and safe set are often computed offline for low-dimensional systems, assuming perfect knowledge of the dynamics. In the presence of uncertainties, however, the pre-computed safe sets might not be valid, and can lead to dangerous situations on the robot. This makes it important to update the safe sets online. [3], [4] address this for low-dimensional systems.

In this work, we present a *least-restrictive* safety framework that can be used in combination with any type of controllers under uncertain dynamics by updating safe sets online for complex high-dimensional dynamical systems. Specifically, we focus on scenarios where the dynamics are inaccurate due to uncertainty in rigid body parameters such as mass, inertia, and center of mass location of links. This is a common source of uncertainty in robots, such as manipulators, but their identification typically does not take safety into account. In our proposed framework, we learn a safe policy offline by considering a distribution of dynamics for high-dimensional systems using reinforcement learning (RL) by building on [5]. Online, we update the distribution of our belief of the dynamics parameters from data, and recompute the safe set by forward simulating the safe policy the new belief of our dynamics parameters. As a result, we are not overly conservative during experiments, while staying safe. Figure 1 provides a high level overview of our approach.

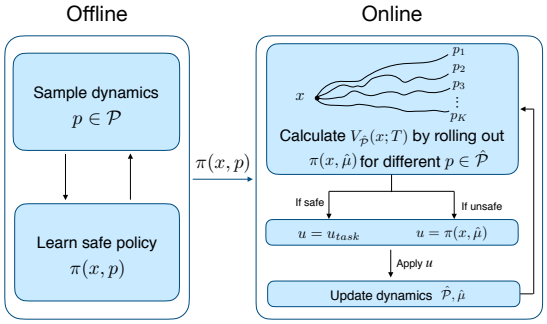


Fig. 1: An outline of our approach. Offline, we learn a safety policy $\pi(x, p)$, which depends on the state x and dynamics parameters p, by randomly sampling different dynamics parameters p. Online, we roll out $\pi(x, \hat{\mu})$ using the current estimate of dynamics, and use it to determine the safety value at current state x. If the state is deemed safe, a task policy is applied, and otherwise a safety policy is applied. Finally, the dynamics parameters $\hat{\mathcal{P}}$ and $\hat{\mu}$ are updated based on the new data.

The central ideas of in our framework can also be directly used with learning approaches with optimization components, like model-based RL [6]. We demonstrate incorporating safety derived from our framework in a model-based RL setting, by adding a safety constraint to the optimization, thereby proposing a safe model-based RL setup where the safe policy and task policy can interact with each other.

Our work is a step towards online updating and sim-to-real transfer of reachability sets for high-dimensional systems. We test the efficacy of our proposed framework at avoiding obstacles during random exploration on a 8-dimensional quadrotor, and a 3-link manipulator (6-dimensional state

Jennifer Shih is with the Department of Electrical Engineering and Computer Sciences, University of California, Berkeley. cshih@berkeley.edu. Franziska Meier, Akshara Rai are with Facebook AI Research. {fmeier, akshararai}@fb.com

space). In our experiments, our approach is able to avoid obstacles reliably, as compared to using a nominal safe set, especially for complex dynamics such as the 3-link manipulator in challenging scenarios. While our work is shown only in simulation, we emulate sim-to-real differences by perturbing dynamics parameters and adding unmodeled noise. We also tested incorporating safety derived from our framework a model-based RL setting on a 2-link manipulator and show that it is much safer when learning to complete a task compared to using standard model-based RL. Our results showcases the robustness of our framework, and opens promising applications to robotic systems in the future.

II. Background and related work

Safety of dynamical systems has been widely studied in control literature, and used in combination with learning approaches to ensure safe learning. Constrained optimization, for example with barrier functions, can also be used to certify probabilistic safety [7] or guide exploration to safe regions [8]. However, barrier function methods are generally limited to control-affine systems and the uncertainty considered in these papers are input-independent. MPC approaches have also been proposed to address safety as in [9], [10]. However, safety computations in these works are either applied to linear systems or locally linear approximations of nonlinear systems. In comparison, our proposed approach can be applied to general nonlinear dynamical systems directly and the uncertainty considered in our proposed framework can be input-dependent.

Works such as [11], [12], [13] use Lyapunov functions to guarantee or encourage safety during learning. However, [11] requires a Lyapunov function for a given system, [12] is limited to discrete action spaces, and [13] uses approximations in its proposed Lyapunov constraints when addressing safety. This can limit their applicability to the safety of complex high-dimensional complex robots.

Our paper focuses addressing safety with HJ reachability [1], which is a control-theoretic framework that provides safety guarantees for general dynamical systems. There is a lot of work on guaranteeing safety of single-agent and multi-agent systems using reachability under different assumptions on the dynamics and agent formation, such as [14], [15], [16], [17]. [18] provides an overview. We will address the most relevant HJ reachability works in the following subsections.

A. Hamilton Jacobi (HJ) Reachability

In this work, we focus on the single-agent setting for HJ reachability. We consider a single-player system with state $x \in \mathcal{X}$ and action $u \in \mathcal{U}$ and dynamics $\dot{x} = f(x, u)$. We assume that f is uniformly continuous, bounded, and Lipschitz continuous in arguments x for fixed u. In addition, the control function $u(\cdot) \in \mathbb{U}$ is drawn from the set of measurable functions.

Let $\mathcal{Z}$ represent the danger zone, which is a set of states that the robot should avoid. We can use a level set function $l(x)$ to capture $\mathcal{Z}$ by defining it so that $l(x) < 0$ if and only if $x \in \mathcal{Z}$.

Let $\xi_x^{\boldsymbol{u}}(\cdot)$ be a trajectory resulting from executing $\boldsymbol{u}(\cdot)$ from state x. Then the value function V at a state x is defined as

$$V(x) = \underset{\boldsymbol{u}(\cdot)}{sup}\ \underset{t \geq 0}{inf}\ l(\xi_x^{\boldsymbol{u}}(t)).$$

Intuitively, $V(x)$ is the closest the trajectory gets to danger zone $\mathcal{Z}$ given the *best possible control* to avoid the danger zone. We define the safe set $\mathcal{K}$ as $\mathcal{K} = \{x : V(x) \geq 0\}$ and the unsafe set as the complement of of the safe set $\mathcal{K}$. Intuitively, the safe set is the set of states such that there exists at least one control strategy for the system to avoid the danger zone $\mathcal{Z}$.

For a finite time horizon $t \in [0, T]$, this value function can be computed by solving the Hamilton-Jacobi-Bellman variational inequality described in [1] for continuous systems. In this work, we build on [5] for approximating the *best possible control* with reinforcement learning (RL), which typically adopts a discrete-time formulation. Hence in this paper, we work with discrete-time version of reachability and the infinite-horizon value function $V(x)$ is defined as

$$V(x) = min\left\{l(x), \underset{u \in \mathcal{U}}{max} V(x + f(x, u)\Delta t)\right\}.$$

In practice, due to errors introduced by discretization, it is common to introduce a safety level $\epsilon > 0$ such that the safe set is defined as $\mathcal{K} = \{x : V(x) \geq \epsilon\}$ in order to ensure safety. We adopt this convention in this paper.

B. Learning to approximate HJ safe sets

It is intractable to compute exact reachable sets for dynamical systems with more than five states, due to discretization of state space, and the resulting curse of dimensionality. As a result, there has been some work on approximating safe sets for high-dimensional problems.

[2] proposed to use function approximators to represent the optimal safety policy for control-affine systems, which in turn generates a safe set. [5] propose a time-discounted Safety Bellman Equation that adapts the standard dynamic programming backup to induce a contraction mapping in the space of value functions. As a result, RL algorithms designed for temporal difference learning can now be used to learn safe sets for single-player systems.

In practice, we found that the method proposed by [5] learns policies for complicated or high dimensional systems more reliably. However, it assumes perfect knowledge of the dynamics and hence does not account for discrepancies between offline training and online environment, which can cause the system to enter the danger zone online. We adapt this approach to learn a safety policy under dynamics uncertainty. Next, we use the learned safety policy to compute the safe set online, while updating the estimate of the dynamics from data.

C. Online updates to HJ safe sets in robotics problems

In this section, we give an overview of papers that address online updates to safe sets in the presence of uncertainties

Algorithm 1: Online loop of proposed framework

Given : safety policy $\pi(x,p)$, $\hat{\mathcal{P}}_0$, $\hat{\mu}_0$, x_0, T, ϵ, $t=0$

while $t < T$ **do**
  $s_t = V_{\hat{\mathcal{P}}_t}(x_t;T)$; // safety value based on $\hat{\mathcal{P}}_t$, $\hat{\mu}_t$
  if $s_t \geq \epsilon$ **then**
    $u_t =$ any action (action from a learning controller or random action)
  else
    $u_t = \pi(x_t, \hat{\mu}_t)$
  end
  $x_{t+1} = f(x_t, u_t; p_{true})$;
  Update $\hat{\mathcal{P}}_{t+1}, \hat{\mu}_{t+1}$ using (x_t, u_t, x_{t+1}) ;
  $t \leftarrow t+1$;
end

in dynamics or the environment. [3] propose to use local updates and warm-start to generate safe sets online, as static obstacles in the environment are detected. However, this still relies on discretization of the state space and is hence intractable for high-dimensional systems.

[4] use a Gaussian process (GP) to model uncertainty in the dynamics of the system, followed by HJ reachability to compute a safe set. However, this also relies on online re-computation of HJ reachable sets, and is hence intractable for high-dimensional systems. Moreover, the uncertainty considered in this work is input-independent, which is easily violated for common platforms such as robot manipulators.

In contrast, our online update framework can be used on high-dimensional problems, and does not assume input-independent disturbance. We assume inaccuracy in dynamics parameters of a robot and aim to identify these parameters online, while maintaining safety. This is relevant for tasks such as lifting heavy objects, where the added mass can affect a manipulator's dynamics.

D. Model-based Reinforcement Learning

Our safety framework can be used in combination with model-based reinforcement learning. In this section, we give a brief overview of model-based RL, a sample-efficient learning framework, that has shown recent success at complex robotics tasks [19].

Model-based reinforcement learning (MBRL) iteratively tries to optimize a policy to accomplish a task, and learns the dynamics of the robot. Similar to [19], we use model-predictive control (MPC) to optimize the policy. The objective of model-predictive control (MPC) is to minimize the cost $J = \sum_{h=t}^{t+H-1} c(\mathbf{x}_h, \mathbf{u}_h)$ with respect to the actions $u_{t:t+H-1} \equiv \{u_t, \ldots, u_{t+H-1}\}$, given the dynamics f over a horizon H from current time t. The first action u_t is applied on the system, and the process repeats, starting with the new current state.

We present an application of safe sets in model-based RL in Section III-C.

III. Framework for safe set computations for uncertain dynamics

Next, we present our framework for offline training and online updates to safe sets, outlined in Figure 1. During the offline phase, we train a safe policy $\pi(x,p)$, which takes the state x and dynamics parameters p as input. During the online phase, we forward simulate this safe policy from the current state x, using our current belief of the dynamics parameters. This computes an approximate safety value function $V(x)$ under our range of estimated dynamics and safety policy $\pi(x,\hat{\mu})$, where $\hat{\mu}$ is the current *best* estimate of the dynamics parameters. If $V(x) \geq \epsilon$, we are in the safe set, and can apply any action relating to the task. Otherwise, we apply the safe policy to avoid entering the danger zone. The resulting state transition on the robot is then used to update our belief about the dynamics parameters p.

A. Offline computation of safety policy

During offline computation, we use reinforcement learning to train a safety policy $\pi(x,p)$ which is a function of both the state x and the dynamics parameters p. Every N episodes, we sample a new set of dynamics parameters p and use them to collect data to train $\pi(x,p)$. This data is generated by sampling 'true' dynamics parameters of the simulator from the set $\mathcal{P} = [p-dp, p+dp]$ for some positive dp. For a fixed p, the safe policy $\pi(x,p)$ is thus trained on data from a distribution of dynamics, with dynamics parameters drawn from $\mathcal{P}$, making it robust to slight variance in the estimated dynamics. The value of dp is chosen based on the estimated uncertainty on the dynamics parameters during test time.

To train $\pi(x,p)$, we adapted the update rule from [5] to suit our purposes. We use Soft Actor-Critic [20] to train the policy and value function together, as a function of the dynamics parameters p, along with the state. For our purpose, the update rule of the Q-Value function we use during RL training is:

$$Q(x,p,a) \leftarrow (1-\gamma)l(x) + \gamma min\left\{l(x), \underset{a' \in \mathcal{U}}{max} Q(f(x,a;p),p,a')\right\},$$

where $f(x,a;p)$ is the discrete dynamics of the robot given that its dynamics parameters are p and γ is the discount factor.

B. Online updates to safe set

Given the safety policy $\pi(x,p)$ learned offline, we can generate safe sets online based on our estimate of the distribution of the parameters p. We iteratively perform the following steps at every time step, t: (1) Compute the safety value $V_{\hat{\mathcal{P}}_t}(x)$ of the current state x for the current estimate of the dynamics parameters set $\hat{\mathcal{P}}_t$. (2) Execute an action on the robot based on whether the safety value is below the safety threshold ϵ. (3) Update the estimate of the distribution of the dynamics parameters using new data gathered.

Algorithm 1 summarizes our framework for the online computations. We describe these three steps during the online phase in detail below.

1) Computing the online safe set: At any given time step t, we maintain an estimate of $\hat{\mathcal{P}}_t$, a set that dynamics parameters p fall in with high probability c. We also maintain a current best estimate $\hat{\mu}_t$ of the parameters. For example, for a one-dimensional p, if we estimate our belief of p with a Gaussian distribution, i.e., $p \sim \mathcal{N}(\hat{\mu}_t, \hat{\sigma}_t)$ and $c = 0.95$, then $\hat{\mathcal{P}}_t = [\hat{\mu}_t - 2\hat{\sigma}_t, \hat{\mu}_t + 2\hat{\sigma}_t]$. $\hat{\mathcal{P}}_t$ can also be a discrete set if the dynamics parameters are drawn from a discrete distribution.

The safety value at any state x at time t for a fixed time horizon T is computed as follows:

$$V_{\hat{\mathcal{P}}_t}(x; T) = \min_{p \in \hat{\mathcal{P}}_t} V(x; p, T), \tag{1}$$

where $V(x; p, T)$ is the estimated safety value for dynamics parameter p and time horizon T. To compute $V(x; p, T)$, we roll out the optimal policy $\pi(x, \hat{\mu}_t)$ from x for horizon T. We denote the resulting trajectory as $\xi_{x,p}^{\pi_{\hat{\mu}_t}, T}(\cdot)$. Hence we compute $V(x; p, T)$ as follows

$$V(x; p, T) = \min_{t' \in \{t, t+1, \ldots, t+T\}} l\left(\xi_{x,p}^{\pi_{\hat{\mu}_t}, T}(t')\right). \tag{2}$$

Intuitively, $V(x; p, T)$ is the minimum distance between the robot and the danger zone, when executing the policy $\pi(x, \hat{\mu}_t)$, if the true dynamics parameters were p, over a trajectory of length T. At any time t, we can compute the safety values at all states x in the space and form a safety set based on this. However, in practice, we only need the safety value at the current state x_t to determine the safety of the robot. By taking the minimum of $V(x; p, T)$ over all possible dynamics parameters in $\hat{\mathcal{P}}_t$ when computing the safety value $V_{\hat{\mathcal{P}}_t}(x; T)$ at state x, the framework employs an avoidance control if *any* dynamics parameter in $\hat{\mathcal{P}}_t$ results in a value of $V(x; p, T)$ that's less than the safety threshold ϵ.

As described in Equation 1, to compute the safety value $V_{\hat{\mathcal{P}}_t}(x_t; T)$ at x_t, we need to forward simulate dynamics for all $p \in \hat{\mathcal{P}}_t$. When $\hat{\mathcal{P}}_t$ is a continuous set, in practice, we discretize finely over $\hat{\mathcal{P}}_t$ and simulate the dynamics with each of the discretized values of the dynamics parameters in parallel.

Given our proposed approach, we now formally present a proof of guaranteed safety of our proposed framework under specific conditions.

Theorem 1: Assume that $\hat{\mathcal{P}}_t$ is a discrete set. For any time t and state x, if the true dynamics parameter $p_{true} \in \hat{\mathcal{P}}_t$, $V_{\hat{\mathcal{P}}_t}(x; T) \leq V^\star(x; p_{true}, T)$ for the fixed time horizon T where $V^\star(x; p_{true}, T)$ is the true value function based on the true dynamics parameters p_{true} under discrete-time dynamics.

Proof: First, we remind the reader that $V(x; p, T)$ is the safety value at x derived from using policy $\pi(x, \hat{\mu}_t)$ as seen in equation 2. Now we know that for any p, $V(x; p, T) \leq V^\star(x; p, T)$ because $\pi(x, \hat{\mu})$ is at most as good as the true optimal policy $\pi^\star(x, p)$. Hence, $V_{\hat{\mathcal{P}}_t}(x; T) = \min_{p \in \hat{\mathcal{P}}_t} V(x; p, T) \leq \min_{p \in \hat{\mathcal{P}}_t} V^\star(x; p, T) \leq V^\star(x; p_{true}, T)$, where the second inequality holds because $p_{true} \in \hat{\mathcal{P}}_t$ under our assumption. ■

This implies that our framework results in a conservative estimate of the true safe set under the aforementioned assumptions, as long as $p_{true} \in \hat{\mathcal{P}}_t$. For cases where dynamics parameters are drawn from a continuous set, we discretely sample in the dynamics set $\hat{\mathcal{P}}_t$. As a result, we lose guaranteed safety, but we still demonstrate empirically that using our proposed approach is safer than using nominal safe sets.

2) Determining the action to take: If $V_{\hat{\mathcal{P}}_t}(x_t; T) \geq \epsilon$, we can guarantee that by executing the safe control $\pi(x_t, \hat{\mu}_t)$, we can avoid the danger zone for time horizon T, under assumptions outlined in 1. Although these conditions may not be satisfied in practice, this decision rule still provides a good criterion for selecting whether or not to execute the safe policy. When the robot is determined safe, we can apply any action, such as an action determined by any learning controller. When $V_{\hat{\mathcal{P}}_t}(x_t; T) < \epsilon$, we apply the action determined by the safety policy. In this sense, we have a *least-restrictive* safety framework such that the robot is free to perform any action until it is close to the unsafe set, at which point, the safety controller takes over.

3) Updating dynamics from data: To update our belief about the dynamics parameters p, we can use any system identification approach that gives us a probabilistic estimate of p. Identifying dynamics parameters such as for robot manipulators is widely studied in robotics, such as in [21], [22]. In this paper, we use Bayesian Linear Regression (BLR) to update our belief over $\hat{\mu}_t$ and $\hat{\sigma}_t$ for the Gaussian distribution, and in turn use them to update $\hat{\mathcal{P}}_t$. Please see Chapter 3 [23] for details about Bayesian Linear Regression (BLR). This gives us an online, continual way of learning dynamics parameters that can easily generalize to tasks such as picking and placing heavy objects. In cases where the uncertain parameters are not linear in the dynamics, we can use more advanced techniques for parameter identification, such as [22].

C. Safe Model-based Reinforcement Learning

Our approach can easily be used in combination with model-based learning approaches, like model-based RL. We add a safety constraint to the model-based RL optimization, which renders the search for the optimal policy to be biased towards safety. Given current state x_t and the current best estimate $\hat{\mu}_t$ of the parameters, the safe model-based RL optimization problem becomes

$$u_{t:t+H-1} = \arg\min_{u_{t:t+H-1}} \sum_{h=t}^{t+H-1} c(x_h, u_h) \tag{3}$$

$$\text{s.t.} \quad x_{h+1} = f(x_h, u_h; \hat{\mu}_t) \tag{4}$$

$$V_{\hat{\mathcal{P}}_t}(x_{h+1}; T) \geq \epsilon \quad \forall h \in \{t, \ldots, t+H-1\} \tag{5}$$

where $f(x_h, u_h; \hat{\mu}_t)$ is the discrete dynamics function assuming the dynamics paramters are $\hat{\mu}_t$. After u_t is applied on the robot, the (state,action,next state) transition is measured and used to update the our belief about the dynamics. The current state is updated to the new state, and the process repeated. If no feasible action sequence can be found, the safety

policy is applied, i.e., $u_t = \pi(x_t, \hat{\mu}_t)$. [24] propose a related setup for model-predictive control, with ellipsoidal safe sets with linearized dynamics. On the other hand, our approach incorporates safety directly using the original dynamical system.

Note that the planning horizon H is typically chosen to be much shorter than the safety horizon T. This is because increasing H increases the dimensionality of the optimization (actions $u_{t:t+H-1}$). Due to non-convexity of the problem, this can lead to poorer solutions for longer horizons. On the other hand, T does not affect the optimization dimension, and can be much larger. This ensures that even though our model-based RL algorithm has a short foresight, our safety constraint maintains a longer horizon plan for safety. We use random sampling in the space of actions to optimize the optimization problem described in Equation 3.

IV. Illustration : Dubins Car

In this section, we demonstrate our proposed framework on a 3-dimensional Dubins car obstacle avoidance problem. The dynamics of Dubins car can be described as:

$$\dot{q_x} = v\cos\theta, \quad \dot{q_y} = v\sin\theta, \quad \dot{\theta} = \omega. \tag{6}$$

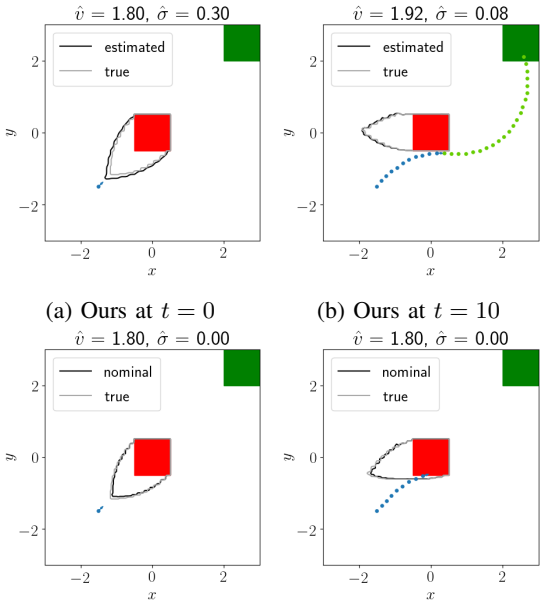
The 3-dimensional state variables q_x, q_y, θ represent the (x, y)-position and the heading of the vehicle. v is the constant speed. The control input ω is the rate of change of the heading of the vehicle. Input is constrained to be $|\omega| \leq 1$. The uncertain dynamics parameter p is the speed v, emulating measurement noise or sensor biases. The initial estimated speed is $\hat{v}_0 = 1.8$ m/s, while the true speed is $v_{true} = 2.0$ m/s. We add Gaussian noise with zero mean and standard deviation 0.1 to the transitions of the true dynamics model to further simulate differences between simulation and real world.

A. Obstacle avoidance with our framework vs. using a nominal safe set

Figure 2 shows an example where the vehicle is avoiding an obstacle (red) and reaching a goal (green). Our method starts with the initial estimate $\hat{v}_0 = 1.8, \hat{\sigma}_0 = 0.3$, which gives us $\hat{\mathcal{P}}_0 = [1.2, 2.4]$. On the other hand, the baseline assumes the nominal speed $\hat{v}_0 = 1.8$ throughout the execution.

During the offline learning phase, we use Q-learning to train the optimal safety policy $\pi(x, p)$. Online, as more data is collected, our estimate of dynamics is updated and the safety set is updated online as described in Section III-B.1.

As seen in Figure 2, with our approach, the vehicle successfully avoids the obstacle and reaches the goal (top row), while with the safe set from inaccurate nominal dynamics, the vehicle hits the obstacle (bottom row). The black and grey curves in each figure show the initial unsafe set computed based on the estimated speed and the true speed, respectively. As expected, the unsafe set (black) computed using our framework is larger than the true unsafe set (grey), as we take into account the uncertainty in the dynamics. However, as we learn more about the dynamics and update our estimated safe set, the difference between our estimate and the true safe set becomes smaller. This experiment shows the benefit



(a) Ours at $t = 0$ (b) Ours at $t = 10$

(c) Nominal at $t = 0$ (d) Nominal at $t = 8$

Fig. 2: Behavior of using our framework (top row) vs. using the nominal safe set (bottom row). Our framework avoids obstacle (red) early and successfully reaches the goal (green). Using the nominal speed, the vehicle avoids too late, and hits the obstacle. The blue dots represent trajectory to the current frame and the green dots in the top right graph illustrate future trajectory. For each safe set plot, we plot the slice of the safe set over x-y plane at the current heading θ of the vehicle.

of using our framework even when the estimation error in the dynamics parameter is small.

For this 3-dimensional system, at each time step, the computation time for updating estimate of dynamics and computing the updated safe value is about 1×10^{-4} s and 0.056 s respectively. In comparison, [4] take about 2 s to recompute the safety set for a 2-dimensional system.

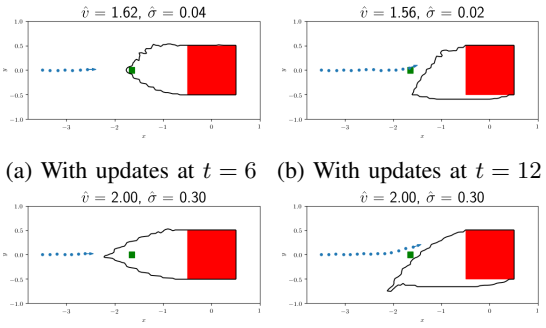
B. Benefit of online update vs. conservative initial estimate

Offline, we learn a conservative estimate of the safe set which can keep the system safe. However, it can be too restrictive in some settings, and online updates can help achieve better performance by making the safe set less restrictive over time. This is illustrated in Figure 3, where using the initial conservative estimate (bottom row) causes the vehicle to miss the goal as it starts avoiding the obstacle too soon. On the other hand, using the updated safe set (our approach, top row) allows the robot to reach the goal.

V. Experiments

In this section, we present experimental results on 2-link and 3-link manipulators (4D and 6D problems), and an 8D quadrotor system. We perform extensive experiments to compare performance of our framework against standard reachability, with inaccurate nominal dynamics. Furthermore we present results on the benefit of using MBRL with safety constraints derived from our framework, versus using MBRL with no safety constraints.

We experiment with two different kinds of scenarios, random and challenging. We observe that the system gets



(a) With updates at $t = 6$ (b) With updates at $t = 12$

(c) No updates at $t = 6$ (d) No updates at $t = 13$

Fig. 3: Benefit of using online updates (top row) vs. using the initial conservative estimate (bottom row) of the safe set without updating our dynamics estimate. The black curve shows the unsafe set. Both methods ensure safety, however, always using the initial estimate can make the system overly cautious and miss the goal.

closer to unsafe states much more commonly in challenging scenarios than random scenarios. In general, our method shines in conditions that were tough. For safety, it is important to avoid obstacles in all scenarios, and challenging initial conditions showcase the robustness of our approach vs. using the nominal safe set.

In all trials across different scenarios and methods, the nominal dynamics parameter is always $\hat{p}_0 = 2$. For the random initialization, the true dynamics is sampled from the set $p_{true} \sim$ uniform$[\hat{p}_0 - 2\sigma, \hat{p}_0 + 2\sigma]$ and for the difficult scenario the true dynamics is sampled from $p_{true} \sim$ uniform$[\hat{p}_0 - 2\sigma, \hat{p}_0]$, for $\sigma = 0.1, 0.3$.

In our first set of experiments, we simulate 200 trials for various dynamical systems. Each trial lasts for $T = 100$ time steps. We define a trial successful if the robot does not hit any obstacle throughout the entire trial, and terminate early if it hits the obstacle. In each trial, we apply a uniformly random action, if the algorithm determines that the robot is safe, and apply the safety policy otherwise. We compare the rate of success of our framework with that of a nominal safe set with inaccurate dynamics. We use Soft Actor-Critic [20], implementation from [25], to train the safety policy $\pi(x, p)$ for all dynamical systems.

Both methods start from the same, safe initial condition. To emulate sim to real differences, we add random Gaussian noise with zero mean and standard deviation of 0.1 to the control inputs for all systems. Note that the noise added to the control input does not satisfy the assumptions of BLR (III-B.3), but our proposed method consistently outperforms the baseline in challenging scenarios. In addition, for the 3-link manipulator, we also performed an extensive experiment where our framework assumes a damping coefficient different from the true damping coefficient, without updating our belief. This shows the robustness of our approach to situations that violate the assumptions of our algorithm.

A. 2-link manipulator

We consider the task of safely controlling a 2-link manipulator in the x-y plane. The dynamics of a manipulator are:

$$M(q)\ddot{q} + C(q, \dot{q})\dot{q} = u. \tag{7}$$

Here $q = [\theta_1, \theta_2]$ are the joint angles of the two links, $M(q)$ is the inertia matrix, and $C(q, \dot{q})$ is the Coriolis matrix. The full state of the system is 4-dimensional, $x = [\theta_1, \theta_2, \dot{\theta}_1, \dot{\theta}_2]$. $u = [\tau_1, \tau_2]$ is the 2-dimensional control input or torque in the joints, which is constrained to $|\tau_1|, |\tau_2| \leq 1$. We use simplified dynamics with masses at the end of each link (each of length $1.0\,\mathrm{m}$), and first mass $m_1 = 1.0\,\mathrm{kg}$. The uncertainty in the dynamics comes from uncertainty in the mass at the end-effector $p = m_2$. The square obstacle has edge length $1.0\mathrm{m}$.

While we did not consider gravity in our dynamics in Equation 8, these dynamics generalize to manipulators with manufacturer-provided gravity compensation, such as Kuka LBR [26]. The initialization of the states for the random and challenging scenarios are:

	$[\theta_1, \theta_2]$	$[\dot{\theta_1}, \dot{\theta_2}]$
Random	$[0, 0] + x_{rand}$	$[0, 0] + dx_{rand}$
Challenging	$[\frac{\pi}{7}, \frac{\pi}{6}] + x_{chal}$	$[0.3, 0.4] + dx_{chal}$

where each variable is sampled from the uniform distribution within the range: $x_{rand} : [-\pi, \pi]$ rad, $dx_{rand} : [-0.5, 0.5]$rad/s, $x_{chal} : [-0.5, 0.5]$ rad, and $dx_{chal} : [-0.5, 0.5]$rad/s. Figure 4 visualizes the challenging initial conditions.

As summarized in Table I, with random initialization, the success rates for our proposed framework and using the nominal safe sets are similar. However, in the challenging scenario with $\sigma = 0.3$, our approach successfully avoids the obstacle with a 99.5% success rate, while using the nominal safe set only succeeds 85% of the time. This shows that even on a 4-dimensional system, inaccurate dynamics can adversely affect the performance of safety approaches, especially in challenging scenarios. In such a case, online updates to the safe sets can significantly improve rate of successfully avoiding obstacles, and other dangerous situations.

B. 3-link manipulator

The 3-link manipulator has similar dynamics to the 2-link system, but with an additional link. This makes the state space 6-dimensional and action space 3-dimensional, adding computational complexity. The full state is $x = [\theta_1, \theta_2, \theta_3, \dot{\theta}_1, \dot{\theta}_2, \dot{\theta}_3]$. $u = [\tau_1, \tau_2, \tau_3]$ is the torque, where $|\tau_1| |\tau_2|, |\tau_3| \leq 1$. Note that this system is intractable for standard reachability. The manipulator has masses $m_1 = 1.0, m_2 = 2.0$ and m_3 at the end of each link (each of length 1.0). The uncertainty in the dynamics comes from uncertainty in the mass at the end-effector $p = m_3$.

The initialization of the states are described as follows:

	$[\theta_1, \theta_2, \theta_3]$	$[\dot{\theta}_1, \dot{\theta}_2, \dot{\theta}_3]$
Random	$[0, 0, 0] + x_{rand}$	$[0, 0, 0] + _{rand}$
Challenging	$[\frac{\pi}{7}, \frac{\pi}{6}, \frac{\pi}{5}] + x_{chal}$	$[0.2, 0.2, 0.2] + dx_{chal}$

where each variable is sampled from the uniform distribution within the range: $x_{rand} : [-\pi, \pi]$ rad, $dx_{rand} : [-0.5, 0.5]$ rad/s, $x_{chal} : [-0.2, 0.2]$ rad, and $dx_{chal} : [-0.2, 0.2]$ rad/s. Figure 4 visualizes the challenging initial conditions.

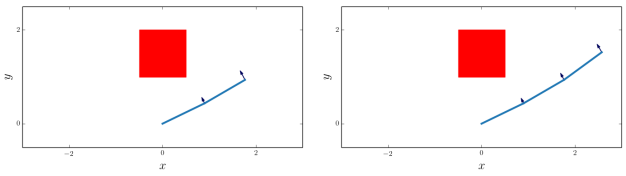


Fig. 4: Environments and the challenging initial conditions that we randomize around for the 2-link and 3-link manipulators experiments. The arrows represent the velocities at the joints and end effectors. The red squares are the obstacles.

In the random initialization scenario, we observe comparable performance between our framework v.s. the baseline, with ours succeeding 96.5% of the time and the baseline succeeding 91.5% of the time for $\sigma = 0.3$. Our framework shines in this complicated dynamical systems in difficult scenarios, with the success rate being 53.0% with our framework and 35.5% for the baseline for $\sigma = 0.1$. The performance of all reachability approaches gets worse as the complexity of the dynamics increase, but nominal safe set computations are more brittle than our framework. This highlights the need for robust, reliable and online updated reachability sets, especially for complex high-dimensional systems. The complete results are summarized in Table I.

C. 3-link damped manipulator

We also consider a damped variant of a 3-link manipulator, whose dynamics are given by

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + B\dot{q} = u. \tag{8}$$

Here, B is the damping coefficient and the rest of the notations are identical to Equation 8. The experimental settings are exactly identical to the non-damped version described previously, except for the added damping. To emulate unmodelled, sim-to-real differences, we do not update the damping coefficient of our dynamics. The inaccuracy in dynamics arises from the mass of the last link m_3, as well as the dampling B. m_3 is updated online, while B is not.

For all experiments, we fix the inaccurate damping coefficient to be 0.4. In each trial, for the random scenarios, the true damping coefficient is sampled from a uniform distribution between $[0.3, 0.5]$ while in the challenging scenario, the true damping coefficient is sampled from a uniform distribution between $[0.3, 0.4]$. Our experiments show that even with such unmodelled disturbances, our approach can avoid obstacles 87% of the time in challenging scenarios with $\sigma = 0.3$. On the other hand, the nominal safe set only avoids the obstacle 64.5% of the time. This shows that our framework learns a robust representation of the safe set, that generalizes to unmodelled disturbances, and successfully avoids the danger zone. Note that the success rates are higher than those from 3-link manipulator without damping in the previous section because adding damping makes it easier to learn a good safe policy offline.

D. Quadrotor (8-dimensional system)

We also consider an 8-dimensional quadrotor dynamical system for our experiments. The states of the quadrotor are $[x, y, z, v_x, v_y, v_z, \theta, \phi]$, where x, y, z are the positions in x-y-z space, v_x, v_y, v_z are the velocities, and θ, ϕ are the roll and yaw angles. Assuming gravity g, the dynamics are:

$$\dot{q_x} = v_x, \quad \dot{q_y} = v_y, \quad \dot{q_z} = v_z \tag{9}$$

$$\dot{v}_x = gtan\theta, \quad \dot{v}_y = -gtan\phi \tag{10}$$

$$\dot{v}_z = \frac{u_z}{m} - g, \quad \dot{\theta} = \frac{u_\theta}{m}, \quad \dot{\phi} = \frac{u_\phi}{m}. \tag{11}$$

The input to the systems are forces u_θ, u_ϕ, u_z that directly affect the vertical and rotational accelerations of the quadrotor. The uncertainty in the dynamics arise from uncertainty over the mass m. The bounds on the control are: $|u_\theta|, |u_\phi| \leq 0.1$ and $u_z \in [g - 2.0, g + 2.0]$. The obstacle is a cube at $[q_x, q_y, q_z] = [0, 0, 0]$ with edge length 1.0.

The initialization of the states are described as follows:

	$[q_x, q_y, q_z]$	$[v_x, v_y, v_z]$
Random	$[0,0,0] + x_{rand}$	$[0,0,0] + dx_{rand}$
Challenging	$q_x = 0, q_y = 0, q_z = x_{chal}$	$[0,0,0] + dx_{chal}$

where x_{rand} and dx_{rand} are sampled from the uniform distribution within the range: x_{rand} : $[-2, 2]$m, dx_{rand} : $[-0.2, 0.2]$m/s, x_{chal} : $[0.55, 0.6]$m, and dx_{chal} : $[-0.1, 0]$m/s. For both random and challenging scenarios, the initial $[\theta, \phi]$ is always $[0, 0]$rad in the experiment.

Even though the quadrotor is higher dimensional than the manipulators, the control of the quadrotor dynamics is simpler. Hence, it is easier to find control policies that can successfully avoid the obstacle from most initial conditions. Both the nominal safe set and updated safe set have close to 100% success at avoiding the obstacle for random initialization. For the challenging scenario, we amplified the difficulty by applying a downward u_z when safe, instead of a random action. In this setting with $\sigma = 0.3$, our framework has a success rate of 95% while the baseline has a success rate of 85%, again displaying the robustness of our framework as compared to the nominal safe set. Results are summarized in Table I.

The compute time of updating the safe sets online for the different dynamical systems is shown in Table II, demonstrating that our framework is very fast at updating dynamics and corresponding safe sets.

E. Experiments with safe model-based RL

In this section, we present experimental results on applying our proposed method for maintaining safety under uncertain dynamics to model-based RL as described in section III-C. First we demonstrate qualitatively the benefit of incorporating safety in MBRL in Figure 5. When considering safety, the 2-link manipulator learns to apply actions that avoid the obstacle (red) while reaching the goal (green). On the other hand, standard MBRL without safety hits the obstacle while getting to the goal. Both approaches start at the same initial configuration, $[\theta_1, \theta_2, \dot{\theta}_1, \dot{\theta}_2] = [-\frac{\pi}{12}, \frac{13\pi}{20}, 0, 0]$. The obstacle is a square centered at $[x, y] = [0, 1.5]$ with side length 1.0 and the goal is a circle centered at $[x, y] = [1.5, 1.0]$ with radius 0.03. The uncertainty comes from the mass m_2 at the end-effector.

		2-link manipulator (4D)		3-link manipulator (6D)		3-link-damped (6D)		Quadrotor (8D)	
		$\sigma = 0.1$	$\sigma = 0.3$	$\sigma = 0.1$	$\sigma = 0.3$	$\sigma = 0.1$	$\sigma = 0.3$	$\sigma = 0.1$	$\sigma = 0.3$
Random	Nominal (Baseline)	99.5	97	93.5	91.5	97	98.5	100	99.5
	Our framework	100	98.5	93.5	96.5	98.5	97	100	98.5
Challenging	Nominal (Baseline)	93	85	35.5	27.5	80.5	64.5	89.5	85
	Our framework	99	99.5	53	39.5	93.5	87	93.5	95

TABLE I: In this table, we show comparisons of success rates between using our framework and the nominal safe set. For the scenario using completely random initialization, our framework performs slightly better than the baseline. This is due to the fact that with this initialization, the robot rarely gets to a situation where it's close to being unsafe. However, to test the robustness of our approach and the baseline, we consider random initialization around challenging scenarios. We can clearly see the performance benefit with using our method in these scenarios, especially for systems with complicated dynamics such as the 3-link robot arm. Here σ determines the range of values we sample the true dynamics parameter p_{true} from and is explained in detail in the text.

	2-link	3-link	Quadrotor
Compute time	0.17 s	0.25 s	0.10 s

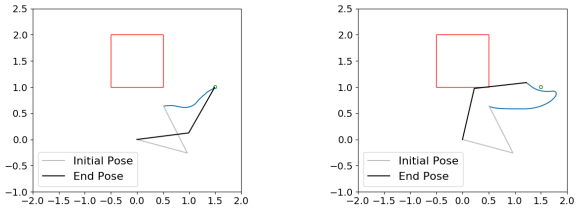
TABLE II: Average computation time for re-computing safety values at each time step. Even though the quadrotor has higher number of dimensions, its dynamics are much simpler than the those of the manipulators

To compare safe MBRL with standard MBRL, we ran a large scale experiment with 100 randomized runs to evaluate the effect of incorporating safety into MBRL. For each trial, the obstacle location and the initial state are identical to that presented in Figure 5. We randomize the goal location for each run. We set the maximum time steps allowed to reach the goal for both methods to be 300, with integration time-step $dt = 0.1$. For each run, both methods (with and without safety constraints) start with the same initial condition and have the same goal. A trial terminates early when the robot hits the obstacle or reaches the goal. For this set of experiments, the dynamics uncertainty comes from the mass at the end-effector. The true mass at the end-effector is $p = m_2 = 2.4$. For both approaches, we start with an initial belief of the uncertain dynamics parameter as a Gaussian distribution with mean 2.2 and standard deviation 0.1 and update the belief over time as MBRL runs. Note that both approaches update the dynamics parameter using BLR, as in Section III-B.3, but MBRL with safety explicitly reasons about safety using our framework when solving the optimization in Equation 3.

The results are summarized in table III. Completion rate indicates the percentage of trials in which the robot reaches the goal without hitting the obstacle, and collision rate refers to the numb er of trials with collision out of all runs. We can see that by incorporating safety in MBRL, the collision rate is 4% compared with 21% when not incorporating safety. Hence using MBRL with safety leads to a much safer learning process.

	With safety	Without safety
Completion rate	93	77
Collision rate	4	21

TABLE III: This table summarizes our large scale experimental results for incorporating safety into MBRL. Completion rate indicates the percentage of runs the robot reaches the goal without hitting the obstacle. We can see that incorporating safety in MBRL decreases the collision rate considerably. Note that the sum of completion rate and collision rate is not 100% because there are runs where the robot neither reaches the goal nor hits an obstacle as we set a limit on the number of time steps allowed for task completion.



(a) MBRL with safety (b) MBRL without safety

Fig. 5: With the same initial configuration (grey), MBRL with safety learns to reach the goal (green) without hitting the obstacle while MBRL without safety hits the obstacle while moving towards the goal. Without safety, the robot speeds towards the goal greedily and needs to turn around due to torque saturation. It ends up hitting the obstacle while with safe MBRL, the robot moves slowly and safely towards the goal.

VI. Conclusion and Future Work

In this work, we present a framework for offline training and online updates to safe sets in the context of Hamilton-Jacobi reachability analysis. We start by learning a robust safe policy by considering a distribution over dynamics. This is then used online to generate safe sets by rolling out the safe policy from a given state and current estimate of the dynamics. Simultaneously, we collect dynamics data to update our estimate of the dynamics parameters. This gives rise to a safe learning framework that can learn about its dynamics, and achieve a task, while maintaining safety. Our framework generalizes to high-dimensional systems, such as 3-link manipulators and quadrotors, and reliably avoids obstacles in challenging scenarios, where using a nominal safe set might fail. In addition, we demonstrate that the central idea of our framework can be used in combination with MBRL to learn to accomplish a task safely.

While our experimental results demonstrate that our framework is robust to uncertainties in dynamics, our experiments are conducted in simulation with added noise and unmodeled inaccuracy in the dynamics parameters. The next step is to study this approach on a real high-dimensional robot arm. In such cases, some of the modeling inaccuracy arises from inertial parameters, but there are also other sources of inaccuracy, such as state-dependent friction co-

efficient, which are not captured in our current setup. In the future, we would like to study more general dynamics and safety learning approaches that can generalize to such problems.

REFERENCES

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

[2] V. Rubies-Royo, D. Fridovich-Keil, S. Herbert, and C. J. Tomlin, "A classification-based approach for approximate reachability," in *2019 International Conference on Robotics and Automation (ICRA)*, May 2019, pp. 7697–7704.

[3] 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," *arXiv preprint arXiv:1905.00532*, 2019.

[4] J. F. Fisac, A. K. Akametalu, M. N. Zeilinger, S. Kaynama, J. Gillula, and C. J. Tomlin, "A general safety framework for learning-based control in uncertain robotic systems," *IEEE Transactions on Automatic Control*, vol. 64, no. 7, pp. 2737–2752, July 2019.

[5] J. F. Fisac, N. F. Lugovoy, V. Rubies-Royo, S. Ghosh, and C. J. Tomlin, "Bridging hamilton-jacobi safety analysis and reinforcement learning," in *2019 International Conference on Robotics and Automation (ICRA)*, May 2019, pp. 8550–8556.

[6] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[7] L. Wang, E. A. Theodorou, and M. Egerstedt, "Safe learning of quadrotor dynamics using barrier certificates," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 2460–2465.

[8] R. Cheng, G. Orosz, R. M. Murray, and J. W. Burdick, "End-to-end safe reinforcement learning through barrier functions for safety-critical continuous control tasks," *CoRR*, vol. abs/1903.08792, 2019. [Online]. Available: http://arxiv.org/abs/1903.08792

[9] K. P. Wabersich and M. N. Zeilinger, "Linear model predictive safety certification for learning-based control," *CoRR*, vol. abs/1803.08552, 2018. [Online]. Available: http://arxiv.org/abs/1803.08552

[10] T. Koller, F. Berkenkamp, M. Turchetta, and A. Krause, "Learning-based model predictive control for safe exploration," in *2018 IEEE Conference on Decision and Control (CDC)*, 2018, pp. 6059–6066.

[11] F. Berkenkamp, M. Turchetta, A. P. Schoellig, and A. Krause, "Safe model-based reinforcement learning with stability guarantees," in *Proc. of Neural Information Processing Systems (NeurIPS)*, 2017. [Online]. Available: https://arxiv.org/abs/1705.08551

[12] Y. Chow, O. Nachum, E. Duenez-Guzman, and M. Ghavamzadeh, "A lyapunov-based approach to safe reinforcement learning," in *Advances in Neural Information Processing Systems*, 2018, pp. 8092–8101.

[13] Y. Chow, O. Nachum, A. Faust, M. Ghavamzadeh, and E. Duenez-Guzman, "Lyapunov-based safe policy optimization for continuous control," *arXiv preprint arXiv:1901.10031*, 2019.

[14] M. Chen, S. L. Herbert, M. S. Vashishtha, S. Bansal, and C. J. Tomlin, "Decomposition of reachable sets and tubes for a class of nonlinear systems," *IEEE Transactions on Automatic Control*, vol. 63, no. 11, pp. 3675–3688, Nov 2018.

[15] A. Dhinakaran, M. Chen, G. Chou, J. C. Shih, and C. J. Tomlin, "A hybrid framework for multi-vehicle collision avoidance," in *56th IEEE Annual Conference on Decision and Control, CDC 2017, Melbourne, Australia, December 12-15, 2017*, 2017, pp. 2979–2984. [Online]. Available: https://doi.org/10.1109/CDC.2017.8264092

[16] M. Chen, J. C. Shih, and C. J. Tomlin, "Multi-vehicle collision avoidance via hamilton-jacobi reachability and mixed integer programming," in *55th IEEE Conference on Decision and Control, CDC 2016, Las Vegas, NV, USA, December 12-14, 2016*, 2016, pp. 1695–1700. [Online]. Available: https://doi.org/10.1109/CDC.2016.7798509

[17] S. Bansal, M. Chen, J. F. Fisac, and C. J. Tomlin, "Safe sequential path planning under disturbances and imperfect information," in *2017 American Control Conference (ACC)*, May 2017, pp. 5550–5555.

[18] S. Bansal, M. Chen, S. Herbert, and C. J. Tomlin, "Hamilton-Jacobi reachability: A brief overview and recent advances," 2017.

[19] K. Chua, R. Calandra, R. McAllister, and S. Levine, "Deep reinforcement learning in a handful of trials using probabilistic dynamics models," in *Advances in Neural Information Processing Systems*, 2018, pp. 4754–4765.

[20] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," 2017.

[21] C. H. An, C. G. Atkeson, and J. M. Hollerbach, "Estimation of inertial parameters of rigid body links of manipulators," in *1985 24th IEEE Conference on Decision and Control*. IEEE, 1985, pp. 990–995.

[22] J.-A. Ting, A. D'Souza, and S. Schaal, "Bayesian robot system identification with input and output noise," *Neural Networks*, vol. 24, no. 1, pp. 99–108, 2011.

[23] C. M. Bishop, *Pattern Recognition and Machine Learning*. New York: Springer, 2007.

[24] T. Koller, F. Berkenkamp, M. Turchetta, J. Boedecker, and A. Krause, "Learning-based model predictive control for safe exploration and reinforcement learning," 2019.

[25] D. Yarats, A. Zhang, I. Kostrikov, B. Amos, J. Pineau, and R. Fergus, "Improving sample efficiency in model-free reinforcement learning from images," *arXiv preprint arXiv:1910.01741*, 2019.

[26] *Kuka LBR iiwa*. [Online]. Available: https://www.kuka.com/en-us/products/robotics-systems/industrial-robots/lbr-iiwa