2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 25-29, 2020, Las Vegas, NV, USA (Virtual)

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 the robot 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 robots or their 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 safe sets and an action policy that can together ensure that the system does not enter a danger 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 safe set computation 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 sets 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 safe sets based on the dynamics of the robot online. [3] addresses 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 of links. This is a common source of uncertainty in robots, such as manipulators, but the process of identifying the uncertain parameters with learning 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 [4]. Online, we start with an initial belief of the distribution of the dynamics parameters, update it as we gather more data, and re-compute the safe set by forward simulating the safe policy based on the new belief of the dynamics parameters. As a result, the robot is not overly conservative during online execution, while maintaining a high level of safety. Figure 1 provides a high level overview of our approach.

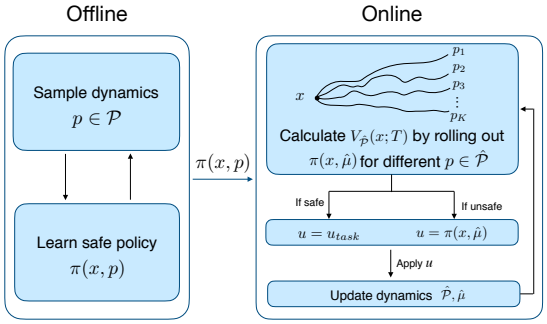


Fig. 1: An outline of our approach. Offline, we learn a safe policy $\pi(x, p)$, which depends on state x and dynamics parameters p, by randomly sampling different dynamics parameters p. Online, we roll out $\pi(x, \hat{\mu})$ using the current estimate of the dynamics, and use it to determine the safety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

The central ideas of our framework can also be directly used with learning approaches with optimization components, like model-based RL [5]. We demonstrate incorporating safety derived from our framework in a model-based RL setting, by adding a safety constraint to the optimization problem, thereby proposing a safe model-based RL setup where the task policy takes safety into account.

Our work is a step towards online updates and sim-to-real transfer of safe sets for high-dimensional systems. We test the efficacy of our proposed framework at avoiding obstacles during random exploration on an 8-dimensional quadrotor

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

and a 3-link manipulator (6-dimensional state 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 test incorporating safe sets learned from our framework in a model-based RL setting on a 2-link manipulator, and show that this results in safer learning when completing a task, compared with standard model-based RL. Our results showcase the robustness of our framework and open 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 or encourage safe learning. Constrained optimization, for example with barrier functions, can be used to certify probabilistic safety [6] or guide exploration to safe regions [7]. 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 [8], [9]. However, safety computations in these works are either applied to linear systems or local 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 [10], [11], [12] use Lyapunov functions to guarantee or encourage safety during learning. However, [10] requires a Lyapunov function for a given system, [11] is limited to discrete action spaces, and [12] 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 on addressing safety with HJ reachability [1], which is a control-theoretic framework that provides safety guarantees for general dynamical systems. There has been a lot of work on guaranteeing safety of single-agent and multi-agent systems using reachability under varying assumptions on the dynamics and agent formation, such as [13], [14], [15], [16]. [17] 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 $x \in \mathcal{X}$, 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.

Let $\mathcal{Z}$ represent the danger zone, which is the set of states that the robot should avoid. We can define $\mathcal{Z}$ by using a level set function $l(x)$ such that $l(x) \leq 0$ if and only if $x \in \mathcal{Z}$.

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

$$V(x) = \sup_{\boldsymbol{u}(\cdot)} \inf_{t \geq 0} l(\xi_x^{\boldsymbol{u}}(t)).$$

Intuitively, $V(x)$ is the closest the trajectory gets to the danger zone $\mathcal{Z}$ given the *best possible control* to avoid the danger zone. The safe set $\mathcal{K}$ is defined as $\mathcal{K} = \{x : V(x) > 0\}$ and the unsafe set is then the complement of $\mathcal{K}$. Intuitively, the safe set $\mathcal{K}$ 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 [4] for approximating the *best possible control* with reinforcement learning (RL), which typically adopts a discrete-time formulation. Hence in this paper, we work with the discrete-time formulation of reachability where in the infinite-horizon scenario, the value function $V(x)$ is defined as

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

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 $\mathcal{K} = \{x : V(x) > \epsilon\}$ in order to ensure safety. We adopt this convention in this paper.

B. Learning to approximate safe sets from HJ reachability

It is intractable to compute exact safe sets for dynamical systems with more than five states, due to discretization of the state space during computation [18], and the resulting curse of dimensionality. As a result, there have been some works on approximating safe sets for high-dimensional problems.

[2] proposes to use function approximators to represent the optimal safe policy for control-affine systems, which in turn generates safe sets. [4] proposes 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. Note that there exists a 2-player formulation of reachability [1] where bounded disturbance is considered. However, no effective approach has been proposed for approximating two-player reachability for general high dimensional systems.

In practice, we found that the method proposed by [4] 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 robot to think it's safe while it's not. We adapt this approach to learn a safe policy under dynamics uncertainty. Next, we use the learned safe policy to compute safe sets

online, while updating the estimate of the dynamics from data.

C. Online updates to safe sets from HJ reachability in robotics problems

In this section, we give an overview of papers that address online updates to safe sets in the presence of uncertainties in dynamics or the environment. [19] proposes to use local updates and warm-start to generate safe sets online, as static obstacles in the environment are detected. However, the proposed methods still rely on discretization of the state space and is hence intractable for high-dimensional systems.

[3] uses a Gaussian process (GP) to model uncertainty in the dynamics of the system, followed by applying standard HJ reachability to compute safe sets based on the estimated dynamics. However, this also relies on online re-computation of safe sets through discretization of the state space, making it inapplicable to 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 uncertainty. We assume inaccuracies 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 [20], even on hardware [21].

Model-based reinforcement learning (MBRL) iteratively tries to optimize a policy to accomplish a task, and learns the dynamics of the robot. Similar to [20], 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 $u_{t:t+H-1} \equiv \{u_t, \ldots, u_{t+H-1}\}$ over a horizon H from current the time step t subject to dynamics constraint. After the MPC problem is solved, the first action u_t is applied to the system, and the process repeats, starting with the new current state.

We present an application of online safe sets derived from our proposed framework in model-based RL in Section III-C.

III. Framework for safe set computations for uncertain dynamics

In this section, 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 the safe policy

Algorithm 1: Online loop of proposed framework

Given : safe 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 $s_t > \epsilon$ **then**
    $u_t =$ any action (can be an action from a learning controller or a random action)
  else
    $u_t = \pi(x_t, \hat{\mu}_t)$
  end
  $x_{t+1} = f(x_t, u_t; p_{true})$;
  Update belief of dynamics parameters with (x_t, u_t, x_{t+1}) and compute $\hat{\mathcal{P}}_{t+1}, \hat{\mu}_{t+1}$;
  $t \leftarrow t + 1$;
end

$\pi(x, \hat{\mu})$ from the current state x, using our current best estimate of the dynamics parameters $\hat{\mu}$. This computes an approximate safety value $V(x)$ based on our belief about the dynamics distribution. If $V(x) > \epsilon$, the robot is in the safe set, and can apply any task-specific actions. Otherwise, the robot applies the safe action derived from the safe policy to avoid entering the danger zone. The resulting state transition on the robot is then used to update the belief about dynamics parameters p.

A. Offline computation of safe policy

During offline computation, we use reinforcement learning to train a safe policy $\pi(x, p)$ which is a function of both the state x and the dynamics parameters p. We assume that we know where the danger zone $\mathcal{Z}$ is during offline training. Every N episodes, we sample new dynamics parameters p and use them to collect data to train $\pi(x, p)$. This data is generated by repeatedly sampling the "true" dynamics parameters from the set $\mathcal{P} = [p - dp, p + dp]$ for some positive dp for the dynamics simulator. 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 determined based on the predicted uncertainty on the dynamics parameters during test time.

To train $\pi(x, p)$, we adapt the update rule from [4] to suit our purposes and use RL algorithms such as Soft Actor-Critic [22] or Q-learning to train the safe policy $\pi(x, p)$. The update rule of the Q-function $Q(x, p, u)$ we use during RL training is

$$Q(x,p,u) \leftarrow (1-\gamma) l(x) + \gamma \min\left\{ l(x), \max_{u' \in \mathcal{U}} Q(x + f(x,u;p)\Delta t, p, u') \right\},$$

where $f(x, u; p)$ is the dynamics of the robot with dynamics parameters p and γ is the discount factor.

B. Online updates to safe set

Given the safe policy $\pi(x, p)$ learned offline, we can generate safe sets online based on our estimate of the

distribution of the dynamics parameters p. We iteratively perform the following steps at every time step t: (1) Compute the safety value $V_{\hat{\mathcal{P}}_t}(x_t)$ of the current state x_t based on 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 above the safety threshold ϵ. (3) Update the estimate of the distribution of p using the new data gathered.

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

1) Computing the safe set online: At any given time step t, we maintain an estimate of $\hat{\mathcal{P}}_t$, a set that dynamics parameters p fall in with some 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 $p \sim \mathcal{N}(\hat{\mu}_t, \hat{\sigma}_t^2)$ and $c = 0.95$, then $\hat{\mathcal{P}}_t = [\hat{\mu}_t - 1.96\hat{\sigma}_t, \hat{\mu}_t + 1.96\hat{\sigma}_t]$. $\hat{\mathcal{P}}_t$ 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 for time horizon T. To compute $V(x; p, T)$, we roll out the safe policy $\pi(x, \hat{\mu}_t)$ from state x for horizon T. By denoting the resulting trajectory from the roll-out as $\xi_{x,p}^{\pi_{\hat{\mu}_t}, T}(\cdot)$, 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, for a horizon of T. At any time t, we can compute the safety values at all states x in the space and form a safe 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 $V(x_t; p, T)$ over all possible dynamics parameters in $\hat{\mathcal{P}}_t$, the robot uses the safe policy if *any* dynamics parameter in $\hat{\mathcal{P}}_t$ results in $V(x_t; p, T)$ less than or equal to 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 the 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 dynamics parameters in $\hat{\mathcal{P}}_t$ in parallel.

Given our proposed approach, we now formally present a proof showing that the safe sets computed using our proposed framework are conservative under specific conditions.

Theorem 1: Assume $\hat{\mathcal{P}}_t$ is a discrete set. For any time t, state x and horizon T, if the true dynamics parameter $p_{true} \in \hat{\mathcal{P}}_t$, then $V_{\hat{\mathcal{P}}_t}(x; T) \leq V^\star(x; p_{true}, T)$ where $V^\star(x; p_{true}, T)$ is the true safety value with p_{true} as the true dynamics parameters.

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 presented 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. For cases where dynamics parameters are drawn from a continuous set, we discretize the dynamics parameters set $\hat{\mathcal{P}}_t$ during forward roll-out. As a result, we lose guaranteed conservatism, but we still demonstrate empirically that using our proposed approach is safer than using the nominal safe sets.

2) Determining the action to take: In the case of HJ reachability with continuous dynamics [1], as long as the optimal safe control policy is applied immediately when the safety value is smaller or equal ϵ, safety of the robot is guaranteed. Although we are working with discrete dynamics, this decision rule still provides a good criterion for selecting whether or not to execute the safe policy. When the safety value is above ϵ, i.e., $V_{\hat{\mathcal{P}}_t}(x_t; T) > \epsilon$, the robot is determined safe and it can apply any action, such as an action determined by any learning controller. When $V_{\hat{\mathcal{P}}_t}(x_t; T) \leq \epsilon$, the robot is deemed unsafe and it applies the action determined by the safe 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 for robots is widely studied in robotics, such as in [23], [24]. In the experiments for this paper, we model the uncertain dynamics parameters as a Gaussian distribution $\mathcal{N}(\mu, \Sigma)$ where μ and Σ are the mean and covariance matrix of the Gaussian respectively. We use Bayesian Linear Regression (BLR) to update our belief of μ and Σ, and in turn use them to update $\hat{\mu}_t$ and $\hat{\mathcal{P}}_t$. Interested readers can consult Chapter 3 in [25] for details about Bayesian Linear Regression. This gives us an online and 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 [24].

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, the current best estimate $\hat{\mu}_t$ of the parameters and the set $\hat{\mathcal{P}}_t$ that dynamics parameters fall in with high probability, the safe model-based RL optimization problem becomes

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

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

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

where $f(x_h, u_h; \hat{\mu}_t)$ is the discrete dynamics function assuming the dynamics paramters are $\hat{\mu}_t$ and cost $c(x_h, u_h)$ is determined based on the task, such as getting the robot to a goal. After u_t is applied on the robot, the (state, action, next state) transition is used to update our belief about the dynamics. The current state is then updated to the new state, and the process repeats. If no feasible action sequence can be found, the action $\pi(x_t, \hat{\mu}_t)$ from the safe policy is applied. [26] proposes a related setup for model-predictive control, with ellipsoidal safe sets computed with linearized dynamics. In contrast, our approach incorporates safety directly using the original dynamics.

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 variables, actions $u_{t:t+H-1}$. This can lead to poorer solutions or high computation time when solving the optimization problem for a large H. On the other hand, T does not affect the optimization dimension, and can be much larger than H. This ensures that even though our model-based RL algorithm has limited foresight for task planning, the safety constraint enforces a longer horizon plan for safety. In our experiments, we use random sampling in the space of actions to solve the optimization problem.

IV. 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 applying safe sets computed 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.

A. Comparison between our proposed framework and using nominal safe sets

We experiment with two different scenarios, random and challenging. For the random scenario, we initialize the robot randomly at states that are safe; for the challenging scenario, we initialize the robot at safe states closer to the obstacles. We observe that the robot rarely gets close to the unsafe states with random initialization. The initialization in the challenging scenario increases the frequency the robot gets close to the unsafe states. In all trials across different scenarios and methods, the nominal dynamics parameter is always $\hat{p}_0 = 2$. For the random scenario, the true dynamics is sampled from the set $p_{true} \sim \text{uniform}[\hat{p}_0 - 2\sigma, \hat{p}_0 + 2\sigma]$ and for the difficult scenario, the true dynamics is sampled from $p_{true} \sim \text{uniform}[\hat{p}_0 - 2\sigma, \hat{p}_0]$, for $\sigma = 0.1, 0.3$. In general, our method shines in conditions that are tough. For safety, it is important to avoid obstacles in all scenarios, and the challenging scenario showcases the robustness of our approach versus using the nominal safe set.

We simulate 200 trials for various dynamical systems. Each trial lasts for $T = 100$ time steps. A trial is successful if the robot does not hit any obstacle throughout the entire trial and unsuccessful otherwise. In each trial, we apply a uniform random action, if it is determined that the robot is safe, and apply the safe policy otherwise. We compare the success rates of our framework with that of using nominal safe sets with inaccurate dynamics. We use Soft Actor-Critic [22] with implementation from [27] to train the safe policy $\pi(x, p)$ for all dynamical systems.

Both methods start from the same initial safe condition. To emulate sim-to-real differences, we add a 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 (Section III-B.3), but our proposed method consistently outperforms the baseline in challenging scenarios. In addition, for the 3-link manipulator, we also perform an extensive experiment where our framework assumes a damping coefficient different from the true damping coefficient, without updating our belief about this coefficient. This further shows the robustness of our approach in situations that violate the assumptions of our proposed framework.

The complete experimental results are summarized in Table I.

1) 2-link manipulator: We consider the task of safely controlling a 2-link manipulator in the x-y plane. In general, the dynamics of a manipulator are:

$$M(q)\ddot{q} + C(q, \dot{q})\dot{q} = u. \quad (6)$$

For a 2-link manipulator, $q = [\theta_1, \theta_2]$ 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, which is constrained to $|\tau_1|, |\tau_2| \leq 1$. While we did not consider gravity in the dynamics in Equation 6, these dynamics generalize to manipulators with manufacturer-provided gravity compensation, such as Kuka LBR [28]. We use simplified dynamics with masses at the end of each link (each of length $1.0\,\mathrm{m}$) with the mass at the end of the first link being $m_1 = 1.0\,\mathrm{kg}$. The uncertainty in the dynamics comes from the uncertainty in the mass at the end effector $p = m_2$. In all experiments for manipulators, the danger zone in the environment is a square obstacle centered at $(x, y) = (0, 1.5)\mathrm{m}$ with edge length $1.0\mathrm{m}$.

The initialization of the states is described as follows:

	$[\theta_1, \theta_2]$	$[\dot{\theta_1}, \dot{\theta_2}]$
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 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 2 visualizes the state that we sample around for the challenging scenario. Note that if the sampled

initial state is unsafe, we re-sample until the initial state is inside the safe set.

As summarized in Table I, in the random scenario, 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, we see that online updates to safe sets can considerably improve the rate of success for avoiding obstacles.

2) 3-link manipulator: The dynamics of a 3-link manipulator can also be described by Equation 6. 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, which is constrained by $|\tau_1|\,|\tau_2|\,, |\tau_3| \leq 1$. Note that computing the safe sets for this 6-dimensional system is intractable with standard reachability [1]. Similar to the 2-link manipulator, we consider a 3-link manipulator with mass at the end of each link where the length of each link is $1.0\,\mathrm{m}$. We denote the mass at the end of first, second, and third link as m_1, m_2, and m_3 where m_1 =$1.0\,\mathrm{kg}$, m_2 =$2.0\,\mathrm{kg}$. The uncertainty in the dynamics comes from the uncertainty in the mass at the end effector $p = m_3$.

The initialization of the states is 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 2 visualizes the state that we sample around for the challenging scenario

In the random scenario, we observe comparable performance between our framework and 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% with the baseline for $\sigma = 0.1$. The performance of both approaches gets worse as the complexity of the dynamics increase, but the nominal safe sets are more brittle than our framework. This highlights the need for robust, reliable, and online updated safe sets, especially for complex high-dimensional systems.

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

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

Here, B is the damping coefficient and the rest of the notations are identical to those of Equation 6. The experimental settings are identical to those of the non-damped

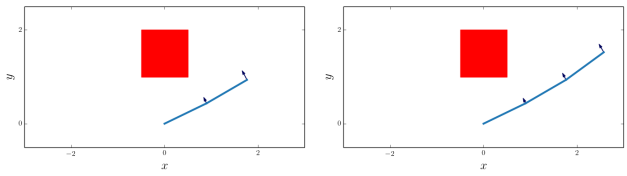


Fig. 2: Environments and the challenging initial conditions that we randomize around for the 2-link and 3-link manipulators experiments. The arrows represent the velocities at the joints and the end effectors. The red squares represent the obstacles.

version described previously, except for the added damping. The uncertainty in the dynamics arises from the mass $p = m_3$ at the end effector, as well as the damping coefficient B. To emulate unmodelled sim-to-real differences, we do not update our belief about the damping coefficient B online. We only update our belief about m_3 online.

For all experiments, we fix the inaccurate damping coefficient to be 0.4. The true damping coefficients are sampled from a uniform distribution within the ranges $[0.3, 0.5]$ and $[0.3, 0.4]$ for the random and challenging scenarios respectively. Our experiments show that even with such unmodelled disturbances, with our approach, the robot avoids obstacles 87% of the time with $\sigma = 0.3$ in the challenging scenario. On the other hand, with the nominal safe set, the robot only avoids the obstacle 64.5% of the time. This shows that our framework generates robust safe sets that generalize to unmodelled disturbances, enabling the robot to avoid the danger zone reliably. 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 reliable safe policy offline.

4) 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 the x-y-z space, v_x, v_y, v_z are the velocities, and θ, ϕ are the roll and yaw angles. Denoting gravity as g, the dynamics are:

$$\dot{q_x} = v_x, \quad \dot{q_y} = v_y, \quad \dot{q_z} = v_z \tag{8}$$

$$\dot{v}_x = gtan\theta, \quad \dot{v}_y = -gtan\phi \tag{9}$$

$$\dot{v}_z = \frac{u_z}{m} - g, \quad \dot{\theta} = \frac{u_\theta}{m}, \quad \dot{\phi} = \frac{u_\phi}{m}. \tag{10}$$

The input to the systems are forces u_θ, u_ϕ, u_z that directly affect the vertical and angular accelerations of the quadrotor. The uncertainty in the dynamics arises from uncertainty in 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 centered at $[q_x, q_y, q_z] = [0, 0, 0]$ with edge length $1.0\,\mathrm{m}$.

The initialization of the states is 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 set to $[0, 0]$rad.

		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 sets. For the random scenario, our framework performs slightly better than the baseline. This is due to the fact that with the initialization scheme in the random scenario, 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 initialization that is challenging. We can clearly see the performance benefit of our framework in the challenging scenario, 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 (Section IV-A).

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

TABLE II: Average computation time for updating the dynamics and re-computing safety values at each time step. The compute time for the quadrotor is smaller because even though the state of the quadrotor has a larger dimension, its dynamics are much simpler than those of the manipulators.

Even though the quadrotor has higher dimensions than the manipulators, it has simpler dynamics, making it easier to learn a good safe policy. Both using the the nominal safe sets and our framework result in close to 100% success rates for avoiding the obstacle for the random scenario. 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 demonstrating the robustness of our framework compared to the nominal safe sets.

The compute time online for all dynamical systems considered is shown in Table II, demonstrating that our framework is fast at updating dynamics and safe sets.

B. Experiments with safe model-based RL

In this section, we present experimental results on applying our proposed framework for enhancing safety under uncertain dynamics to model-based RL as described in Section III-C. We consider a 2-link manipulator identical to that used in the previous experiment IV-A.1 and the uncertainty similarly comes from the mass m_2 at the end effector. The objective is to get the end-effector of the manipulator to some target location while avoid hitting the obstacle.

We compare safe MBRL with standard MBRL without safety. 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. Figure 3 demonstrates qualitatively the benefit of incorporating safety in MBRL. 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, with standard MBRL without safety, the robot hits the obstacle while trying to get to the goal.

To compare safe MBRL with standard MBRL quantitatively, we also ran an experiment with 100 randomized runs to evaluate the effect of incorporating safety into MBRL.

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

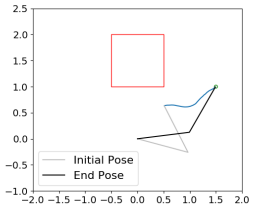
TABLE III: This table summarizes our large scale experiment results for incorporating safety into MBRL. Completion rate indicates the percentage of trials the robot reaches the goal without hitting the obstacle within the maximum time steps allowed for task completion. Collision rate refers to the percentage of trials the robot hits the obstacle. We can see that incorporating safety in MBRL increases the success rate and decreases the collision rate considerably.

For each trial, the obstacle location and the initial state are identical to those presented in Figure 3 and the goal location is randomized for each run. We set the maximum time steps allowed to reach the goal for both methods to be 300, with integration time-step $\Delta t = 0.1$. 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. The true mass at the end effector is $p = m_2 = 2.4$. Note that both approaches update the dynamics parameter using BLR, described in Section III-B.3, but MBRL with safety explicitly reasons about safety using our framework when solving the optimization problem 3. For both approaches, we start with an initial belief of the uncertain dynamics parameter p as a Gaussian distribution with mean 2.2 and standard deviation 0.1 and update the belief over time as MBRL runs.

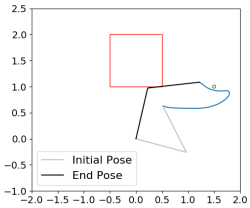
The results are summarized in table III. Completion rate indicates the percentage of trials in which the robot reaches the goal without hitting the obstacle, and collision rate refers to the percentage of trials where the robot hits the obstacle. We can see that by incorporating safety in MBRL, the collision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

V. 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. 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 states and current estimate of the dynamics. Simultaneously, we collect dynamics data to update our



(a) MBRL with safety



(b) MBRL without safety

Fig. 3: With the same initial configuration (grey), MBRL with safety learns to reach the goal (green) without hitting the obstacle (red) while MBRL without safety hits the obstacle. The blue curves illustrate the trajectories of the end effector. Without safety, the robot starts out speeding towards the goal greedily, turns around, and speeds to the goal again. Due to torque saturation, it misses the goal then ends up hitting the obstacle. On the other hand, with safe MBRL, the robot moves slowly and safely towards the goal.

belief about the dynamics parameters. This gives rise to a safe learning framework that allows robots to learn about its dynamics and achieve a task with reliable safety. Our framework generalizes to high-dimensional systems, such as 3-link manipulators and quadrotors, and reliably avoids obstacles in challenging scenarios where using nominal safe sets might fail. In addition, we demonstrate that the central idea of our framework can be used in combination with MBRL to have robots learn to accomplish tasks 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 may also be other sources of inaccuracy, such as state-dependent friction coefficient, which are not captured in our current setup. In the future, we aim to develop safety approaches that consider more general dynamics uncertainty and require less computation online.

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] 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.

[4] 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.

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

[6] 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.

[7] 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

[8] 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

[9] 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.

[10] 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

[11] 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.

[12] 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.

[13] 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.

[14] 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

[15] 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

[16] 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.

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

[18] I. M. Mitchell, "A Toolbox of Level Set Methods." *UBC Department of Computer Science Technical Report TR-2007-11 (June 2007)*.

[19] 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.

[20] 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.

[21] S. Bechtle, Y. Lin, A. Rai, L. Righetti, and F. Meier, "Curious ilqr: Resolving uncertainty in model-based rl," in *Conference on Robot Learning*, 2020, pp. 162–171.

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

[23] 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.

[24] 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.

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

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

[27] 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.

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