Conformal Predictive Safety Filter for RL Controllers in Dynamic Environments

Kegan J. Strawn[1], Nora Ayanian[2], and Lars Lindemann[1]

***Abstract*— The interest in using reinforcement learning (RL) controllers in safety-critical applications such as robot navigation around pedestrians motivates the development of additional safety mechanisms. Running RL-enabled systems among uncertain dynamic agents may result in high counts of collisions and failures to reach the goal. The system could be safer if the pre-trained RL policy was uncertainty-informed. For that reason, we propose *conformal predictive safety filters* that: 1) predict the other agents' trajectories, 2) use statistical techniques to provide uncertainty intervals around these predictions, and 3) learn an additional safety filter that closely follows the RL controller but avoids the uncertainty intervals. We use conformal prediction to learn uncertainty-informed predictive safety filters, which make no assumptions about the agents' distribution. The framework is modular and outperforms the existing controllers in simulation. We demonstrate our approach with multiple experiments in a collision avoidance gym environment and show that our approach minimizes the number of collisions without making overly conservative predictions.**

I. Introduction

While impressive progress has been made in deep reinforcement learning (RL) for motion planning, it remains challenging to ensure the safety of RL policies [1]–[5]. Developing safe policies for dynamical systems that operate around dynamic agents is a fundamental challenge in robotics. Other agents' intents and policies are usually unknown yet critical to any safe motion planning algorithm. An increasingly popular approach is to use RL to learn a reactive collision avoidance policy [2]. However, such policies alone do not quantify uncertainty in a principled way, provide no safety guarantees, and have been observed to be unsafe. In these scenarios, when the RL policy is not guaranteed to be safe, a predictive safety filter is desirable that considers the uncertainty about the dynamic agents' future motion. Such a safety filter will ideally guarantee safety probabilistically while being minimally invasive, i.e., following the RL policy closely. Another motivation for our work is that RL policies may have been trained in different training environments or under different objectives, while a safety filter can be trained independently.

In this paper, we learn a predictive safety filter for a given RL policy that uses predicted agent trajectories, e.g., from long short-term memory networks [6], [7] or transformer architectures [8], along with uncertainty intervals around each prediction that are obtained using conformal prediction [9]. The result is an uncertainty-informed predictive safety filter. Choosing to use conformal prediction, a distribution-free statistical tool [10]–[14], we do not make any assumptions about the distribution of the agent trajectories, e.g., being Gaussian distributed. We highlight our main contributions in this paper as such:

- We propose an algorithm to train uncertainty-informed predictive safety filters for pre-trained RL controllers using conformal prediction. The filter ensures probabilistic safety and incentivizes imitating the RL policy.
- We evaluate our method in a widely-used RL collision avoidance simulator in which we reduce collisions when paired with an RL controller [2] by 80%, reduce failures (where the agent timed out) when paired with a more conservative controller [15] by 67%, and produce shorter paths when compared to a Gaussian-based safety approach by 18%.

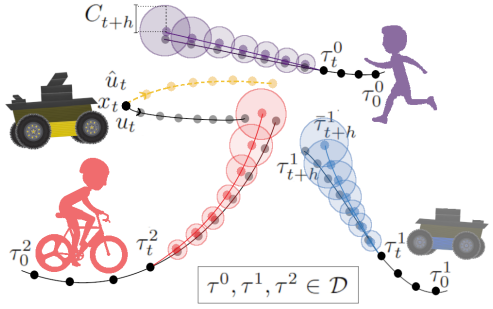

Fig. 1. The system applies an RL policy π that may lead to a collision with the other agents. Our proposed predictive safety filter $\hat{\pi}$ uses predictions of the three nearby agents (point dots) along with uncertainty intervals (shaded circles) to obtain a series of future actions that avoids collisions.

II. Related Work

Planning in dynamic environments: Model Predictive Control (MPC) is a popular planning approach that selects a minimum-cost action sequence using predictions of the agents conditioned on the current state and the history seen so far [16]. Actions are implemented in a receding horizon fashion where only the first action is applied before new sensor measurements are obtained, and the process is repeated. Reactive-based methods use geometric or physics-based rules to ensure collision avoidance on each step and rely on a fast update rate to react to changes in the other agents' motions [15], [17], [18]. Reactive-based RL

This work was supported in part by NSF awards CNS-1837779 and IIS-2311967.

[1]Kegan J. Strawn and Lars Lindemann are with the Thomas Lord Department of Computer Science, University of Southern California, Los Angeles, CA 90089, USA. kegan.j.strawn@usc.edu

[2]Nora Ayanian is with the Department of Computer Science and School of Engineering, Brown University, Providence, RI 02912, USA. nora_ayanian@brown.edu

controllers are computationally efficient but often generate sub-optimal trajectories [3]–[5]. Predictive-based methods first estimate the trajectories of other agents and then plan the system's actions. These methods often yield a smoother plan but are more computationally expensive and may require additional knowledge about the other agents. It is important to note that predicted paths can be too conservative or inaccurate and block the agent's path, potentially slowing down or deadlocking all agents, known as the freezing robot problem [19]. Some predictive and behavior-based planners mimic what a human (or another robot) would do, require expert demonstrations, estimate the cost functions of other agents, or perform some form of additional work to understand the intents of other agents [20]–[26]. Learning-based controllers attempt to predict the trajectories of pedestrians that could be used in collision avoidance systems or predict the reaction-based next step. An example from recent work learns from complex and multi-modal distributions of agents' motions and predictions in real-time for planning [27]. Other efforts look to integrate machine learning and model predictive control under uncertainty [28], [29].

Uncertainty Quantification: We use conformal prediction (CP), see [9], for quantifying the uncertainty of trajectory predictions [30], [31]. Alternative methods model the underlying distribution as a Gaussian distribution and use Kalman filters or apply methods for finding safe sets such as forward and backward reachability [32]–[37]. These alternatives can be helpful but are often overly conservative, computationally complex, or make unrealistic assumptions. Similar to CP, a Bayesian framework provides probabilistic uncertainty quantification but requires access to prior knowledge about the distribution the data is sampled from, and probably approximately correct (PAC) learning theory can be used for producing upper bounds on the probability of error for a given algorithm and confidence level, but the results often involve large constants for the overall algorithmic error [38].

CP provides distribution-free uncertainty quantification in such scenarios and has generally been used to quantify the uncertainty of machine learning models [9], [39]–[41]. CP is an increasingly popular approach to obtain guarantees on a predictor's false negative rate, estimate reachable sets, and design model predictive controllers with safety guarantees [10]–[14]. Recently, CP methods have been integrated with policy training for safety [42], [43], time series forecasting [31] and MPC of robots in dynamic environments [30], [44]. Another work combines conformal prediction with reachable sets for efficient and safe MPC [45]. These approaches integrate CP into an MPC, which reduces the framework's modularity and cannot be directly applied to learning-based controllers.

Safety Filters: Most safety techniques in RL constrain the search during training updates, train with noise or adversarial agents, restrict the inputs to the policy, or attempt to learn the uncertainty of the entire system [29], [46]–[49]. Work exists in the safe exploration of action spaces, such as filtering unsafe actions to prevent immediate negative consequences [50]. The training safety methods have been shown to perform negligibly better than non-safe RL methods, and many robotic applications may prevent re-training, necessitating safety methods for pre-trained RL controllers [50]–[53]. While there could be an advantage in using CP during the training of the RL policy, we design in this paper a predictive safety filter for a pre-trained RL policy to enable the use of high-performance, off-the-shelf controllers and to increase generalizability to different applications as well as provide stronger safety guarantees.

First introduced in [54], using safety filters for a controller in a closed-loop system has seen continued growth. Predictive safety filters assess if a proposed learning-based control input can lead to constraint violations and modify it if necessary to improve safety for future time steps. Many techniques exist, such as control barrier functions for verifying and enforcing system safety [35], [55]–[57], learning frameworks integrated with control barrier functions [58], [59], safety certification that continuously solves the optimization problem for a safe set at every online step [60], and model predictive control safety filters with system level synthesis have been proposed [61]. These approaches can provide system guarantees but require explicitly modeling a system's safety requirements which are not trivial to design or implement, can be overly restrictive in the safe action sets, and increase online computational effort.

III. Problem Formulation: Safety Filtering

We first define the safety filtering problem in which we would like to find a control policy (the safety filter) that closely follows a given policy (a pre-trained RL controller) while ensuring that other dynamic agents are avoided. For this purpose, consider the discrete-time dynamic control system:

$$x_{t+1} = f(x_t, u_t),\, x_0 := \zeta \tag{1}$$

where $x_t \in X \subseteq \mathbb{R}^N$ and $u_t \in U \subseteq \mathbb{R}^P$ denote the state and the control input at time $t \in \mathbb{N} \cup \{0\}$. The sets U and X denote the permissible control inputs and the system's workspace. The measurable function $f : \mathbb{R}^N \times \mathbb{R}^P \to \mathbb{R}^N$ describes the system dynamics and $\zeta \in \mathbb{R}^N$ is the initial condition of the system. The system operates in an environment with $A := \{1, 2, \cdots, m\}$ dynamic agents whose trajectories are unknown as they move from start to goal locations. Specifically, let $\mathcal{D}$ be an unknown distribution over agent trajectories, and let $(\mathcal{T}_0, \mathcal{T}_1, \cdots) \sim \mathcal{D}$ describe a random trajectory where the stacked agent states $\mathcal{T}_t := (\mathcal{T}_t^1, \cdots, \mathcal{T}_t^m)$ at time t is drawn from $\mathbb{R}^{2m}$. In this paper, each agent is assumed to operate in $\mathbb{R}^{2m}$ with a two-dimensional position, but we remark that our method is not limited to $\mathbb{R}^2$. We use $\tau_t := \{\tau_t^1, \cdots, \tau_t^m\}$ when referring to a realization of $\mathcal{T}_t$ and assume access to the history of observations $\tau_{0:t} := \{\tau_0, \cdots, \tau_t\}$ online at time t. For example, the agent trajectories in Figure 1 can be described by distributions D_1, D_2, D_3 with a joint distribution $\mathcal{D}$. We make no assumptions on the form of the distribution $\mathcal{D}$ but assume i) that $\mathcal{D}$ is independent of the system (1), and ii) the availability of calibration data independently drawn from $\mathcal{D}$. We comment on these assumptions in our experiments.

Assumption 1: For any time $t \geq 0$, the control inputs $(u_0, \cdots, u_{t-1})$ and the resulting trajectory $(x_0, \cdots, x_t)$, following (1), do not change the distribution of $(\mathcal{T}_0, \mathcal{T}_1, \cdots) \sim D$.

Assumption 1 holds approximately in many robotic applications, such as autonomous vehicles, where a car is likely to behave in ways that result in socially acceptable trajectories. In our experiments, we argue and show that the system is unlikely to drastically change the behavior of other agents, in which case conformal prediction still provides valid guarantees [62]. We later comment on such distribution shifts in more detail and reserve a thorough treatment for future work. We also assume the availability of training and calibration data drawn from $\mathcal{D}$.

Assumption 2: We have a dataset of trajectories $D := \{\tau^{(0)}, \tau^{(1)}, \cdots, \tau^{(K)}\}$ in which each of the K trajectories $\tau^{(i)} := \{\tau_0^{(i)}, \tau_1^{(i)}, \cdots\}$ is independently drawn from $\mathcal{D}$.

Assumption 2 is not restrictive as large amounts of data can be obtained from rapidly advancing high-fidelity simulators or from robotic applications such as autonomous vehicles where datasets are becoming widely available. Lastly, we split the dataset D into training datasets D_{Ytrain} and D_{train} from which we will train trajectory predictor and safety filter, respectively, and a calibration dataset D_{cal} from which we will quantify the uncertainty of the trajectory predictor.

For this system, we are given a pre-defined controller $\pi : \mathbb{R}^N \times \mathbb{R}^{2m} \to \mathbb{R}^P$ providing the control inputs

$$u_t := \pi(x_t, \tau_t). \tag{2}$$

Here, the policy π is an RL policy that aims to reach a final location while avoiding collisions with agents in A. However, RL policies may not always be successful at this task. Our goal is to construct a predictive safety filter that closely follows the RL policy while guaranteeing that the probability of a collision is upper bounded by a desired failure probability.

IV. Conformal Predictive Safety Filter

We provide an overview of our technique upfront in Figure 2. The goal is to add a safety filter $\hat{\pi}$ to the pre-defined RL policy π that may not have any safety certification or is only valid under certain assumptions, e.g., $\mathcal{D}$ being Gaussian. Specifically, we use a trajectory predictor Y to predict future agent trajectories from past agent observations and conformal prediction (CP) to obtain uncertainty intervals for these predictions. The predictive safety filter uses this information to achieve the safety of the RL policy π while minimally deviating from π. In the remainder, we explain the offline training of our safety filter $\hat{\pi}$, as summarized in Algorithm 1. The online execution of $\hat{\pi}$ is summarized in Algorithm 2 and later explained and validated.

Trajectory Predictor: Given a prediction horizon H and the history of agent observations $\tau_{0:t}$, we desire a trajectory predictor $Y : \mathbb{R}^{(t+1)2m} \to \mathbb{R}^{2mH}$ that predicts the H future agent states $(\mathcal{T}_{t+1}, \ldots, \mathcal{T}_{t+H})$ as $\bar{\tau}_{t+1:H} := Y(\tau_{0:t})$ where

$$Y(\tau_{0:t}) := (\bar{\tau}_{t+1}, \cdots, \bar{\tau}_{t+H}). \tag{3}$$

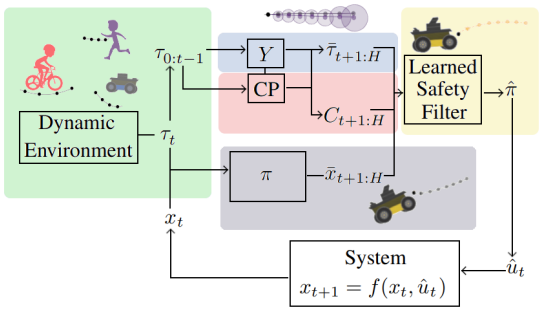

Fig. 2. Overview of our predictive safety filter that produces a control policy $\hat{\pi}$ for the potentially unsafe RL policy π .

In principle, we can use any trajectory predictor Y, e.g., long short-term memory networks [6], [7] or transformer architectures [8]. For training Y, we independently sample from $\mathcal{D}$ a dataset D_{Ytrain} with trajectories from time 0 to time T, i.e., $\tau_{0:T}^{(i)} := (\tau_0^{(i)}, \cdots, \tau_t^{(i)}, \tau_{t+1}^{(i)}, \cdots, \tau_T^{(i)})$ is the ith trajectory in the dataset D_{Ytrain}. In this work, we implement our predictor by training a long short-term memory network by minimizing the following loss function over the training set D_{Ytrain} (line 3 in Algorithm 1):

$$\min_Y \frac{1}{|D_{Ytrain}|} \sum_{i=1}^{|D_{Ytrain}|} \|\tau_{t+1:H}^{(i)} - Y(\tau_{0:t}^{(i)})\|^2 \tag{4}$$

Conformal Prediction Regions: We use CP to construct regions around the predicted trajectories that contain the true but unknown trajectory with high probability. For a general introduction to CP, we refer the reader to [9]. For trajectory predictions, the authors in [30], [31] present a technique to construct valid prediction regions applied to recurrent neural networks, which we briefly summarize next. Given observations $\tau_{0:t} := (\tau_0, \cdots, \tau_t)$ at time t, where $\tau_t := (\tau_t^0, \cdots, \tau_t^m)$, we can use the trajectory predictor Y to obtain predictions $\bar{\tau}_{t+1:H} := (\bar{\tau}_{t+1}, \cdots, \bar{\tau}_{t+H})$ for the specified prediction horizon H. Given a failure probability of $\delta \in (0, 1)$, we seek values $C_{t+1:H} := (C_{t+1}, \cdots, C_{t+H})$ as prediction intervals around each prediction such that:

$$\text{Prob}(||\tau_{t+h} - \bar{\tau}_{t+h}|| \leq C_{t+h},\ \forall h \in \{1, \cdots, H\}) \geq 1 - \delta \tag{5}$$

The approach is summarized in lines 4 - 10 of Algorithm 1. First, we define the non-conformity score function $R_{t+h} := ||\tau_{t+h} - \bar{\tau}_{t+h}||$ in line 7, and we evaluate the predictor based on this function across a conformal calibration dataset D_{cal} that we independently sample from the distribution $\mathcal{D}$. Note that a small non-conformity score corresponds to accurate predictions, while a large score indicates that Y is inaccurate. Second, we sort the non-conformity scores from the calibration dataset D_{cal} in non-decreasing order and append infinity as the $(|D_{cal}|+1)$-th value (see lines 8 and 9 in Algorithm 1). Third, we define $p := \lceil (|D_{cal}|+1)(1-\bar{\delta}) \rceil$ (line 5 of Algorithm 1) and let the prediction interval C_{t+h} correspond to the pth quantile over the sorted non-conformity scores for each prediction step $h \in \{1, \ldots, H\}$ (line 10 of Algorithm 1). From [30, Theorem 1], we set $\bar{\delta} := \delta / T$

Algorithm 1 Offline Training & CP Calibration

1: **Input:** Failure probability δ, horizons H and T, pre-trained controller π, D_{Ytrain}, D_{cal}, D_{train} from $\mathcal{D}$,
2: **Output:** Conformal Predictor Y, CPSF-controller $\hat{\pi}$
3: Learn trajectory predictor Y from D_{Ytrain} as in (4)
4: $\bar{\delta} \leftarrow \delta / T$
5: $p \leftarrow \lceil (|D_{cal}|+1)(1-\bar{\delta}) \rceil$
6: **for** h from 1 to H **do**
7: $\quad R^{(i)}_{t+h} \leftarrow ||\tau^{(i)}_{t+h} - Y(\tau^{(i)}_{0:t+h})||$ # for each $i \in D_{cal}$
8: $\quad R^{|D_{cal}|+1}_{t+h} \leftarrow \inf$
9: $\quad$ Sort $R^{(i)}_{t+h}$ in non-decreasing order
10: $\quad C_{t+h} \leftarrow R^{p}_{t+h}$
11: **for** instance i in D_{train} **do**
12: $\quad \bar{\tau}^{(i)}_{t+1:H}, \leftarrow Y(\tau^{(i)}_{0:t})$
13: $\quad \bar{x}^{(i)}_{t+1:H} \leftarrow$ simulate the system forward by H using π
14: $\quad D^{(i)}_{sftrain} \leftarrow (D^{(i)}_{train}, \bar{\tau}^{(i)}_{t+1:H}, \bar{u}^{(i)}_{t:H-1}, \bar{x}^{(i)}_{t+1:H}, C_{t+1:H})$
15: Learn $\hat{\pi}$, given $D_{sftrain}$ as the solution of (7)

in line 4 to ensure collision avoidance with a probability of at least $1-\delta$ across the steps. Finally, when we make predictions online, we use the values $C_{t+1:H}$ as prediction intervals around each predicted trajectory $\bar{\tau}_{t+1:H}$.

Training Predictive Safety Filters: Our approach to training a predictive safety filter $\hat{\pi}$ is to 1) forward simulate the system from (1) under the nominal RL policy π from (2) using the trajectory predictions from (3), and 2) enforce that the trajectory of the system from (1) under the safety filter $\hat{\pi}$ imitates the trajectory under the RL policy π while incorporating the conformal predictions regions $C_{t+1:H}$ to account for uncertainty in the trajectory predictions.

For the first step, we use the pre-defined RL policy π and the predictions from Y to forward simulate the system dynamics under the RL policy into the future by H. We obtain this nominal future trajectory $\bar{x}_{t+1:H} := (\bar{x}_{t+1}, ..., \bar{x}_{t+H})$ as

$$\begin{aligned} \bar{x}_t &:= x_t \\ \bar{u}_t &:= \pi(x_t, \tau_t) \\ \bar{x}_{t+h} &:= f(\bar{x}_{t+h-1}, \bar{u}_{t+h-1}),\ \forall h \in \{1, ..., H\} \\ \bar{u}_{t+h} &:= \pi(\bar{x}_{t+h}, \bar{\tau}_{t+h}),\ \forall h \in \{1, ..., H-1\} \end{aligned} \tag{6}$$

Now, the safety filter $\hat{\pi} : \mathbb{R}^{2mH} \times \mathbb{R}^{HM} \times \mathbb{R}^{HN} \times \mathbb{R}^{H} \rightarrow \mathbb{R}^{HP}$ optimizes the following objective:

$$\begin{aligned} \min_{\hat{\pi}} \quad & \sum_{h=1}^{H} ||\bar{x}_{t+h} - \hat{x}_{t+h}||^2 \\ \text{s.t.} \quad & \hat{x}_t := x_t \\ & \hat{x}_{t+h} := f(\hat{x}_{t+h-1}, \hat{u}_{t:H-1}(h)),\ \forall h \in \{1, ..., H\} \\ & \hat{u}_{t:H-1} := \hat{\pi}(\bar{\tau}_{t+1:H}, \bar{u}_{t:H-1}, \bar{x}_{t+1:H}, C_{t+1:H}) \\ & ||\bar{\tau}^{j}_{t+h} - \hat{x}_{t+h}|| \geq C_{t+h} + \epsilon,\ \forall h \in \{1, ..., H\}, \forall j \in A \end{aligned} \tag{7}$$

The (to be learned) safety filter produces $\hat{u}_{t:H-1} := \hat{\pi}(\bar{\tau}_{t+1:H}, \bar{u}_{t:H-1}, \bar{x}_{t+1:H}, C_{t+1:H})$, where $\hat{u}_{t:H-1}$ contains control inputs for the next H time steps. We use the notation $\hat{u}_{t:H-1}(h)$ to access the control input for the hth time step. The filter is recursively applied to the system across timesteps T. Intuitively, the safety filter minimizes the distance between the nominal RL trajectory $\bar{x}_{t+1:H}$ and the safety filter trajectory $\hat{x}_{t+1:H}$, i.e., the trajectory obtained under the safety filter $\hat{\pi}$. Additionally, the safety filter trajectory $\hat{x}_{t+1:H}$ should avoid the agent predictions $\bar{\tau}^{j}_{t+h}$ (for all agents $j \in A$) and uncertainty intervals C_{t+h}. Specifically, we enforce the safety constraint $||\tau^{j}_{t+h} - \hat{x}_{t+h}|| \geq \epsilon$, where τ^{j}_{t+h} is unknown, and $\epsilon > 0$ is a user-defined minimum collision avoidance distance, where $||\bar{\tau}^{j}_{t+h} - \hat{x}_{t+h}|| \geq C_{t+h} + \epsilon$ holds. Since we know Prob$(||\tau_{t+h} - \bar{\tau}_{t+h}|| \leq C_{t+h},\ \forall h \in \{1, \ldots, H\}) \geq 1-\delta$, it is ensured that Prob$(||\tau^{j}_{t+h} - \hat{x}_{t+h}|| \geq \epsilon,\ \forall h \in \{1, \ldots, H\}) \geq 1-\delta$.

Algorithm 2 Online Uncertainty Informed Safety Filter

1: **Input:** prediction and mission horizons H and T, controller π, trajectory predictor Y, predictive safety filter $\hat{\pi}$
2: **for** t from 0 to $T-1$ **do**
3: $\quad$ Sense x_t and τ_t
4: $\quad$ Compute $\bar{\tau}_{t+1:H}, \bar{u}_{t:H-1}, \bar{x}_{t+1:H}, C_{t+1:H}$
5: $\quad$ Compute safety filter as $\hat{u}_{t:H-1} \leftarrow \hat{\pi}(\bar{\tau}_{t+1:H}, \bar{u}_{t+1:H}, \bar{x}_{t+1:H}, C_{t+1:H})$
6: $\quad$ Apply $\hat{u}_{t:H-1}(1)$ to the system

In lines 11- 14 of Algorithm 1, we sample another dataset D_{train} of independent trajectories from $\mathcal{D}$ to train the safety filter. For each training trajectory, we construct the corresponding predictions and nominal RL trajectories (lines 12 and 13), from which we construct the combined training dataset $D_{sftrain}$ (line 14) that we use to solve (7). Specifically, we train a multi-layer feedforward neural network $\hat{\pi}$ in line 15. In particular, we solve (7) approximately over the training set $D_{sftrain}$ and rewrite the constrained optimization problem (7) into an unconstrained optimization problem. Therefore, we augment the original cost function with the constraints of (7) that we then minimize over to convergence.

Online Execution: Once the safety filter is trained, it can be used during runtime by first computing the set of H next control inputs $\hat{u}_{t:H-1} := \hat{\pi}(\bar{\tau}_{t+1:H}, \bar{u}_{t:H-1}, \bar{x}_{t+1:H}, C_{t+1:H})$. Specifically, we apply the safety filter in a receding horizon fashion by applying the first element $\hat{u}_{t:H-1}(1)$ at each time step t. The online execution is summarized in Algorithm 2. In line 4, we pass the history of trajectories $\tau_{0:t}$ to our predictor Y to obtain predictions $\bar{\tau}_{t+1:H}$, we also obtain the nominal RL control inputs and trajectory $\bar{u}_{t:H-1}$ and $\bar{x}_{t+1:H}$, as well as the conformal uncertainty intervals $C_{t+1:H}$. In lines 5 and 6, we compute the safety filtered control input and apply $\hat{u}_{t:H-1}(1)$ to the system.

Guarantees: Under the assumption that the safety filter achieves the objective in (7), we remark that our safety filter guarantees probabilistic safety, i.e., that Prob$(||\tau^{j}_{t} - \hat{x}_{t}|| \geq$

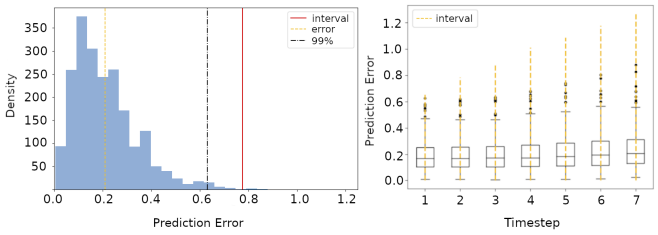

Fig. 3. (Left) The average prediction error, with the conformal interval at a timestep. (Right) Boxplots of the prediction errors by timesteps ahead, with the conformal interval coverage displayed as a yellow vertical dashed line.

$\epsilon,\ \forall t \in \{1,\dots,H\}) \geq 1-\delta$ as we use $\bar{\delta} := \delta/T$. This follows the same reasoning as in [30, Theorems 1 and 3]. Note that these guarantees hold under idealized assumptions. In practice, there may be reasons why we do not achieve exact probabilistic coverage. One reason is the way we approximately solve the optimization problem (7) over the training set $D_{sftrain}$ by rewriting (7) as an unconstrained optimization problem. In our experimental results, however, we observe that a safety filter $\hat{\pi}$ trained with sufficient data performs well in practice and that our method provides an added layer of safety over the baseline RL policy.

Furthermore, we note that the prediction intervals C_{t+h} naturally depend on the underlying distribution $\mathcal{D}$, the predictor's accuracy, and the user-specified risk tolerance δ. Without the intervals C_{t+h}, the safety filter $\hat{\pi}$ would only mimic the baseline RL controller π. Therefore, the safety filter $\hat{\pi}$ may perform differently than π. Additionally, Figure 2 demonstrates the modularity of our approach. This attribute enables the use of potentially different policies with the learned safety filter, which we will investigate in future work.

As per Assumption 1, we assume that the distribution of trajectories does not change from offline training to online testing. In practice, we argue that small distribution shifts in $\mathcal{D}$ will not significantly affect our algorithm and its guarantees. A supporting argument of our claim is given in [62], where it is formally shown that small distribution shifts only lead to small deviations from the desired conformal prediction guarantees. In our experiments, we explicitly check that agent interaction does not introduce a larger distribution shift, as we will verify in Figure 4. One may increase the robustness of the prediction intervals using adaptive conformal prediction [44].

V. Experimental Evaluation

The experiments were run on a 5.0 GHz Turbo Intel Core i9-9900K computer with 16GB RAM in the Collision Avoidance Gym [2]. The multiagent gym environment is open-sourced and provides pre-trained RL policies for navigation from given start to goal locations. Agents can sense the locations and velocities of the other agents. The gym allows customizable dynamic models and supports quick benchmark testing against future techniques. In [2], the authors demonstrate the transferability of policies from training in their gym simulator to deployment on real-world aerial and ground robots in real-time with diverse sensors. For the dataset generation and online experiments, we consider bicycle system dynamics, a planning frequency of 10Hz, and an agent radius of $\epsilon := 0.5$. All dynamic agents use $CADRL$, a collision avoidance RL policy trained to mimic socially acceptable pedestrian movement [3]. We consider three scenarios with sets of $(2, 4, 6)$ dynamic agents A. For each scenario, we ran 20000 instances of trajectories sampled from $\mathcal{D}$ that we split equally between the predictor and safety filter training datasets D_{Ytrain} and D_{train}, respectively. An additional 1000 trajectories were collected for the conformal calibration dataset D_{cal}.

We use GA3C-CADRL (GA3C) as our given RL policy π around agents from $\mathcal{D}$ to train our safety filter. A pre-trained policy for GA3C is available in the gym environment, has received significant attention, and has demonstrated good performance when transferred to a fully autonomous robot moving around pedestrians. The authors note that collisions still occur. For comparison, we generate a dataset and train a second safety filter that uses the more conservative reciprocal velocity avoidance algorithm ORCA controller [63]. The flexibility of our method allows us to apply our safety filter to both GA3C and ORCA controllers to produce CPSF-GA3C and CPSF-ORCA.

For uncertainty quantification of our predictor, we set $\delta := 0.01$ to provide our safety filter 99% confidence during training. This can be tuned to produce larger avoidance actions or smaller ones, depending on the level of conservatism desired. The average length of an experiment is around 8 seconds, i.e., $T := 80$, and we set $H := 7$. We tested various lengths of prediction horizons H and found negligible improvement beyond $H := 7$ future steps. We set the architecture for the predictor LSTM to be 2 layers of 128 hidden units and the safety filter network to be 3 layers of 128.

In Figure 3, we visualize our CP method performance across the calibration dataset and observe the average interval relative to all non-conformal scores. On the right, we see the growth of the intervals with each step into the future. The environment consists of only the system and the agents sampled from the distribution $\mathcal{D}$. The agents may approach each other, and interactions in close proximity could produce a change in the distribution $\mathcal{D}$. However, to verify that Assumption 1 approximately holds, we group trajectories from a test set by their minimum inter-agent distances of all agents into four categories $[0, 1.5), [1.5, 2.5), [2.5, 3.5)$, and $[3.5, \infty)$. We then plot the histograms of the prediction errors for the 1, 4, and 7 step ahead prediction for each category in Figure 4. We can see that the histograms between the first three categories look nearly identical, indicating no significant distribution shift. For the last category, the histograms show differences. However, we note that there were fewer trajectories in this category. Therefore, the histogram for that category may not be as accurate, and we note that trajectories from this category are less likely to occur in our experiments. The plot approximately justifies the independence assumption of the trajectories made in CP.

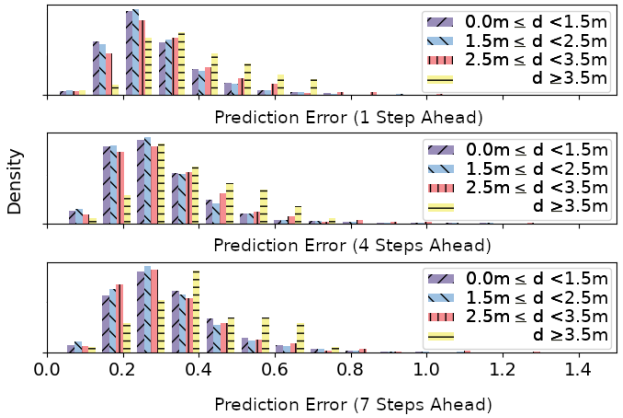

Fig. 4. For a set of test trajectories, we categorize each trajectory based on the minimum inter-agent distance d between all agents. We consider four different categories of minimum inter-agent distances $[0, 1.5), [1.5, 2.5), [2.5, 3.5)$, and $[3.5, \infty)$. We use this as a proxy for analyzing agent interactions. We plot the histograms of prediction errors for each category's 1st, 4th, and 7th step-ahead predictions. We can see that the histograms between the first three categories look almost identical, indicating no significant distribution shift. For the last category, where we consider larger inter-agent distances, we observe slightly different histograms.

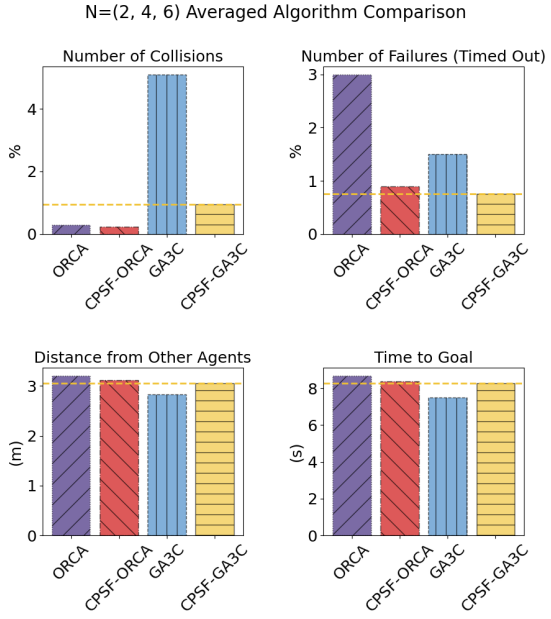

Fig. 5. The averaged performance across all agent set sizes.

Our safety filter method is not limited to CP for statistical guarantees. By swapping out the CP for a standard Gaussian process, which learns a mean and standard deviation around each prediction as a safety guarantee, we can train a second safety filter, GASF-GA3C.

VI. Comparative Results for Predictive Safety Filtering

For the online experiments, we ran 1000 instances for each set of agents $(2, 4, 6)$. We present the average performance across all agent test sets, comparing the percentage of collisions and failures (where the agent timed out), the average minimum distances to other agents, and the time to goal in Figure 5. Table I lists the average performance across all instances for the 4 agent case.

GA3C performs less conservative trajectories that lead to a shorter time to goal but a higher number of collisions than ORCA. However, ORCA performs overly conservative trajectories that reduce collisions but increase failure to reach the goal. Our conformal predictive safety filter minimizes the negative side effects of each control policy, with fewer collisions for CPSF-GA3C and fewer failures for CPSF-ORCA. We highlight that the results match our risk tolerance set by $\delta := 0.01$, with the nonzero collisions under a configurable threshold. GA3C, without the safety filter, has the shortest distance from other agents and time to goal, but our safety filter performs as well as the other algorithms in time to goal and minimally impacts GA3C.

We acknowledge that the simulated gym environment is dense, and the distances between agents can be small, with subtle changes during key interactions. Fine-tuning the hyperparameters or δ could produce more drastic changes as desired. We note that the learned CPSF algorithms, trained to mimic the original policies, may differ from the expert policy due to conformal prediction constraints during learning, imperfect training, and finite training data. Overall, the CPSF algorithms outperform the non-CPSF algorithms in the number of collisions and failures. The CPSF algorithms maintain similar time to goals and distances from other agents and show an awareness of the uncertainty in the underlying approaches.

We ran an additional 1000 instances with CPSF-GA3C and listed the results in Table II for 4 agents. The Gaussian approach captures the prediction uncertainty but increases the agent's time to the goal. CP does not assume a Gaussian distribution and can deliver a specified-risk level guarantee that the predictions contain the true location of the agent, resulting in a lower time to the goal.

Figure 6 visualizes the trajectories, showing a run of the simulation in the collision avoidance gym. The agents perform smooth trajectories that avoid collisions while efficiently reaching their goals. Our method in a widely-used RL collision avoidance simulator achieves 80% fewer collisions when paired with GA3C-CADRL and 67% fewer failures when paired with ORCA, while planning paths shorter by 18% than the Gaussian-based safety filter.

VII. Conclusion

While there has been impressive progress in RL for robot motion planning in dynamic environments, RL policies are often unsafe, and the rate of collisions can be at an undesirable level. We present a framework for learning predictive safety filters for safe motion planning by predicting future trajectories of other agents and quantifying prediction uncertainty using conformal prediction. We evaluated our framework for multiple dynamic agents in a collision avoidance gym environment that has been shown to transfer to physical robots. Our proposed predictive safety filter achieved fewer

TABLE I

N=4 COLLISION AVOIDANCE EXPERIMENTS

System Controller	Collisions	Failures (timed out)	Distance from Other Agents (m)	Time to Goal (s)
ORCA	3	27	3.201	8.678
CPSF-ORCA	2	15	3.131	8.412
GA3C	49	14	2.835	7.503
CPSF-GA3C	10	8.0	3.057	8.278

TABLE II

N=4 SAFETY COVERAGE

Method	% Inside Interval	Distance from Other Agents (m)	Time to Goal (s)
CPSF-GA3C	99%	3.057	8.278
GASF-GA3C	100%	3.860	10.05

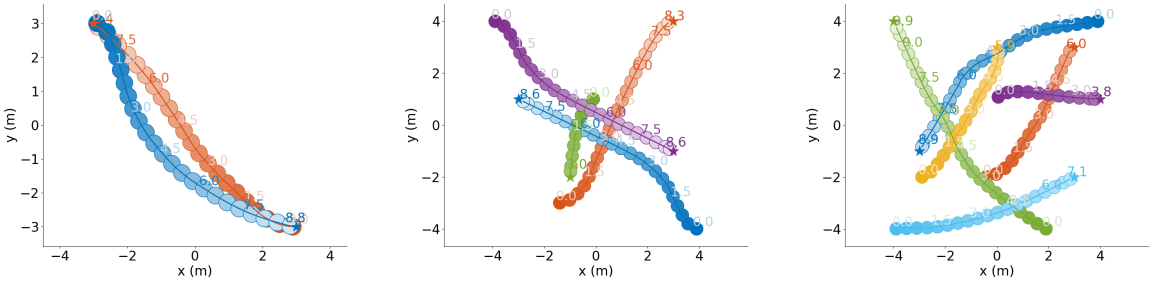

Fig. 6. Our system (orange) starts at $(-3, 3)$, $(-2, -3)$, and $(-1, -2)$ in each case, respectively. The path darkens as time increases.

collisions without increasing the time-to-goal or number of failures to reach the goal.

In the future, we will test the predictive safety filter for other controllers (not only RL controllers) and verify the modularity, e.g., using the safety filter for a controller not considered during training. Future work on applying adaptive CP is a promising direction to relax some of the assumptions made in this work, e.g., when distribution shifts occur in $\mathcal{D}$.

REFERENCES

[1] R. Kümmerle, M. Ruhnke, B. Steder, C. Stachniss, and W. Burgard, "A navigation system for robots operating in crowded urban environments," in *2013 IEEE Int. Conf. on Robotics and Automation*. IEEE, 2013, pp. 3225–3232.

[2] M. Everett, Y. F. Chen, and J. P. How, "Collision avoidance in pedestrian-rich environments with deep reinforcement learning," *IEEE Access*, vol. 9, pp. 10 357–10 377, 2021.

[3] Y. F. Chen, M. Everett, M. Liu, and J. P. How, "Socially aware motion planning with deep reinforcement learning," in *2017 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*. IEEE, 2017, pp. 1343–1350.

[4] Y. F. Chen, M. Liu, M. Everett, and J. P. How, "Decentralized non-communicating multiagent collision avoidance with deep reinforcement learning," in *2017 IEEE Int. Conf. on robotics and automation (ICRA)*. IEEE, 2017, pp. 285–292.

[5] M. Everett, Y. F. Chen, and J. P. How, "Motion planning among dynamic, decision-making agents with deep reinforcement learning," in *2018 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 3052–3059.

[6] A. Alahi, K. Goel, V. Ramanathan, A. Robicquet, L. Fei-Fei, and S. Savarese, "Social lstm: Human trajectory prediction in crowded spaces," in *Proceedings of the IEEE Conf. on computer vision and pattern recognition*, 2016, pp. 961–971.

[7] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[8] N. Nayakanti, R. Al-Rfou, A. Zhou, K. Goel, K. S. Refaat, and B. Sapp, "Wayformer: Motion forecasting via simple & efficient attention networks," *arXiv preprint arXiv:2207.05844*, 2022.

[9] A. N. Angelopoulos and S. Bates, "A gentle introduction to conformal prediction and distribution-free uncertainty quantification," *arXiv preprint arXiv:2107.07511*, 2021.

[10] R. Luo, S. Zhao, J. Kuck, B. Ivanovic, S. Savarese, E. Schmerling, and M. Pavone, "Sample-efficient safety assurances using conformal prediction," in *Algorithmic Foundations of Robotics XV: Proceedings of the Fifteenth Workshop on the Algorithmic Foundations of Robotics*. Springer, 2022, pp. 149–169.

[11] T. G. Dietterich and J. Hostetler, "Conformal prediction intervals for markov decision process trajectories," *arXiv preprint arXiv:2206.04860*, 2022.

[12] L. Bortolussi, F. Cairoli, N. Paoletti, S. A. Smolka, and S. D. Stoller, "Neural predictive monitoring," in *Runtime Verification: 19th Int. Conf., RV 2019, Porto, Portugal, October 8–11, 2019, Proceedings 19*. Springer, 2019, pp. 129–147.

[13] C. Fan, X. Qin, Y. Xia, A. Zutshi, and J. Deshmukh, "Statistical verification of autonomous systems using surrogate models and conformal inference," *arXiv preprint arXiv:2004.00279*, 2020.

[14] Y. Chen, U. Rosolia, C. Fan, A. Ames, and R. Murray, "Reactive motion planning with probabilisticsafety guarantees," in *Conf. on Robot Learning*. PMLR, 2021, pp. 1958–1970.

[15] J. Van Den Berg, S. J. Guy, M. Lin, and D. Manocha, "Reciprocal n-body collision avoidance," in *Robotics Research: The 14th Int. Symposium ISRR*. Springer, 2011, pp. 3–19.

[16] J. B. Rawlings, "Tutorial overview of model predictive control," *IEEE control systems magazine*, vol. 20, no. 3, pp. 38–52, 2000.

[17] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," *The Int. journal of robotics research*, vol. 5, no. 1, pp. 90–98, 1986.

[18] G. Ferrer, A. Garrell, and A. Sanfeliu, "Social-aware robot navigation in urban environments," in *2013 European Conf. on Mobile Robots*. IEEE, 2013, pp. 331–336.

[19] P. Trautman, J. Ma, R. M. Murray, and A. Krause, "Robot navigation in dense human crowds: Statistical models and experimental studies of human–robot cooperation," *The Int. Journal of Robotics Research*, vol. 34, no. 3, pp. 335–356, 2015.

[20] M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp,

P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang *et al.*, "End to end learning for self-driving cars," *arXiv preprint arXiv:1604.07316*, 2016.

[21] B. Kim and J. Pineau, "Socially adaptive path planning in human environments using inverse reinforcement learning," *Int. Journal of Social Robotics*, vol. 8, pp. 51–66, 2016.

[22] H. Kretzschmar, M. Spies, C. Sprunk, and W. Burgard, "Socially compliant mobile robot navigation via inverse reinforcement learning," *The Int. Journal of Robotics Research*, vol. 35, no. 11, pp. 1289–1307, 2016.

[23] M. Pfeiffer, U. Schwesinger, H. Sommer, E. Galceran, and R. Siegwart, "Predicting actions to act predictably: Cooperative partial motion planning with maximum entropy models," in *2016 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*. IEEE, 2016, pp. 2096–2101.

[24] L. Tai, G. Paolo, and M. Liu, "Virtual-to-real deep reinforcement learning: Continuous control of mobile robots for mapless navigation," in *2017 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*. IEEE, 2017, pp. 31–36.

[25] A. Farid, S. Veer, B. Ivanovic, K. Leung, and M. Pavone, "Task-relevant failure detection for trajectory predictors in autonomous vehicles," in *Conf. on Robot Learning*. PMLR, 2023, pp. 1959–1969.

[26] P. Long, T. Fan, X. Liao, W. Liu, H. Zhang, and J. Pan, "Towards optimally decentralized multi-robot collision avoidance via deep reinforcement learning," in *2018 IEEE Int. Conf. on robotics and automation (ICRA)*. IEEE, 2018, pp. 6252–6259.

[27] A. Mészáros, J. Alonso-Mora, and J. Kober, "Trajflow: Learning the distribution over trajectories," *arXiv preprint arXiv:2304.05166*, 2023.

[28] A. Mesbah, K. P. Wabersich, A. P. Schoellig, M. N. Zeilinger, S. Lucia, T. A. Badgwell, and J. A. Paulson, "Fusion of machine learning and mpc under uncertainty: What advances are on the horizon?" in *2022 American Control Conf. (ACC)*. IEEE, 2022, pp. 342–357.

[29] K. P. Wabersich, L. Hewing, A. Carron, and M. N. Zeilinger, "Probabilistic model predictive safety certification for learning-based control," *IEEE Transactions on Automatic Control*, vol. 67, no. 1, pp. 176–188, 2021.

[30] L. Lindemann, M. Cleaveland, G. Shim, and G. J. Pappas, "Safe planning in dynamic environments using conformal prediction," *IEEE Robotics and Automation Letters*, vol. 8, no. 8, pp. 5116–5123, 2023.

[31] K. Stankeviciute, A. M Alaa, and M. van der Schaar, "Conformal time-series forecasting," *Advances in Neural Information Processing Systems*, vol. 34, pp. 6216–6228, 2021.

[32] F. Berkenkamp and A. P. Schoellig, "Safe and robust learning control with gaussian processes," in *2015 European Control Conf. (ECC)*. IEEE, 2015, pp. 2496–2501.

[33] S. Thrun, "Probabilistic robotics," *Communications of the ACM*, vol. 45, no. 3, pp. 52–57, 2002.

[34] L. Niu, H. Zhang, and A. Clark, "Safety-critical control synthesis for unknown sampled-data systems via control barrier functions," in *2021 60th IEEE Conf. on Decision and Control (CDC)*. IEEE, 2021, pp. 6806–6813.

[35] A. D. Ames, S. Coogan, M. Egerstedt, G. Notomista, K. Sreenath, and P. Tabuada, "Control barrier functions: Theory and applications," in *2019 18th European control Conf. (ECC)*. IEEE, 2019, pp. 3420–3431.

[36] N. Rober, S. M. Katz, C. Sidrane, E. Yel, M. Everett, M. J. Kochenderfer, and J. P. How, "Backward reachability analysis of neural feedback loops: Techniques for linear and nonlinear systems," *IEEE Open Journal of Control Systems*, 2023.

[37] S. Muntwiler, K. P. Wabersich, A. Carron, and M. N. Zeilinger, "Distributed model predictive safety certification for learning-based control," *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 5258–5265, 2020.

[38] H. Papadopoulos, "Inductive conformal prediction: Theory and application to neural networks," in *Tools in artificial intelligence*. Citeseer, 2008.

[39] V. Vovk, A. Gammerman, and G. Shafer, *Algorithmic Learning in a Random World*. Springer Science & Business Media, 2005.

[40] G. Shafer and V. Vovk, "A tutorial on conformal prediction," *Journal of Machine Learning Research*, vol. 9, no. 3, 2008.

[41] M. Fontana, G. Zeni, and S. Vantini, "Conformal prediction: a unified review of theory and new challenges," *Bernoulli*, vol. 29, no. 1, pp. 1–23, 2023.

[42] D. Foffano, A. Russo, and A. Proutiere, "Conformal off-policy evaluation in markov decision processes," *arXiv preprint arXiv:2304.02574*, 2023.

[43] M. F. Taufiq, J.-F. Ton, R. Cornish, Y. W. Teh, and A. Doucet, "Conformal off-policy prediction in contextual bandits," *arXiv preprint arXiv:2206.04405*, 2022.

[44] A. Dixit, L. Lindemann, S. Wei, M. Cleaveland, G. J. Pappas, and J. W. Burdick, "Adaptive conformal prediction for motion planning among dynamic agents," *arXiv preprint arXiv:2212.00278*, 2022.

[45] A. Muthali, H. Shen, S. Deglurkar, M. H. Lim, R. Roelofs, A. Faust, and C. Tomlin, "Multi-agent reachability calibration with conformal prediction," *arXiv preprint arXiv:2304.00432*, 2023.

[46] J. Garcıa and F. Fernández, "A comprehensive survey on safe reinforcement learning," *Journal of Machine Learning Research*, vol. 16, no. 1, pp. 1437–1480, 2015.

[47] F. Berkenkamp, M. Turchetta, A. Schoellig, and A. Krause, "Safe model-based reinforcement learning with stability guarantees," *Advances in neural information processing systems*, vol. 30, 2017.

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

[49] L. Brunke, M. Greeff, A. W. Hall, Z. Yuan, S. Zhou, J. Panerati, and A. P. Schoellig, "Safe learning in robotics: From learning-based control to safe reinforcement learning," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 5, pp. 411–444, 2022.

[50] G. Dalal, K. Dvijotham, M. Vecerik, T. Hester, C. Paduraru, and Y. Tassa, "Safe exploration in continuous action spaces," *arXiv preprint arXiv:1801.08757*, 2018.

[51] C. R. Glossop, J. Panerati, A. Krishnan, Z. Yuan, and A. P. Schoellig, "Characterising the robustness of reinforcement learning for continuous control using disturbance injection," *arXiv preprint arXiv:2210.15199*, 2022.

[52] B. Tearle, K. P. Wabersich, A. Carron, and M. N. Zeilinger, "A predictive safety filter for learning-based racing control," *IEEE Robotics and Automation Letters*, vol. 6, no. 4, pp. 7635–7642, 2021.

[53] K. P. Wabersich and M. N. Zeilinger, "A predictive safety filter for learning-based control of constrained nonlinear dynamical systems," *Automatica*, vol. 129, p. 109597, 2021.

[54] D. Seto, B. Krogh, L. Sha, and A. Chutinan, "The simplex architecture for safe online control system upgrades," in *Proceedings of the 1998 American Control Conf.. ACC (IEEE Cat. No. 98CH36207)*, vol. 6. IEEE, 1998, pp. 3504–3508.

[55] S. Prajna and A. Jadbabaie, "Safety verification of hybrid systems using barrier certificates," in *HSCC*, vol. 2993. Springer, 2004, pp. 477–492.

[56] P. Wieland and F. Allgöwer, "Constructive safety using control barrier functions," *IFAC Proceedings Volumes*, vol. 40, no. 12, pp. 462–467, 2007.

[57] K. P. Wabersich and M. N. Zeilinger, "Predictive control barrier functions: Enhanced safety mechanisms for learning-based control," *IEEE Transactions on Automatic Control*, 2022.

[58] A. Taylor, A. Singletary, Y. Yue, and A. Ames, "Learning for safety-critical control with control barrier functions," in *Learning for Dynamics and Control*. PMLR, 2020, pp. 708–717.

[59] A. Didier, R. C. Jacobs, J. Sieber, K. P. Wabersich, and M. N. Zeilinger, "Approximate predictive control barrier functions using neural networks: A computationally cheap and permissive safety filter," *arXiv preprint arXiv:2211.15104*, 2022.

[60] A. Didier, K. P. Wabersich, and M. N. Zeilinger, "Adaptive model predictive safety certification for learning-based control," in *2021 60th IEEE Conf. on Decision and Control (CDC)*. IEEE, 2021, pp. 809–815.

[61] A. P. Leeman, J. Köhler, S. Benanni, and M. N. Zeilinger, "Predictive safety filter using system level synthesis," *arXiv preprint arXiv:2212.02111*, 2022.

[62] M. Cauchois, S. Gupta, A. Ali, and J. C. Duchi, "Robust validation: Confident predictions even when distributions shift," *arXiv preprint arXiv:2008.04267*, 2020.

[63] P. Fiorini and Z. Shiller, "Motion planning in dynamic environments using velocity obstacles," *The Int. journal of robotics research*, vol. 17, no. 7, pp. 760–772, 1998.