2023 62nd IEEE Conference on Decision and Control (CDC)
December 13-15, 2023. Marina Bay Sands, Singapore

Closing the Loop on Runtime Monitors with Fallback-Safe MPC

Rohan Sinha, Edward Schmerling, and Marco Pavone

***Abstract*— When we rely on deep-learned models for robotic perception, we must recognize that these models may behave unreliably on inputs dissimilar from the training data, compromising the closed-loop system's safety. This raises fundamental questions on how we can assess confidence in perception systems and to what extent we can take safety-preserving actions when external environmental changes degrade our perception model's performance. Therefore, we present a framework to certify the safety of a perception-enabled system deployed in novel contexts. To do so, we leverage robust model predictive control (MPC) to control the system using the perception estimates while maintaining the feasibility of a safety-preserving fallback plan that does not rely on the perception system. In addition, we calibrate a runtime monitor using recently proposed conformal prediction techniques to certifiably detect when the perception system degrades beyond the tolerance of the MPC controller, resulting in an end-to-end safety assurance. We show that this control framework and calibration technique allows us to certify the system's safety with orders of magnitudes fewer samples than required to retrain the perception network when we deploy in a novel context on a photo-realistic aircraft taxiing simulator. Furthermore, we illustrate the safety-preserving behavior of the MPC on simulated examples of a quadrotor. We open-source our simulation platform and provide videos of our results at our project page: `https://tinyurl.com/fallback-safe-mpc`.**

I. INTRODUCTION

Autonomous robotic systems increasingly rely on machine learning (ML)-based components to make sense of their environment. In particular, deep-learned perception models have become indispensable to extract task-relevant information from high-dimensional sensor streams (e.g., images, pointclouds). However, it is well known that modern ML systems can behave erratically and unreliably on data that is dissimilar from the training data — inputs commonly termed out-of-distribution (OOD) [1]–[3]. During deployment, ML-enabled robots inevitably encounter OOD inputs corresponding to edge cases and rare, anomalous scenarios [1], [4], which pose a significant safety risk to ML-enabled robots.

Therefore, we examine vision-based control settings in this work, where we rely on a deep neural network (DNN) to extract task-relevant information from a high-dimensional image observation. When the DNN fails, access to this information is lost, so that we can no longer estimate the full state. For example, the drone in Fig. 1 solely relies on a DNN for obstacle detection and subsequent avoidance, and the aircraft in Fig. 2 utilizes a DNN to estimate its runway position for tracking control. Because failures caused by OOD data are difficult to anticipate, recent years have seen much progress on algorithms that monitor the performance of ML-enabled components at runtime [5]–[7]. These OOD detection algorithms aim to detect inference errors so that downstream safety-preserving

The authors are with the Dept. of Aeronautics and Astronautics at Stanford University, Stanford, CA. {rhnsinha, schmrlng, pavone}@stanford.edu. The NASA University Leadership initiative (grant #80NSSC20M0163) provided funds to assist the authors with their research, but this article solely reflects the opinions and conclusions of its authors and not any NASA entity.

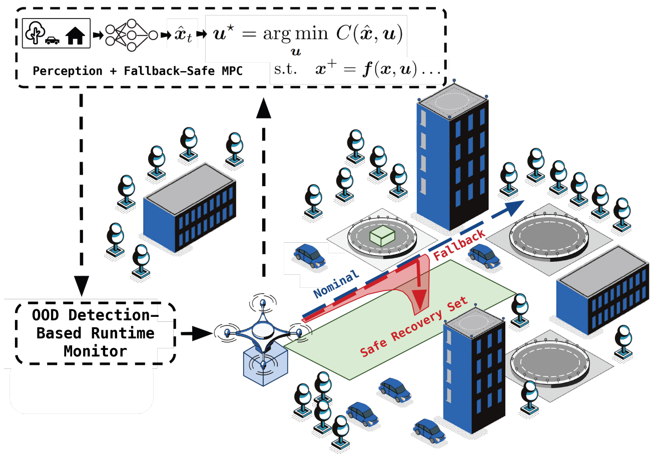
$\hat{x}_t$
$u^\star = \arg\min_u C(\hat{x}, u)$
s.t. $x^+ = f(x,u)\ldots$
Perception + Fallback-Safe MPC
OOD Detection-Based Runtime Monitor
Nominal
Fallback
Safe Recovery Set

Fig. 1. Overview of the proposed approach: A drone delivery service uses camera-vision to navigate around a city. However, the ML perception system behaves unreliably on out-of-distribution (OOD) inputs. Therefore, we construct a runtime monitor to trigger a fallback strategy when the perception system is unreliable. To do so, we calibrate heuristic OOD detection scores to decide when to land the drone. To operate this fallback safely, we must ensure that it does not drop down into trees or roads. Therefore, we minimally modify the nominal system operation to ensure we fallback into a safe recovery set using a robust MPC.

interventions may be adopted. However, few works attempt to integrate such monitors into a perception and control stack. Instead, existing work typically assumes that estimation errors will always satisfy nominal, in-distribution bounds or that there exists a safe fallback that can always be triggered under loss of sensing. To derive end-to-end certificates on the safety of the monitor-in-the-loop system, two key challenges must be addressed: (1) an OOD detector must be calibrated to detect violations of assumptions underpinning the nominal control design and (2) the control strategy itself must be aware of the limitations of a safety-preserving intervention. This latter challenge is of particular importance for safety, since, as illustrated in Fig. 1, naively executing a specified fallback (e.g., landing the drone) can introduce additional hazards.

To address these challenges, we present the Fallback-Safe Model Predictive Control (MPC) framework which, while maintaining safety, aims to derive maximum utility from the DNN component necessary for nominal task success. Our framework satisfies three key desiderata associated with the above challenges, namely: (1) we ensure the safety of the fallback strategy, i.e., we do not assume the existence of a "catch-all" fallback, (2) we explicitly quantify DNN and runtime monitor performance to inform control without resorting to conservative worst-case assumptions and (3) in the context of robust control, we account for the existence of errors, i.e., DNN failures, of arbitrary severity through the development of a runtime monitor.

In our framework, we first specify an error bound on the quality of state estimates produced by the ML perception system when it operates in-distribution (i.e., in nominal condi-

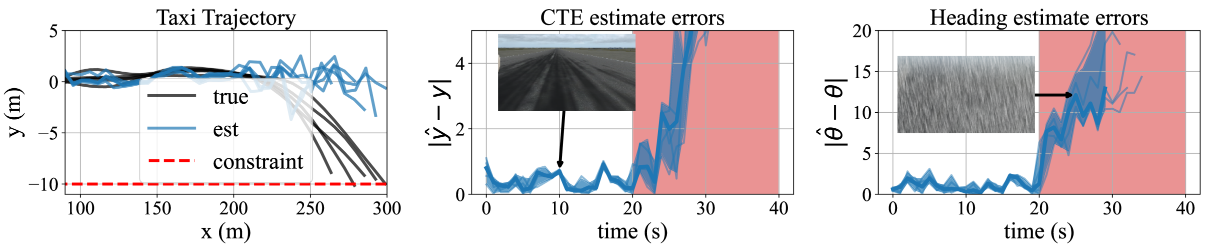

Fig. 2. 20 simulated trajectories of an autonomous aircraft taxiing down a runway. A DNN – trained on data collected in morning, clear-sky weather – estimates the cross-track error (CTE) and the heading error (HE) used by a tracking controller from vision. At the start of the trajectories, the weather is clear. At $t = 20s$, it starts raining. Leftmost plot: x is the down-track position along the runway, and y is the CTE. Middle and Rightmost plots: Errors of the DNN perception cross-track error and heading error estimate. In both cases, the estimate is initially of sufficient quality. However, when it starts raining, estimates immediately degrade (OOD time steps highlighted in red), causing the robot to fail (running off the runway) for all trajectories.

tions). In nominal conditions, we robustly control the system with respect to the specified perception error bound. Then, we calibrate an OOD detection heuristic to trigger a safety-preserving fallback strategy when the chosen perception error bound is violated, resulting in an end-to-end guarantee that the system will satisfy state and input constraints with high probability, regardless of DNN failure. This calibration procedure does require examples of such OOD cases, though notably the strength of our guarantees scales with calibration dataset size, irrespective of DNN complexity (as would, e.g., retraining on OOD cases). As such, the setting we consider captures the practice of running a few pre-deployment trials when we want to deploy a pre-trained system in a new context, which we know differs from the training distribution. For example, we can anticipate shifted conditions and run some calibration trials when we deploy an autonomous aircraft trained on images of American runways in Europe or when we deploy an image classifier trained on ImageNet as a part of our autonomy stack.

Related Work: Most similar to our approach are several works that consider triggering a fallback controller by thresholding an OOD detection algorithm [8]–[10]. These works use fallback strategies that are domain-specific or naively assumed to always be safe, like a hand-off to a human, and some approaches assume the ML models function perfectly nominally [8]. Moreover, to detect OOD conditions, they either rely on additional DNNs for OOD detection [8], [10], or use approximate techniques to quantify uncertainty [9], and thus do not make strong guarantees of system safety. Similar to existing work on fault-tolerant control that maintains feasibility of passive-backup [11], abort-safe [12], or contingency plans [13] under actuation or sensor failure using MPC, we modify the nominal operation to ensure the existence of a safety preserving fallback. However, in many such works, it is assumed that faults are perfectly detected and that these systems function perfectly nominally. It is challenging to detect failures in ML-based systems and, as illustrated in Fig. 2, errors are nominally tolerable, but nonzero. Therefore, we jointly design the control stack and runtime monitor to account for such errors.

Our approach takes inspiration from *safety filters* as defined in [14]. Such methods minimally modify a black-box policy's actions to ensure invariance of a safe subset of the state space when the black-box policy would take actions to leave that set. Many such algorithms have been proposed in recent years, for example by using robust MPC [15], control barrier functions (CBFs) [16] and Hamilton-Jacobi reachability analysis [17], [18]. However, our method differs from such approaches in two important ways: First, existing safety filters operate under assumptions of perfect state knowledge, or that an estimate of known accuracy is always available [19]. However, ML-enabled components for perception are necessary to complete the control task in many applications. When these components fail, it becomes impossible to estimate the full state (e.g., see Fig. 1, Fig. 2). We account for these discrete information modes by defining *recovery sets* to fallback into, which formalize the intuition that e.g., the drone in Fig. 1 does not need an obstacle detector to avoid mid-air collisions when it is landed in a field. Secondly, existing safety filters take a zero-confidence view in black-box learned components: They continually ensure the safe operation of the system by aligning an ML-enabled controller's output with those of a backup policy that never uses ML models. Instead, we recognize that ML-enabled components are generally reliable, but leverage OOD detection to transition to a fallback strategy in rare failure modes.

Robust control from imperfect measurements, output-feedback, classically relies on state estimators that persistently satisfy known error bounds, constructed using known measurement models with assumptions on system observability. In particular, output-feedback MPC controllers robustly satisfy state and input constraints for all time (e.g., see [20]–[25]). Typically, these methods robustly control the state estimate dynamics using a robust MPC algorithm, tightening constraints to account for the estimation error [20], [21], [23], [25]. However, we cannot model high-dimensional measurements like images from first principles, and as a result, we must rely on neural networks to extract scene information. Some recent approaches propose to learn the error behavior of a vision system as a function of the state and robustly plan while taking these error bounds into account [26]–[28], for example, by making smoothness assumptions on the vision's error behavior [26]. These algorithms require the environment to remain fixed (i.e., that the mapping from state to image is constant). However, oftentimes external changes to the environment, like the changing weather in Fig. 2, result in OOD inputs that cause arbitrarily poor perception errors. We rely on the ideas of output-feedback MPC to nominally control the robot, but maintain feasibility to a backup plan in case the perception unexpectedly degrades.

To make an end-to-end guarantee, we leverage recent results in conformal prediction to learn how to rely on a heuristic OOD detector (see [29] for an overview). Conformal methods are attractive in this setting because they produce strong guarantees on the correctness of predictions, are highly sample efficient, and the guarantees are distribution-free – that is, they do not depend on assumptions on the data-generating distribution [29], [30]. However, existing conformal prediction methods do not make high-probability guarantees jointly over predictions along a dynamical system's trajectory, where inputs are highly correlated over time. While some recent work has aimed to move beyond the exchangeable data setting [31], [32], or makes sequentially valid predictions across exchangeable samples [33], these methods cannot be applied sequentially over correlated observations within a trajectory. Instead, we adapt an existing algorithm, [34], to yield a guarantee jointly over the repeated evaluations of the predictor within a trajectory.

We include a more extensive review of existing work, further experimental details, and all proofs of our theoretical results in an extended version, available at [35].

Contributions: In brief, our contributions are that

1) First, we introduce the notion of the safe recovery set, a safe subset of the state space that is invariant under a recovery policy that does not rely on the ML-enabled perception or a full state estimate.
2) Second, we develop a framework to synthesize a fallback controller and modify the nominal operation of the robot to ensure the existence of a safe fallback strategy for all time by planning into a recovery set.
3) Third, we propose a conformal prediction algorithm that calibrates the OOD detector, resulting in a runtime-monitor-in-the-loop framework for which we make an end-to-end safety guarantee.

II. PROBLEM FORMULATION

We consider discrete time dynamical systems

$$\begin{aligned} \boldsymbol{x}_{t+1} &= \boldsymbol{f}(\boldsymbol{x}_t,\boldsymbol{u}_t,\boldsymbol{w}_t), \\ (\boldsymbol{w}_t,\boldsymbol{z}_t) &\sim p_\rho(\boldsymbol{w}_t,\boldsymbol{z}_t|\boldsymbol{w}_{0:t-1},\boldsymbol{z}_{0:t-1},\boldsymbol{x}_{0:t}), \end{aligned} \tag{1}$$

where $\boldsymbol{x}_t \in \mathbb{R}^n$ is the system state, $\boldsymbol{u}_t \in \mathbb{R}^m$ is the control input, $\boldsymbol{w}_t \in \mathcal{W} \subseteq \mathbb{R}^d$ is a disturbance signal contained in the known compact set $\mathcal{W}$ for all time steps $t \geq 0$, and $\boldsymbol{z}_t \in \mathbb{R}^o$ is a high dimensional observation consisting of an image and more conventional measurements (e.g., GPS), such that $o \gg n$. The joint distribution p_ρ from which disturbances and observations are sampled during an episode depends on an unobserved environment variable $\rho \sim P_\rho$, drawn once at the start of each episode from an environment distribution P_ρ.

Our goal is to ensure the system satisfies state and input constraints over trajectories of finite duration.

Definition 1 (Safety): Under a risk tolerance $\delta \in [0,1]$ and time limit $t_{\lim} \in \mathbb{N}_{\geq 0}$, the system (1) is *safe* if

$$\operatorname{Prob}(\boldsymbol{x}_t \in \mathcal{X},\ \boldsymbol{u}_t \in \mathcal{U},\ \forall t \in \{0,1,\ldots,t_{\lim}\}) \geq 1-\delta, \tag{2}$$

where $\mathcal{X} \subseteq \mathbb{R}^n$, $\mathcal{U} \subseteq \mathbb{R}^m$ are state and input constraint sets. Our setting differs from classic output feedback because the high-dimensional $\boldsymbol{z}_t$ cannot be used directly for control. Instead, we consider the setting where a black-box perception system generates an estimate of the state at each time step.

Assumption 1 (Perception): At each time step $t \geq 0$ a perception system produces an estimate of the state $\hat{\boldsymbol{x}}_t := \operatorname{perception}(\boldsymbol{z}_{0:t})$.

Crucially, the environment distribution P_ρ may differ from the distribution that generated the training data for the learned perception components, which means that we will eventually encounter OOD observations on which the learned components behave erratically. When the learned systems fail, only a restricted amount of information (e.g., an IMU measurement) remains accurate for control. Therefore, we do not make any assumptions on the quality of the learned system outputs, but we assume that the remaining information is accurate for all time.

Assumption 2 (Fallback Measurement): At each time step $t \geq 0$ we can access a *fallback measurement* $\boldsymbol{y}_t \in \mathbb{R}^r$ such that

$$\boldsymbol{y}_t = \boldsymbol{g}(\boldsymbol{x}_t), \quad \forall t \geq 0. \tag{3}$$

We assume $\boldsymbol{g}$ is known and call $\mathcal{Y} := \{\boldsymbol{y} = \boldsymbol{g}(\boldsymbol{x}) \ : \ \boldsymbol{x} \in \mathcal{X}\}$ the *fallback output set*. We restrict ourselves to the setting where the system $\{\boldsymbol{f},\boldsymbol{g}\}$ is not observable; perception is nominally necessary.

Since the estimates $\hat{\boldsymbol{x}}_t$ may become corrupted unexpectedly, we need to monitor the system's performance online with an intent of detecting conditions that degrade its performance. To do so, we assume we can compute a heuristic OOD detector.

Assumption 3 (OOD/Anomaly Detection): At each time step, an OOD/anomaly detection algorithm outputs a scalar anomaly signal $a_t = \operatorname{anomaly}(\boldsymbol{z}_{0:t}) \in \mathbb{R}$ as an indication of the quality of the state estimate $\hat{\boldsymbol{x}}_t$ (greater values indicate that the detector has lower confidence in the quality of $\hat{\boldsymbol{x}}_t$). Note that we make no assumptions on the quality of the OOD detector in Assumption 3: Our approach will certify the safety of the closed-loop system regardless of the correlations between a_t and the perception error $\boldsymbol{e}_t := \boldsymbol{x}_t - \hat{\boldsymbol{x}}_t$. However, the conservativeness of our algorithm will depend on the quality of the heuristic a_t. Furthermore, this framework readily allows us to incorporate algorithms that provably guarantee detection of perception errors by letting a_t be an indicator on whether the perception system is reliable. This would only simplify the control design procedure we develop in §III-§IV.

III. FALLBACK-SAFE MPC

We propose to control the system in state-feedback based on the estimates $\hat{\boldsymbol{x}}_t$ when the system operates in-distribution and use the anomaly signal a_t to monitor when $\hat{\boldsymbol{x}}_t$ becomes unreliable. Then, if the monitor triggers, we transition to a fallback policy $\pi : \mathcal{Y} \to \mathcal{U}$ that only relies on the remaining reliable information, i.e., the fallback measurement $\boldsymbol{y}_t$.

To avoid that a naively executed fallback creates additional hazards, we make two contributions in this section. First, we introduce the notions of *recovery policies* and *recovery sets*, safe subsets of the state space that we can make invariant without full state knowledge. Second, we develop a method to synthesize a fallback controller and modify the nominal operation of the robot to ensure the fallback strategy is feasible for all time.

A. Recovery Policies and Recovery Sets

Definition 1 requires that a safe fallback satisfies state and input constraints for all time, despite the fact that certain elements of the state are no longer observable. Our insight is that while this is not achievable in general, we can often identify subsets of the state space in which the robot is always safe under some $\pi_R : \mathcal{Y} \to \mathcal{U}$.

Definition 2 (Recovery Set, Policy): A set $\mathcal{X}_R \subseteq \mathcal{X}$ is a *safe recovery set* under a given recovery policy $\pi_R : \mathcal{Y} \to \mathcal{U}$ if it is a robust positive invariant (RPI) set under the recovery policy. That is, if

$$\boldsymbol{f}\big(\boldsymbol{x},\pi_R(\boldsymbol{g}(\boldsymbol{x})),\boldsymbol{w}\big) \in \mathcal{X}_R \quad \forall \boldsymbol{w} \in \mathcal{W}, \quad \forall \boldsymbol{x} \in \mathcal{X}_R. \tag{4}$$

For example, consider the quadrotor in Fig. 1. The set of all states with altitude $z=0$ and velocity $v=0$ forms a recovery set under the recovery policy $\pi_R(\boldsymbol{y}_t) := 0$. Definition 2 differs from typical definitions in output-feedback problems, because output-feedback control designs typically focus on 1) maintaining the system output $\boldsymbol{y}_t$ of a partially observed system within a set of constraints despite the unobserved dynamics, or 2) respecting constraints on the true state by bounding the estimation error of an observer. In contrast, existence of a recovery policy allows us to persistently satisfy state constraints, even when estimation errors are unbounded on OOD inputs.

B. Planning With Fallbacks

We now develop the Fallback-Safe MPC framework. First, to ensure that we satisfy safety constraints nominally, we need to define what it means for the perception system to be reliable in-distribution. Therefore, we choose a parametric compact state uncertainty set as a bound on the quality of the perception system in nominal conditions.

Definition 3 (Reliability): Let the *perception error set* $\mathcal{E}_\theta \subseteq \mathbb{R}^n$ be a symmetric compact set so that $0 \in \mathcal{E}_\theta$. We say an estimate is *reliable* when

$$\boldsymbol{e}_t := \boldsymbol{x}_t - \hat{\boldsymbol{x}}_t \in \mathcal{E}_\theta. \tag{5}$$

We say the perception system is *unreliable*, or experiences a *perception fault*, when $\boldsymbol{e}_t \notin \mathcal{E}_\theta$.
We explicitly use the subscript θ in the construction of $\mathcal{E}_\theta$ to emphasize that choosing which estimates we consider reliable is a hyperparameter. In this work, we take the perception error set as $\mathcal{E}_\theta = \{\boldsymbol{e} \in \mathbb{R}^n : \|\mathbf{A}\boldsymbol{e}\|_\infty \leq \alpha\}$ for some $\theta = (\mathbf{A}, \alpha)$ consisting of a matrix $\mathbf{A}$ and bound $\alpha \in \mathbb{R}$ (see §V). The hyperparameter θ introduces a trade-off between conservativeness in nominal operation and eagerness to trigger the fallback: Note that as $\mathcal{E}_\theta$ increases in size, the more state uncertainty we must handle as part of the tolerable estimation errors in nominal conditions, increasing nominal conservatism. If we decrease the size of $\mathcal{E}_\theta$, the more eager we will be to trigger the fallback.

Then, in nominal conditions, we control the system using the state estimates $\hat{\boldsymbol{x}}_t$ with robust MPC, minimally modifying the nominal control objective so that there always exists a fallback strategy that reaches a *recovery set* within $T+1 \in \mathbb{N}_{>0}$ time steps. To do so, we optimize two policy sequences: 1) a sequence of parametric fallback policies $\boldsymbol{u}^F_{t:t+T|t} \subset \mathcal{P}_F \subset \{\pi : \mathbb{R}^r \to \mathbb{R}^m\}$, which may only rely on the fallback measurement $\boldsymbol{y}_t$ and 2) a nominal policy sequence $\boldsymbol{u}_{t:t+T|t} \subset \mathcal{P}_N \subseteq \{\pi : \mathcal{X} \to \mathcal{U}\}$ within a state-feedback policy class $\mathcal{P}_N$, which we assume respects input constraints. In addition, we assume that for any $\boldsymbol{u} \in \mathcal{U}$ and $\hat{\boldsymbol{x}} \in \mathcal{X}$, there exists a $\pi \in \mathcal{P}_N$ such that $\boldsymbol{u} = \pi(\boldsymbol{x})$. We can trivially satisfy this assumption by, e.g., optimizing over open-loop nominal input sequences. Note that the estimator dynamics satisfy

$$\hat{\boldsymbol{x}}_{t+1} = \boldsymbol{f}(\hat{\boldsymbol{x}}_t + \boldsymbol{e}_t, \boldsymbol{u}_t, \boldsymbol{w}_t) - \boldsymbol{e}_{t+1}. \tag{6}$$

We bound the evolution of the state estimates over time for a given fallback policy sequence as follows:

Lemma 1 (Reachable Sets): Assume we apply a fixed fallback policy sequence $\boldsymbol{u}^F_{0:T} \subset \mathcal{P}_F$ from timestep t to $t+T$. Define the $k-$step reachable sets of the estimate $\hat{\boldsymbol{x}}_t$ recursively as $\widehat{\mathcal{R}}_0(\hat{\boldsymbol{x}}_t, \boldsymbol{u}^F_{0:T}) := \{\hat{\boldsymbol{x}}_t\}$ and

$$\widehat{\mathcal{R}}_{k+1}(\hat{\boldsymbol{x}}_t, \boldsymbol{u}^F_{0:T}) := \left\{ \begin{matrix} \boldsymbol{f}(\hat{\boldsymbol{x}}+\boldsymbol{e}, \boldsymbol{u}^F_k(\boldsymbol{g}(\hat{\boldsymbol{x}}+\boldsymbol{e})), \boldsymbol{w}) \\ -\boldsymbol{e}' \end{matrix} : \begin{matrix} \hat{\boldsymbol{x}} \in \widehat{\mathcal{R}}_k(\hat{\boldsymbol{x}}_t, \boldsymbol{u}^F_{0:T}), \\ \boldsymbol{w} \in \mathcal{W}, \\ \boldsymbol{e}, \boldsymbol{e}' \in \mathcal{E}_\theta \end{matrix} \right\}$$

for $k \in \{0, ..., T\}$. Furthermore, let

$$\mathcal{R}_k(\hat{\boldsymbol{x}}_t, \boldsymbol{u}^F_{0:T}) := \widehat{\mathcal{R}}_k(\hat{\boldsymbol{x}}_t, \boldsymbol{u}^F_{0:T}) \oplus \mathcal{E}_\theta \tag{7}$$

be the $k-$step reachable set of the true state $\boldsymbol{x}_t$. If $\boldsymbol{e}_{t:t+T+1} \subset \mathcal{E}_\theta$, then it holds that $\hat{\boldsymbol{x}}_{t+k} \in \widehat{\mathcal{R}}_k(\hat{\boldsymbol{x}}_t, \boldsymbol{u}^F_{0:T}) \subseteq \mathcal{R}_k(\hat{\boldsymbol{x}}_t, \boldsymbol{u}^F_{0:T})$ and $\boldsymbol{x}_{t+k} \in \mathcal{R}_k(\hat{\boldsymbol{x}}_t, \boldsymbol{u}^F_{0:T})$ for all $k \in \{0, ..., T+1\}$. Moreover, it holds that $\mathcal{R}_k(\hat{\boldsymbol{x}}_{t+1}, \boldsymbol{u}^F_{1:T}) \subseteq \mathcal{R}_{k+1}(\hat{\boldsymbol{x}}_t, \boldsymbol{u}^F_{0:T})$ for $k \in \{0, ..., T\}$.

To maintain feasibility of the fallback policy despite estimation errors and disturbances, we solve the following finite time robust optimal control problem online:

$$\begin{aligned} \underset{\substack{\boldsymbol{u}^F_{t:t+T|t} \subset \mathcal{P}_F, \\ \boldsymbol{u}_{t:t+T|t} \subset \mathcal{P}_N}}{\text{minimize}} \quad & C(\hat{\boldsymbol{x}}_t, \boldsymbol{u}_{t:t+T|t}, \boldsymbol{u}^F_{t:t+T|t}) \\ \text{subject to} \quad & \mathcal{R}_k(\hat{\boldsymbol{x}}_t, \boldsymbol{u}^F_{t:t+T|t}) \subseteq \mathcal{X} \ \forall k \in \{0, ..., T\}, \\ & \boldsymbol{u}^F_{t+k|t}(\boldsymbol{g}(\mathcal{R}_k(\hat{\boldsymbol{x}}_t, \boldsymbol{u}^F_{t:t+T|t}))) \subset \mathcal{U} \ \forall k \in \{0, ..., T\}, \\ & \mathcal{R}_{T+1}(\hat{\boldsymbol{x}}_t, \boldsymbol{u}^F_{t:t+T|t}) \subseteq \mathcal{X}_R, \\ & \boldsymbol{u}_{t|t}(\hat{\boldsymbol{x}}_t) = \boldsymbol{u}^F_{t|t}(\boldsymbol{y}_t). \end{aligned} \tag{8}$$

The MPC problem (8) robustly optimizes the trajectory of the robot along a $T+1$ step prediction horizon and maintains both a nominal policy sequence $\boldsymbol{u}_{t:t+T|t}$, and a fallback tube $\mathcal{R}_{0:t+T+1}(\hat{\boldsymbol{x}}_t, \boldsymbol{u}^F_{t:t+T|t})$. Let $\{\boldsymbol{u}^\star_{t+k|t}, \boldsymbol{u}^{F,\star}_{t+k|t}\}_{k=0}^T$ be an optimal collection of policy sequences for (8) at time step t. Executing the fallback policy sequence $\boldsymbol{u}^{F,\star}_{t:t+T|t}$ guarantees that we reach a given recovery set $\mathcal{X}_R$ within $T+1$ time steps for any disturbance sequence $\boldsymbol{w}_{t:t+T|t}$ and perception errors $\boldsymbol{e}_{t:t+T+1|t} \subset \mathcal{E}_\theta$. Because we ensure that the first inputs of both the nominal and the fallback policies are identical, i.e., that $\boldsymbol{u}^F_{t|t}(\boldsymbol{y}_t) = \boldsymbol{u}_{t|t}(\hat{\boldsymbol{x}}_t)$, we can guarantee that we can recover the system to $\mathcal{X}_R$ if we detect a fault at $t+1$ by applying the current fallback policy sequence $\boldsymbol{u}^{F,\star}_{t+1:t+T|t}$.

We assume the recovery set is invariant with respect to the estimator dynamics (6) in nominal conditions.

Assumption 4: We are given a *recovery policy* $\pi_R : \mathcal{Y} \to \mathcal{U}$ associated with a nonempty recovery set $\mathcal{X}_R$ under the estimate dynamics (6) in nominal conditions, so that $\mathcal{R}_1(\hat{\boldsymbol{x}}, \pi_R) \subseteq \mathcal{X}_R$ for all $\hat{\boldsymbol{x}} \in \mathcal{X}_R$.
Assumption 4 follows the classic assumption in the robust MPC literature—access to a terminal controller associated with a nonempty RPI set—that enables guarantees on persistent feasibility and constraint satisfaction [20], [21], [25], [36], except that we explicitly enforce that the recovery policy only uses fallback measurements $\boldsymbol{y}_t$. We note that Assumption 4 can be satisfied by designing a recovery policy (e.g., LQR or human-insight as in the drone landing example) and verifying whether a chosen set satisfies Definition 2 offline. Alternatively, we can compute $\mathcal{X}_R$ using existing algorithms for robust invariant set computation (e.g., see [36]), since under π_R and the assumption that $\boldsymbol{e} \in \mathcal{E}_\theta$, (6) is an autonomous system subject to bounded disturbances.

We choose the objective C in problem (8) to minimally interfere with the nominal operation of the robot by optimizing a disturbance free nominal trajectory as is common in the MPC literature (e.g., see [14], [20], [25], [36]). An alternative is to minimally modify the outputs of another controller [16], [19]. We develop our framework in generic terms in this section, so we provide a tractable reformulation of (8) for linear-quadratic systems with a fixed feedback gain based on classic tube MPC algorithms [20] in [35]. For nonlinear systems, it is common to approximate a solution via sampling (e.g., [37], [38]), so we also provide an approximate formulation using the PMPC algorithm [37] in [35], which combines uncertainty sampling with sequential convex programming.

```
Algorithm 1: Fallback-Safe MPC
Input: Initial state estimate $\hat{\boldsymbol{x}}_0$ such that
       (8) is feasible, runtime monitor $w:\mathbb{R}\to\{0,1\}$.
1 $t_{\text{fail}} \leftarrow \infty$
2 for $t\in\{0,1,...t_{\text{lim}}\}$ do
3     Observe $\hat{\boldsymbol{x}}_t$, $\boldsymbol{y}_t$, $a_t$
4     if $w(a_t)=1$ then
5         $t_{\text{fail}} \leftarrow \min\{t_{\text{fail}}, t\}$
6     end
7     Apply control input
      $\boldsymbol{u}_t = \begin{cases} \boldsymbol{u}^{\star}_{t|t}(\hat{\boldsymbol{x}}_t) & \text{if } t_{\text{fail}} > t \\ \boldsymbol{u}^{F,\star}_{t|t_{\text{fail}}-1}(\boldsymbol{y}_t) & \text{if } t_{\text{fail}} \le t < t_{\text{fail}}+T \\ \pi_R(\boldsymbol{y}_t) & \text{if } t \ge t_{\text{fail}}+T \end{cases}$
8 end
```

Here we assume access to a runtime monitor w to decide when we trigger the fallback; we construct a monitor with provable guarantees in §IV.

Definition 4 (Runtime Monitor): A runtime monitor $w:\mathbb{R}\to\{0,1\}$ is a function of the anomaly signal, where $w(a_t)=1$ implies the monitor raises an alarm.

We solve (8) online at each time step t and apply the first optimal control input in a receding horizon fashion. If the runtime monitor triggers, indicating a detection of a perception fault (i.e., $\boldsymbol{x}_t - \hat{\boldsymbol{x}}_t \notin \mathcal{E}_t$), we apply the previously computed fallback policy sequence until we reach the recovery set. Then, we revert to the known recovery policy. We summarize this procedure in Algorithm 1.

Let t_{fail} be the timestep at which the runtime monitor w triggers the fallback. As long as the runtime monitor does not miss a detection of a perception fault, i.e., that $\boldsymbol{x}_t - \hat{\boldsymbol{x}}_t \in \mathcal{E}_\theta$ for all $0 \le t < t_{\text{fail}}$, the MPC in (8) is recursively feasible. As a result, Algorithm 1 persistently satisfies state and input constraints in the presence of disturbances and estimation errors.

Theorem 1 (Fallback Safety): Consider the closed-loop system formed by the dynamics (1) and the Fallback-Safe MPC (Algorithm 1). Suppose that $\pi_R \in \mathcal{P}_F$, and that $\boldsymbol{x}_t - \hat{\boldsymbol{x}}_t \in \mathcal{E}_\theta$ for all $0 \le t < t_{\text{fail}}$. Then, if the Fallback-Safe MPC problem (8) is feasible at $t=0$ and $w(a_0)=0$, we have that 1) the MPC problem (8) is feasible for all $t < t_{\text{fail}}$ and that 2) the closed-loop system satisfies $\boldsymbol{x}_t \in \mathcal{X}$, $\boldsymbol{u}_t \in \mathcal{U}$ for all $t \ge 0$.

We emphasize that Theorem 1 simply recovers a standard recursive feasibility argument for the MPC scheme in the specific case in which a perception failure never occurs.

IV. CALIBRATING OOD DETECTORS WITH CONFORMAL INFERENCE

In the previous section, we developed the Fallback-Safe MPC framework, which guarantees safety under the condition that the perception is reliable at all time steps before we trigger the fallback. We could trivially ensure this is the case by setting $w(a)=1\ \forall a \in \mathbb{R}\setminus a_0$, so that the fallback always triggers, no matter the quality of the perception. However, a trivial runtime monitor will unnecessarily disrupt nominal operations, so it is not useful. Therefore, in this section, we aim to construct a runtime monitor $w:\mathbb{R}\to\{0,1\}$ that provably triggers with high probability when a perception fault occurs, but does not raise too many false alarms in practice. To do so, we adapt the conformal prediction algorithm in [34], which can only certify a prediction on a single test point, to retain a safety assurance when we sequentially query the runtime monitor online with the anomaly scores $a_0, a_1, ...$ generated during a test trajectory. Our procedure requires some ground truth data to calibrate the runtime monitor. As a shorthand, we use the notation τ to denote a trajectory with ground truth information, $\tau := (\{\boldsymbol{x}_i, \boldsymbol{u}_i, \hat{\boldsymbol{x}}_i, a_i\})_{i=0}^{t_{\text{lim}}} \in \mathcal{T}$.

Assumption 5 (Calibration Data): We have access to a trajectory dataset $\mathcal{D} = \{\tau_i\}_{i=0}^{N} \overset{\text{iid}}{\sim} P$ sampled independently and identically distributed (iid) from a trajectory distribution P.

The trajectory distribution P is a result of 1) the environment distribution P_ρ and 2) the controller we use to collect data [1]. Therefore, in general, when we deploy the Fallback-Safe MPC policy (Alg. 1), the resulting trajectory distribution P' may differ from P. The results we present in this section capture both the scenario where the environment distribution changes between data collection and deployment or where the data collection policy differs from the Fallback-Safe MPC.

Theorem 1 requires that we trigger the fallback at any time step before or when a perception fault occurs. Therefore, we must compare the step that the runtime monitor triggers an alarm, t_{fail}, with the first step that a perception fault occurs.

Definition 5 (Stopping Time): Let $t_{\text{stop}}: \mathcal{T} \to \mathbb{N}_{\ge 0}$ be the stopping time of a trajectory τ, defined as

$$t_{\text{stop}}(\tau) := \inf_{t\ge 0}\{t\ :\ (\boldsymbol{x}_t - \hat{\boldsymbol{x}}_t \notin \mathcal{E}_\theta)\ \vee\ (t = t_{\text{lim}})\}, \tag{9}$$

where t_{lim} is the episode time limit in Definition 1.

To guarantee safety as in Definition 1, our insight is that it is sufficient to guarantee that our runtime monitor raises an alarm at the *first* time step for which $\boldsymbol{e} \notin \mathcal{E}_\theta$ with high probability. Therefore, we can circumvent the need for a complex sequential analysis that accounts for the correlations between the runtime monitor's hypothesis tests over time. Instead, we can directly apply methods developed for i.i.d. samples to the dataset of stopping time observables $\mathcal{D}_{\text{stop}} := \{(\boldsymbol{x}^i_{t_{\text{stop}}(\tau_i)}, \hat{\boldsymbol{x}}^i_{t_{\text{stop}}(\tau_i)}, a^i_{t_{\text{stop}}(\tau_i)})\}_{i=0}^{N}$, since the stopping time observables are i.i.d. because $\mathcal{D}$ is i.i.d. We outline this approach in Algorithm 2, which adapts the conformal prediction algorithm in [34] to our sequential setting.

Lemma 2 (Conformal Calibration): Set a risk tolerance $\delta \in (0,1]$, and sample a deployment trajectory $\tau \sim P'$ by executing the Fallback-Safe MPC (Algorithm 1) using Algorithm 2 as runtime monitor w. Then, the false negative rate of Algorithm 2 is bounded as

$$\begin{aligned} &\text{Prob(False Negative)} := \\ &\text{Prob}\big(w(a_t)=0\ \forall t\in[0, t_{\text{stop}}(\tau)]\ \big|\ \boldsymbol{e}_{t_{\text{stop}}(\tau)} \notin \mathcal{E}_\theta\big) \le \\ &\quad \delta + \frac{1}{|\mathcal{A}|+1} + \text{TV}(P_{t_{\text{stop}}|\text{fault}}, P'_{t_{\text{stop}}|\text{fault}}), \end{aligned} \tag{10}$$

where $P_{t_{\text{stop}}|\text{fault}}$ is the distribution of $(\boldsymbol{x}, \hat{\boldsymbol{x}}, a)$ at t_{stop} conditioned on the event that $\boldsymbol{e}_{t_{\text{stop}}} \notin \mathcal{E}_\theta$ under a trajectory sampled from P, and $P'_{t_{\text{stop}}|\text{fault}}$ is the distribution of $(\boldsymbol{x}, \hat{\boldsymbol{x}}, a)$ at t_{stop} under a trajectory sampled from P', conditioned on both $\boldsymbol{e}_{t_{\text{stop}}} \notin \mathcal{E}_\theta$ and $t_{\text{fail}} \ge t_{\text{stop}}$. Here, $TV(\cdot,\cdot)$ denotes the total variation distance.

Lemma 2 shows that Algorithm 2 will issue a timely warning with probability at least $\delta + 1/(|\mathcal{A}|+1)$ without relying on properties of P (i.e., our guarantee is distribution-free), but that this guarantee degrades when a distribution shift occurs between the calibration runs in $\mathcal{D}$ and the test trial.

[1]For notational simplicity and without loss of generality, we consider $\boldsymbol{x}_0$ as a deterministic function of the environment variable ρ in this section.

Algorithm 2: Modification of [34] for Conformal Calibration of Runtime Monitor

```
Input: Dataset $\mathcal{D}=\{\tau_i\}_{i=0}^{N}\overset{\text{iid}}{\sim}P$,
       perception system, OOD detector,
       state uncertainty tolerance $\mathcal{E}_\theta$, risk tolerance
       $\delta\in(0,1]$, new test anomaly score $a_{\text{test}}\in\mathbb{R}$.
Output: 0 or 1
1 Compute the dataset
    of stopping states, estimates, and anomaly scores as
    $\mathcal{D}_{\text{stop}}:=\{(\boldsymbol{x}^i_{t_{\text{stop}}(\tau_i)},\hat{\boldsymbol{x}}^i_{t_{\text{stop}}(\tau_i)},a^i_{t_{\text{stop}}(\tau_i)})\}_{i=0}^{N}$.
2 Compute the set
    $\mathcal{A}:=\{a\,:\,\boldsymbol{x}-\hat{\boldsymbol{x}}\notin\mathcal{E}_\theta,\,(\boldsymbol{x},\hat{\boldsymbol{x}},a)\in\mathcal{D}_{\text{stop}}\}$.
3 Sample $U$ uniformly from
    $U\sim\{0,1,...,|\{a\in\mathcal{A}:a=a_{\text{test}}\}|\}$
4 Compute
    $q=\frac{|\{a\in\mathcal{A}:a>a_{\text{test}}\}|+U+1}{|\mathcal{A}|+1}$
5 if $q\leq 1-\delta$ then return 1 else return 0
```

Next, we leverage Lemma 2 to analyze the end-to-end safety of the system.

Theorem 2 (End-to-end Guarantee): Consider the closed-loop system formed by the dynamics (1) and the Fallback Safe MPC (Algorithm 1), using Algorithm 2 as the runtime monitor w. Then, if the MPC problem (8) is feasible at $t=0$ and $w(a_0)=0$, it holds that

$$\begin{aligned}&\text{Prob}(\boldsymbol{x}_t\in\mathcal{X},\,\boldsymbol{u}_t\in\mathcal{U}\ \forall t\in[0,t_{\text{lim}}])\geq\\ &1-\delta-\frac{1}{|\mathcal{A}|+1}-\text{TV}(P_{t_{\text{stop}}|\text{fault}},P'_{t_{\text{stop}}|\text{fault}}).\end{aligned}$$

Theorem 2 gives a general end-to-end guarantee on the safety of the Fallback-Safe MPC framework when we use Algorithm 2 as a runtime monitor. It is not possible to tightly bound the TV distance term in (10) without further assumptions. However, if 1) we use the Fallback-Safe MPC for data collection and 2) the environment distribution is fixed between data collection and deployment, then we can certify that we satisfy Definition 1:

Corollary 1: Suppose the environment distribution P_ρ is fixed between collecting $\mathcal{D}$ and the test trajectory, and that we collect the dataset $\mathcal{D}$ by running the Fallback-Safe MPC and a runtime monitor using privileged information, that is, $w(\cdot):=1-\mathbf{1}\{\boldsymbol{e}_t\in\mathcal{E}_\theta\}$. Then, it holds that $P_{t_{\text{stop}}|\text{fault}}=P'_{t_{\text{stop}}|\text{fault}}$. Therefore, 1) we satisfy state and input constraints during data collection with probability 1, and 2) we satisfy Definition 1 with probability at least $1-\delta-\frac{1}{|\mathcal{A}|+1}$ during a test trajectory.

Corollary 1 informs the following two-step procedure to yield a provable end-to-end safety guarantee on a fixed environment distribution P_ρ. We use this procedure in §V. First, collect $\mathcal{D}$ using the Fallback-Safe MPC with a ground-truth supervisor $w(\cdot):=1-\mathbf{1}\{\boldsymbol{e}_t\in\mathcal{E}_\theta\}$, then deploy the Fallback-Safe MPC with Algorithm 2 as the runtime monitor. We can then satisfy Definition 1 for a risk tolerance δ by evaluating Algorithm 2 using $\delta'=\delta-1/(|\mathcal{A}|+1)$. As long as we have sufficient data on failure modes, that is, when $(1/\delta)-1\leq|\mathcal{A}|$, our runtime monitor will exhibit nontrivial behavior (i.e., that w does not always output 1).

V. SIMULATIONS

In this section, we first simulate a simplified example of a quadrotor to illustrate the behavior of the Fallback-Safe MPC framework. We then demonstrate the efficacy of the conformal algorithm, and the resulting end-to-end safety guarantee, in the photo-realistic X-Plane 11 aircraft simulator. For a detailed description of our simulations, including e.g., the specific cost functions used, we refer the reader to [35].

Planar Quadrotor: We consider a planar version of the quadrotor dynamics for simplicity, with 2D pose $\boldsymbol{p}=[x,y,\theta]^T$, state $\boldsymbol{x}=[\boldsymbol{p}^T,\dot{\boldsymbol{p}}^T]^T$, and front and rear input thrust inputs $\boldsymbol{u}=[u_f,u_r]^T$ [39]. We linearize the the dynamics around $\bar{\boldsymbol{x}}=0$ and $\bar{\boldsymbol{u}}=\frac{mg}{2}[1,1]^T$, and discretize the dynamics using Euler's method with a time step of $\mathrm{d}t=0.15s$. The drone is subject to bounded wind disturbances, so that the drone may drift with $\approx 0.33m/s$ in the $x-$direction without actuation. In our example, the drone has internal sensors to estimate its orientation and velocity, so that $\boldsymbol{y}=[\theta,\dot{\boldsymbol{p}}^T]^T$. The drone estimates its $xy-$position using a hypothetical vision sensor. To do so, we nominally simulate that the perception system's $xy-$position estimate is within a 10cm-wide box around the true position, and perfectly outputs $\boldsymbol{y}$. When the vision system fails, we randomly sample the $xy-$position within the range $(-10,10)$. In these simulations we give the Fallback-Safe MPC a perfect runtime monitor, so that $w(\cdot)=1-\mathbf{1}\{\boldsymbol{e}_t\in\mathcal{E}\}$. We implement the Fallback-Safe MPC using the tube MPC formulation in [35].

First, in Fig. 3, we simulate a scenario where the drone attempts a vision-based landing at the origin. Here, the state constraint is not to crash into the ground ($y\leq 0$). We set the recovery policy to $\pi_R(\boldsymbol{y}):=\epsilon+K\boldsymbol{y}$, where K stabilizes the orientation θ around 0 and ϵ is a small offset to continually fly upward, choosing the recovery set to allow the drone to fly away starting from a sufficient altitude. We verify using reachability analysis [36, Sec. 10.2] that under the recovery policy, the recovery set is invariant under both the estimator dynamics (6) in nominal conditions and the state dynamics (1), so the Fallback-Safe MPC is recursively feasible by Theorem 1. We compare the Fallback-Safe MPC with a naive tube MPC that optimizes only a single trajectory and assumes perception is always reliable (i.e., it assumes perception errors always satisfy (5)) As shown in Fig. 3, the Fallback-Safe MPC plans fallback trajectories that safely abort the landing and fly away into open space. In contrast, the naive tube MPC does

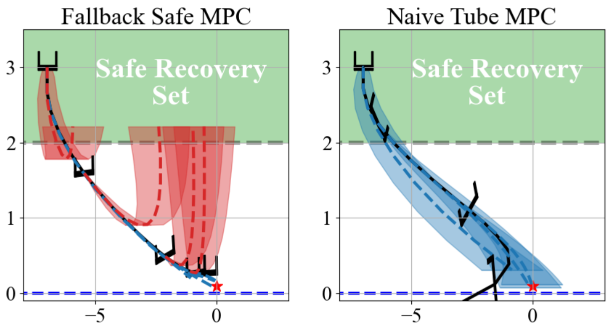

Fig. 3. Trajectories of the planar quadrotor in the xy-plane. The realized quadrotor trajectories are in black, and the icons show the orientation of the quadrotor at every $k=7$ time steps. The safe recovery set is highlighted in green. The blue-dashed line indicate the state constraint. Left: In red, we plot the predicted reachable sets of the fallback strategy and in blue, we plot the predicted nominal trajectories, both at $k=7$ step intervals. Right: In blue, we plot the predicted reachable tubes of the Naive Tube MPC at k step intervals.

not reason about perception faults, and crashes badly when the perception fails starting at $t = 10\mathrm{dt}$. Therefore, this example demonstrates the necessity of planning with a fallback.

Secondly, we simulate a scenario where the drone must navigate towards an in-air xy goal location while remaining within a box in the xy-plane. Here, when the drone loses its vision, it is no longer possible to avoid the boundaries of $\mathcal{X}$ using only the fallback measurement $\boldsymbol{y}$. Instead, as in the example in Fig. 1, our recovery set is to land the drone. To model the drone as having landed, we modify the dynamics to freeze the state for all remaining time once the state $\boldsymbol{x}$ enters the $y \leq 0$ region with low velocity. However, in this example, the drone must cross an unsafe ground region, such as the busy road in the example in Fig. 1, specified as the region of states with $|x| < 1.5m$, $y \leq 0$. Therefore, we take the recovery set $\mathcal{X}_R$ as all landed states with $|x| \geq 1.5m$. Clearly, $\mathcal{X}_R$ is a safe recovery set for $\pi_R(\boldsymbol{y}) = 0$, under the true dynamics (1).[2] For the drone to cross the road, we need to maintain recoverability with respect to either of the two disjoint recovery sets. We compare our approach with another naive baseline, which we label the Unsafe Fallback MPC, that executes a nominal MPC policy and naively tries to compute a fallback trajectory post-hoc, using the previous estimate before the fault occurred. As shown in Fig. 4, our Fallback-Safe MPC first maintains feasibility of the fallback with respect to the rightmost recovery set, slows down, and then switches to the leftmost recovery set once a feasible trajectory crossing the road is found. In contrast, the Unsafe Fallback MPC does not maintain the feasibility of the fallback by modifying nominal operations, and is forced to crash land in the unsafe ground region (rather than throwing an infeasibility error, our implementation relies on slack variables). Therefore, this example illustrates that it is necessary to modify nominal operations to maintain the feasibility of a fallback.

X-Plane Aircraft Simulator: Finally, we evaluate the conformal prediction Algorithm 2 and the end-to-end safety

[2] We note that in this example, $\mathcal{X}_R$ is not RPI under the state estimate dynamics (6) in nominal conditions, because the estimation error bound allows $\hat{\boldsymbol{x}}$ to leave the $\mathcal{X}_R$ even if $\boldsymbol{x} \in \mathcal{X}_R$. To retain the safety guarantee, we also trigger the fallback if (8) is infeasible, a simple fix first proposed in [40]. We did not observe recursive feasibility issues in the simulations.

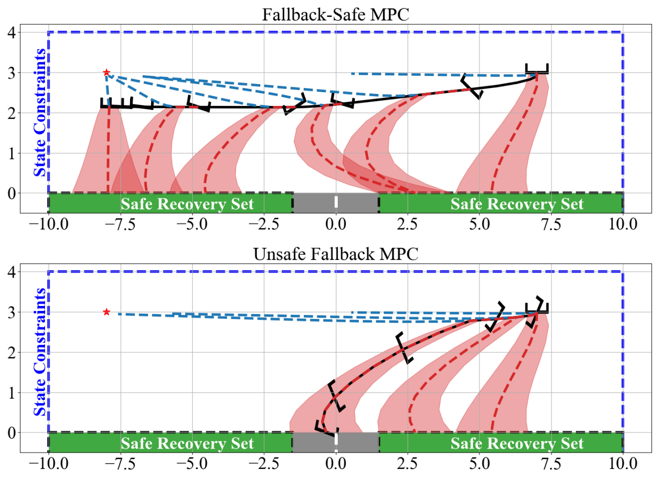

Fig. 4. Trajectories of the planar quadrotor in the xy-plane. The disjoint safe recovery sets are highlighted in green, the unsafe ground region (e.g., a road), $|x| < 1$, $y \leq 0$, is in gray. Both the top and bottom figure follow the layout in Fig. 3 (left).

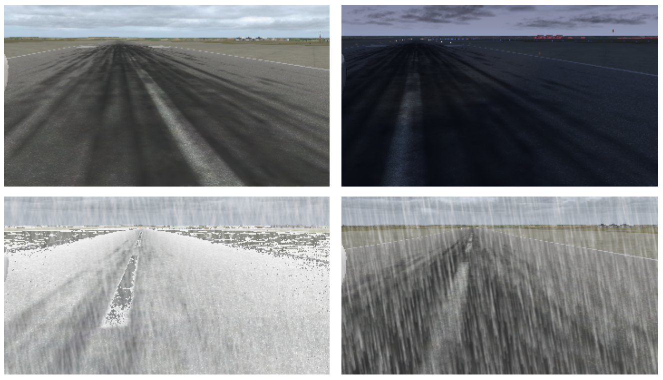

Fig. 5. Simulated environments in the X-Plane 11 simulator. Top left: Morning, no weather. Top right: Night, no weather. Bottom left: Afternoon, snowing. Bottom right: Afternoon, raining.

guarantee of our framework using the photo-realistic X-Plane 11 simulator. We simulate an autonomous aircraft taxiing down a runway with constant reference velocity, while using a DNN to estimate its heading error (HE) θ and center-line distance (cross-track error (CTE)) y from an outboard camera feed. Here, internal encoders always correctly output the velocity v, so that $\boldsymbol{x} = [y, \theta, v]$ and $\boldsymbol{y} = v$. The aircraft must not leave the runway, given by the state constraint $|y| \leq 6\mathrm{m}$.

We train the DNN perception model on 4×10^4 labeled images collected only in morning, clear sky weather, but we deploy the system in a context P_ρ where it may experience a variety of weather conditions (depicted in Fig. 5). We parameterize the environment $\rho := (\text{weather type}, t_{\text{start}}, \text{severity})$ as a triplet indicating the weather type, severity level, and starting time from which the visibility starts to degrade, so that under the environment distribution P_ρ, we randomly sample an environment that starts with clear-skies and high visibility, but may cause OOD errors during an episode. As shown in Fig. 2, heavy weather degrades the perception significantly. The fallback is to brake the aircraft to a stop, where the stopped states are invariant[2] under $\pi_R(\boldsymbol{y}) := 0$. As in [8], we train an autoencoder alongside the DNN on the morning, clear sky data and use the reconstruction error as the anomaly signal a_t. We define the perception error set to include at most 7 degree HE and 1.3m CTE.

We record 100 training trajectories using the Fallback-Safe MPC (8) and a ground-truth supervisor $w(\cdot) = 1 - \mathbf{1}\{\boldsymbol{e}_t \in \mathcal{E}\}$ to calibrate Algorithm 2, and then evaluate on 900 test trajectories with environments sampled i.i.d. from P_ρ and using Algorithms 1-2. This ensures we satisfy Definition 1 by Corollary 1. We compute the empirical false positive and false negative rate when we evaluate Algorithm 2 with various values of $\delta \in [0, 1]$ in Fig. 6 (left). As Fig. 6 (left) shows, the FNR of Algorithm 2 satisfies Lemma 2's $\delta' = \delta + 1/(|\mathcal{A}| + 1)$ bound for all values of δ, validating our guarantees. Moreover, the false positive rate, the rate at which we incorrectly trigger the fallback, is near 0 for risk tolerances as small as $\delta' = 5\%$. This shows that algorithm 2 is highly sample efficient and not overly conservative, since it hardly issues incorrect alarms with orders of magnitudes fewer samples than we needed to train the perception. In Fig. 6 (right), we control the system with an end-to-end safety guarantee of $\delta' = .1$ using our framework and observe no constraint violations. For the trajectories in which we triggered the fallback, Fig. 6 (right) shows that over 80% would have led to an aircraft failure had we not interfered. This shows that our framework

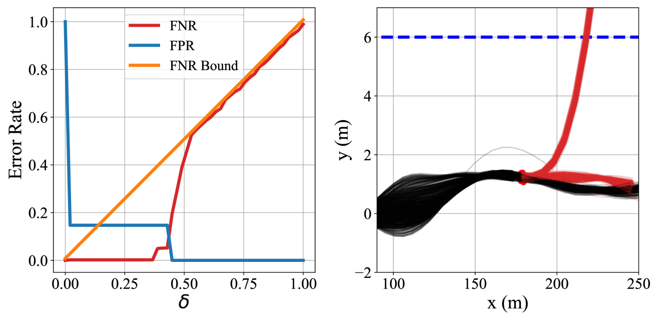

Fig. 6. Left: FNR Bound indicates the $\delta + 1/(|\mathcal{A}|+1)$ bound on the FNR from Lemma 2. The FPR indicates the empirical rate at which we trigger the fallback without a perception fault ever occurring. The FNR indicates the empirical rate at which a perception fault occurs before we trigger the fallback. Right: closed-loop trajectories of the aircraft are in black. For trajectories in which the fallback triggered, we plot a red dot where the aircraft stopped, and plot in red the trajectory that would have occurred had we not triggered the fallback.

is effective at avoiding robot failures, and experiences few unnecessary interruptions with an effective OOD detection heuristic like the autoencoder reconstruction loss.

VI. Conclusion

In this work, we have formalized the design of safety-preserving fallback strategies under perception failures by ensuring the feasibility of a fallback plan with respect to a safe recovery set, a subset of the state space that we can make invariant without full knowledge of the state. Similar to the terminal invariant in a standard MPC, we have demonstrated that recovery sets can readily be identified offline. Our simulations also showed that the calibration procedure, which enables strong safety assurances, is particularly amenable to limited data collection pre-deployment. Still, we observe that our runtime monitor occasionally triggers the fallback when the closed-loop system would not have violated safety constraints because we rely on an imperfect heuristic for OOD detection. Therefore, future work should investigate how to tune runtime monitors to only detect downstream failures more effectively. In addition, future work may explore more complex statistical analysis on the runtime monitor since our framework currently does not permit a switch back to nominal operations after a fault occurs.

References

[1] R. Sinha, A. Sharma, S. Banerjee *et al.*, "A system-level view on out-of-distribution data in robotics," *arXiv preprint arXiv:2212.14020*, 2022, Available at https://arxiv.org/abs/2212.14020.
[2] R. Geirhos, J.-H. Jacobsen, C. Michaelis *et al.*, "Shortcut learning in deep neural networks," *Nature Machine Intelligence*, Nov 2020.
[3] A. Torralba and A. A. Efros, "Unbiased look at dataset bias," in *CVPR*, 2011.
[4] S. A. Seshia, D. Sadigh, and S. S. Sastry, "Towards verified artificial intelligence," 2020. [Online]. Available: https://arxiv.org/abs/1606.08514
[5] Q. M. Rahman, P. Corke, and F. Dayoub, "Run-time monitoring of machine learning for robotic perception: A survey of emerging trends," *IEEE Access*, 2021.
[6] M. Salehi, H. Mirzaei, D. Hendrycks *et al.*, "A unified survey on anomaly, novelty, open-set, and out-of-distribution detection: Solutions and future challenges," 2021. [Online]. Available: https://arxiv.org/abs/2110.14051
[7] L. Ruff, J. R. Kauffmann, R. A. Vandermeulen *et al.*, "A unifying review of deep and shallow anomaly detection," *Proceedings of the IEEE*, 2021.
[8] C. Richter and N. Roy, "Safe visual navigation via deep learning and novelty detection," in *RSS*, July 2017.
[9] A. Filos, P. Tigas, R. McAllister *et al.*, "Can autonomous vehicles identify, recover from, and adapt to distribution shifts?" in *ICML*, ser. ICML'20, 2020.
[10] R. McAllister, G. Kahn, J. Clune *et al.*, "Robustness to out-of-distribution inputs via task-aware generative uncertainty," in *ICRA*, 2019.
[11] T. Guffanti and S. D'Amico, "Passively-safe and robust multi-agent optimal control with application to distributed space systems," 2023. [Online]. Available: https://arxiv.org/abs/2209.02096
[12] D. A. Marsillach, S. Di Cairano, and A. Weiss, "Abort-safe spacecraft rendezvous in case of partial thrust failure," in *CDC*, 2020.
[13] J. P. Alsterda, M. Brown, and J. C. Gerdes, "Contingency model predictive control for automated vehicles," in *ACC*, 2019.
[14] L. Brunke, M. Greeff, A. W. Hall *et al.*, "Safe learning in robotics: From learning-based control to safe reinforcement learning," *An. Rev. CRAS*, 2022.
[15] K. P. Wabersich and M. N. Zeilinger, "Linear model predictive safety certification for learning-based control," in *CDC*, 2018.
[16] R. Cheng, G. Orosz, R. M. Murray *et al.*, "End-to-end safe reinforcement learning through barrier functions for safety-critical continuous control tasks," in *AAAI*, ser. AAAI'19/IAAI'19/EAAI'19, 2019.
[17] J. F. Fisac, A. K. Akametalu, M. N. Zeilinger *et al.*, "A general safety framework for learning-based control in uncertain robotic systems," *IEEE TAC*, 2019.
[18] K. Leung, E. Schmerling, M. Zhang *et al.*, "On infusing reachability-based safety assurance within planning frameworks for human-robot vehicle interactions," *IJRR*, 2020.
[19] L. Brunke, S. Zhou, and A. P. Schoellig, "Robust predictive output-feedback safety filter for uncertain nonlinear control systems," in *CDC*, 2022.
[20] D. Mayne, S. Raković, R. Findeisen *et al.*, "Robust output feedback model predictive control of constrained linear systems," *Automatica*, 2006.
[21] J. Lorenzetti and M. Pavone, "A simple and efficient tube-based robust output feedback model predictive control scheme," in *ECC*, 2020.
[22] C. Løvaas, M. M. Seron, and G. C. Goodwin, "Robust output-feedback model predictive control for systems with unstructured uncertainty," *Automatica*, 2008.
[23] J. Köhler, M. A. Müller, and F. Allgöwer, "Robust output feedback model predictive control using online estimation bounds," 2021. [Online]. Available: https://arxiv.org/abs/2105.03427
[24] R. Findeisen, L. Imsland, F. Allgower *et al.*, "State and output feedback nonlinear model predictive control: An overview," *EJC*, 2003.
[25] P. J. Goulart and E. C. Kerrigan, "A method for robust receding horizon output feedback control of constrained systems," in *Proceedings of the 45th IEEE Conference on Decision and Control*, 2006.
[26] S. Dean, N. Matni, B. Recht *et al.*, "Robust guarantees for perception-based control," 2019.
[27] G. Chou, N. Ozay, and D. Berenson, "Safe output feedback motion planning frommages viaearned perception modules andontraction theory," in *Algorithmic Foundations of Robotics XV*, 2023.
[28] B. Ichter, B. Landry, E. Schmerling *et al.*, "Perception-aware motion planning via multiobjective search on gpus," in *Robotics Research*, 2020.
[29] A. N. Angelopoulos and S. Bates, "A gentle introduction to conformal prediction and distribution-free uncertainty quantification," 2022.
[30] V. N. Balasubramanian, S.-S. Ho, and V. Vovk, *Conformal Prediction for Reliable Machine Learning*. Morgan Kaufmann, 2014.
[31] R. F. Barber, E. J. Candes, A. Ramdas *et al.*, "Conformal prediction beyond exchangeability," 2023.
[32] R. J. Tibshirani, R. Foygel Barber, E. Candes *et al.*, "Conformal prediction under covariate shift," in *NeurIPS*, H. Wallach, H. Larochelle, A. Beygelzimer *et al.*, Eds., 2019.
[33] R. Luo, R. Sinha, A. Hindy *et al.*, "Online distribution shift detection via recency prediction," *arXiv preprint arXiv:2211.09916*, 2023.
[34] R. Luo, S. Zhao, J. Kuck *et al.*, "Sample-efficient safety assurances using conformal prediction," in *Algorithmic Foundations of Robotics XV*, 2023.
[35] R. Sinha, E. Schmerling, and M. Pavone, "Closing the loop on runtime monitors with Fallback-Safe MPC," *arXiv preprint arXiv:2309.08603*, 2023, Available at https://arxiv.org/abs/2309.08603.
[36] F. Borrelli, A. Bemporad, and M. Morari, *Predictive control for linear and hybrid systems*, 2017.
[37] R. Dyro, J. Harrison, A. Sharma *et al.*, "Particle mpc for uncertain and learning-based control," in *IROS*, 2021.
[38] T. Lew, L. Janson, R. Bonalli *et al.*, "A simple and efficient sampling-based algorithm for reachability analysis," in *L4DC*, 2022.
[39] R. Tedrake, "Underactuated robotics: Algorithms for walking, running, swimming, flying, and manipulation," 2021, Available at http://underactuated.mit.edu.
[40] T. Koller, F. Berkenkamp, M. Turchetta *et al.*, "Learning-based model predictive control for safe exploration," in *CDC*, 2018.