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.**

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.

In this work we examine vision-based control settings, where a deep neural-network (DNN) is used to extract task-relevant information from a high-dimensional image observation, and when this DNN fails, access to this information is lost. For example, the drone in Fig. 1 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 interventions may be adopted. However, in order to integrate such monitors into a perception and control stack and derive end-to-end certificates on the safety of the combined 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, a naively specified fallback can introduce additional hazards.

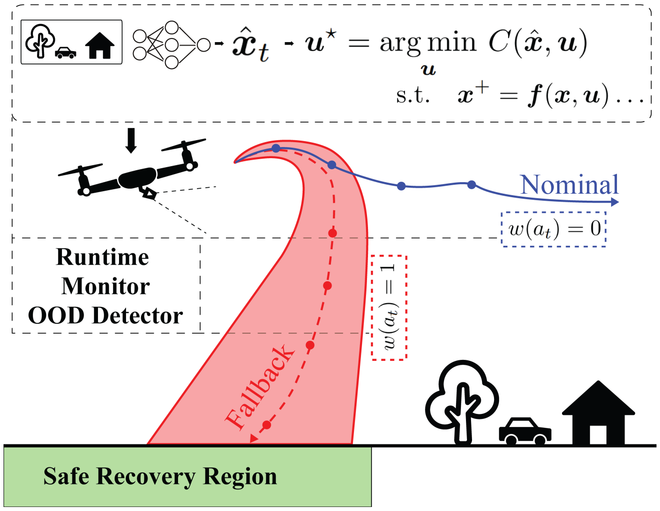
$\hat{\boldsymbol{x}}_t$
$\boldsymbol{u}^\star = \arg\min_{\boldsymbol{u}} C(\hat{\boldsymbol{x}}, \boldsymbol{u})$
s.t. $\boldsymbol{x}^+ = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}) \dots$
Nominal
$w(a_t) = 0$
Runtime
Monitor
OOD Detector
$w(a_t) = 1$
Fallback
Safe Recovery Region

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 w to trigger a fallback strategy when the perception system is unreliable. To do so, we calibrate heuristic OOD detection scores a_t 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 region using a robust MPC.

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. In this 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 conditions). 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). 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

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.

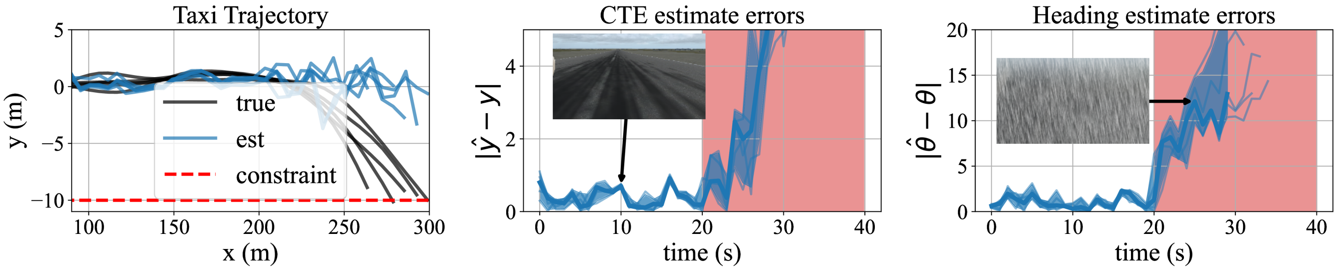

Fig. 2. 20 simulated trajectories of an autonomous aircraft taxiing down a runway. A DNN – trained on data collected in morning, clear-sky weather – estimates the cross-track error (CTE) and the heading error (HE) used by a tracking controller from vision. At the start of the trajectories, the weather 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.

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.

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 assumed to 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. Moreover, as illustrated in Fig. 1, naively executing a specified fallback (e.g., landing the drone) can create additional hazards. 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 region 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). To account for these discrete information modes, our insight is that we can often specify *recovery regions* to fallback into, safe regions of the state space that we can make invariant without full knowledge of the state. For example, 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 model outputs. 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, conditions that degrade perception systems are oftentimes externally caused by a changing environment, like the changing weather in Fig. 2, and perception systems may exhibit arbitrarily poor error behavior on OOD inputs. 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 (including additional approaches for coping with OOD data), appendices, further experimental details, and 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 region, which formalizes the intuitive idea that there often exist safe subsets of the state space that are invariant under a recovery policy that does not rely on the ML-enabled perception.
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. We do so by adapting an existing conformal inference algorithm [34], which only guarantees coverage on individual samples, to a sequential decision-making setting.

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 $o >> n$. The system operates in an environment $\rho \sim P_\rho$ drawn from a deployment context P_ρ which determines the joint distribution p_ρ of the process that generates disturbances and observations.

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 $t_{\mathrm{lim}} \in \mathbb{N}_{\geq 0}$, the system (1) is* safe *if*

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

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 in that 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 $\hat{\boldsymbol{x}}_t := \mathrm{perception}(\boldsymbol{z}_{0:t})$.*

Crucially, the deployment context P_ρ may be misaligned with the distribution that generated the data on which the learned components in the perception system were trained, and these learned components will behave erratically on OOD observations. 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. But, 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 $a_t = \mathrm{anomaly}(\boldsymbol{z}_{0:t}) \in \mathbb{R}$ 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, after which we transition to a fallback policy $\pi : \mathcal{Y} \rightarrow \mathcal{U}$ 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 regions*, 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 Regions

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 regions of the state space in which the robot is always safe under some $\pi_R : \mathcal{Y} \rightarrow \mathcal{U}$.

Definition 2 (Recovery Region, Policy)**.** *A set $\mathcal{X}_R \subseteq \mathcal{X}$ is a* safe recovery region *under a given recovery policy $\pi_R : \mathcal{Y} \rightarrow \mathcal{U}$ 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 region 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*, or that the system operates* nominally*, when*

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

We say the perception system is unreliable*, operates* off-nominally*, or experiences a* perception fault*, when* $\mathbf{e}_t \notin \mathcal{E}_\theta$.

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. For example, we choose the perception error set as a warped infinity norm box $\mathcal{E}_\theta = \{\boldsymbol{e} \in \mathbb{R}^n : \|\mathbf{A}\boldsymbol{e}\|_\infty \leq \alpha\}$ for some $\theta = (\mathbf{A}, \alpha)$ consisting of a matrix $\mathbf{A}$ and bound $\alpha \in \mathbb{R}$ in §V. To pick the hyperparameter θ, 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. In practice, one should aim to choose $\mathcal{E}_\theta$ at least to contain the perception error in reasonable conditions, after which a trade-off between conservativeness in nominal operation and eagerness to trigger the fallback must be made.

Then, in nominal conditions, we control the state estimates $\hat{\boldsymbol{x}}_t$ with robust MPC in a way that minimally modifies 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 $\boldsymbol{u}^F_{t:t+T} \subset \mathcal{P}_F \subset \{\pi : \mathbb{R}^r \to \mathbb{R}^m\}$, 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* $\boldsymbol{u}^F_{1:T} \subset \mathcal{P}_F$ *from timestep* t *to* $t+T$. *Define the* $k-$*step reachable sets of the estimate* $\hat{\boldsymbol{x}}_t$ *recursively as* $\widehat{\mathcal{R}}_0(\hat{\boldsymbol{x}}_t, \boldsymbol{u}^F_{1:T}) := \{\hat{\boldsymbol{x}}_t\}$ *and*

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

for $k \in \{0, \ldots, T\}$. *Furthermore, let* $\mathcal{R}_k(\hat{\boldsymbol{x}}_t, \boldsymbol{u}^F_{1:T}) := \widehat{\mathcal{R}}_k(\hat{\boldsymbol{x}}_t, \boldsymbol{u}^F_{1:T}) \oplus \mathcal{E}_\theta$ *be the* $k-$*step reachable set of the true state* $\boldsymbol{x}_t$. *If* $\mathbf{e}_{t:t+T+1} \subset \mathcal{E}_\theta$, *then it holds that* $\hat{\boldsymbol{x}}_{t+k} \in \widehat{\mathcal{R}}_k(\hat{\boldsymbol{x}}_t, \boldsymbol{u}^F_{1:T}) \subseteq \mathcal{R}_k(\hat{\boldsymbol{x}}_t, \boldsymbol{u}^F_{1:T})$ *and* $\boldsymbol{x}_{t+k} \in \mathcal{R}_k(\hat{\boldsymbol{x}}_t, \boldsymbol{u}^F_{1:T})$ *for all* $k \in \{0, \ldots, T+1\}$. *Moreover, it holds that* $\mathcal{R}_k(\hat{\boldsymbol{x}}_{t+1}, \boldsymbol{u}^F_{2:T}) \subseteq \mathcal{R}_{k+1}(\hat{\boldsymbol{x}}_t, \boldsymbol{u}^F_{1:T})$ *for* $k \in \{0, \ldots, T\}$.

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

$$\begin{aligned} \underset{\substack{\boldsymbol{u}^F_{t:t+T|t} \subset \mathcal{P}_F, \\ \boldsymbol{u}_{t:t+T|t} \subset \mathcal{P}_N}}{\text{minimize}} \quad & C(\hat{\boldsymbol{x}}_t, \boldsymbol{u}_{t:t+T|t}, \boldsymbol{u}^F_{t:t+T|t}) \\ \text{subject to} \quad & \mathcal{R}_k(\hat{\boldsymbol{x}}_t, \boldsymbol{u}^F_{t:t+T|t}) \subseteq \mathcal{X} \ \forall k \in \{0, \ldots, T\}, \\ & \boldsymbol{u}^F_{t+k|t}(\boldsymbol{g}(\mathcal{R}_k(\hat{\boldsymbol{x}}_t, \boldsymbol{u}^F_{t:t+T|t}))) \subset \mathcal{U} \ \forall k \in \{0, \ldots, T\}, \\ & \mathcal{R}_{T+1}(\hat{\boldsymbol{x}}_t, \boldsymbol{u}^F_{t:t+T|t}) \subseteq \mathcal{X}_R, \\ & \boldsymbol{u}_{t|t}(\hat{\boldsymbol{x}}_t) = \boldsymbol{u}^F_{t|t}(\boldsymbol{y}_t). \end{aligned} \tag{7}$$

The MPC problem (7) 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 (7) at time step t. Executing the fallback policy sequence $\boldsymbol{u}^{F,\star}_{t:t+T|t}$ guarantees that we reach a given recovery region $\mathcal{X}_R$ 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 $\boldsymbol{u}^{F,\star}_{t:t+T|t}$.

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 recovery region* $\mathcal{X}_R$ *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 computing $\mathcal{X}_R$ 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 (7) 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 provide a tractable reformulations of (7) for linear-quadratic systems with a fixed feedback gain based on classic tube MPC algorithms in [35]. For nonlinear systems, it is common to approximate a solution via sampling (e.g., [37],

Algorithm 1: Fallback-Safe MPC

```
Input: Initial state estimate $\hat{\boldsymbol{x}}_0$ such that
       (7) is feasible, runtime monitor $w:\mathbb{R}\rightarrow\{0,1\}$.
$t_{\text{fail}} \leftarrow \infty$
for $t\in[0,t_{\text{lim}}]$ do
    Observe $\hat{\boldsymbol{x}}_t$, $\boldsymbol{y}_t$, $a_t$
    if $w(a_t)=1$ then
        $t_{\text{fail}} \leftarrow \min\{t_{\text{fail}},\, t\}$
    end
    Apply control input
        $\boldsymbol{u}_t = \begin{cases} \boldsymbol{u}^{\star}_{t|t}(\hat{\boldsymbol{x}}_t) & \text{if} \quad t_{\text{fail}} > t \\ \boldsymbol{u}^{F,\star}_{t|t_{\text{fail}}-1}(\boldsymbol{y}_t) & \text{if} \quad t_{\text{fail}} \leq t < t_{\text{fail}}+T \\ \pi_R(\boldsymbol{y}_t) & \text{if} \quad t \geq t_{\text{fail}}+T \end{cases}$
end
```

[38]), so we provide an approximate formulation using the PMPC algorithm [37] in [35], which combines uncertainty sampling with sequential convex programming.

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* $w:\mathbb{R}\rightarrow\{0,1\}$ *is a function of the anomaly signal, where* $w(a_t)=1$ *implies the monitor raises an alarm.*

We solve (7) 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. We prove the MPC in (7) is recursively feasible, and thereby that Algorithm 1 is safe, as long as the runtime monitor does not miss a detection of a perception fault.

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 the runtime monitor* w *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* $0\leq t<t_{\text{fail}}$*. Then, if the Fallback-Safe MPC problem (7) is feasible at* $t=0$ *and* $w(a_0)=0$*, we have that 1) the MPC problem (7) 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* $t\geq 0$*.*

We emphasize that if we know a perception failure never occurs, Theorem 1 simply recovers a standard recursive feasibility argument for the MPC scheme.

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 $w:\mathbb{R}\rightarrow\{0,1\}$ 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 context P_ρ and 2) the controller we use to collect data. 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 context 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* $t_{\text{stop}}:\mathcal{T}\rightarrow\mathbb{N}_{\geq 0}$ *be the stopping time of a trajectory* τ*, defined as*

$$t_{\text{stop}}(\tau):=\inf_{t\geq 0}\{t\ :\ (\boldsymbol{x}_t-\hat{\boldsymbol{x}}_t\notin\mathcal{E}_\theta)\ \vee\ (t=t_{\text{lim}})\},\tag{8}$$

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*

$$\begin{aligned}&\text{Prob(False Negative)}:=\\&\text{Prob}\big(w(a_t)=0\ \forall t\in[0,t_{\text{stop}}(\tau)]\ |\ \boldsymbol{e}_{t_{\text{stop}}(\tau)}\notin\mathcal{E}_\theta\big)\leq\\&\quad\delta+\frac{1}{|\mathcal{A}|+1}+\text{TV}(P_{t_{\text{stop}}|\text{fault}},P'_{t_{\text{stop}}|\text{fault}}),\end{aligned}\tag{9}$$

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* $t_{\text{fail}}\geq t_{\text{stop}}$*. Here,* $TV(\cdot,\cdot)$ *denotes the total variation distance.*

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

```
Input: Dataset $\mathcal{D} = \{\tau_i\}_{i=1}^N \overset{\text{iid}}{\sim} P$,
       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
Compute the dataset
  of stopping states, estimates, and anomaly scores as
    $\mathcal{D}_{\text{stop}} := \{(\boldsymbol{x}^i_{t_{\text{stop}}(\tau_i)}, \hat{\boldsymbol{x}}^i_{t_{\text{stop}}(\tau_i)}, a^i_{t_{\text{stop}}(\tau_i)})\}_{i=0}^N.$
Compute the set
    $\mathcal{A} := \{a \,:\, \boldsymbol{x} - \hat{\boldsymbol{x}} \notin \mathcal{E}_\theta,\ (\boldsymbol{x}, \hat{\boldsymbol{x}}, a) \in \mathcal{D}_{\text{stop}}\}.$
Sample $U$ uniformly from
    $U \sim \{0,1,...,|\{a \in \mathcal{A} : a = a_{\text{test}}\}|\}$
Compute
    $q = \frac{|\{a \in \mathcal{A} : a > a_{\text{test}}\}| + U + 1}{|\mathcal{A}| + 1}$
if $q \leq 1 - \delta$ then return 1 else return 0
```

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.

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 (7) 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 (9) 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 context 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 1) we are safe during data collection with probability* 1*, and 2) that we are safe with probability at least $1 - \delta - \frac{1}{|\mathcal{A}|+1}$ during a test trajectory. This is because in this setting, it holds that $P_{t_{\text{stop}}|\text{fault}} = P'_{t_{\text{stop}}|\text{fault}}$ in distribution.*

Corollary 1 gives us a foothold to design algorithms with a provable end-to-end safety guarantee when we aim to deploy an autonomous system in a fixed context P_ρ: 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). We use this procedure in §V.

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, 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 $\mathrm{dt} = 0.15s$. 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 $10\mathrm{cm}$-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 ($y \leq 0$). The fallback is for the drone to abort the landing and fly away at a sufficient altitude. We note that under the recovery policy $\pi_R(\boldsymbol{y}) := \epsilon + K\boldsymbol{y}$, where K stabilizes the orientation θ around 0 and ϵ is a small offset to continually fly upward, 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 does not anticipate vision failures. 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 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

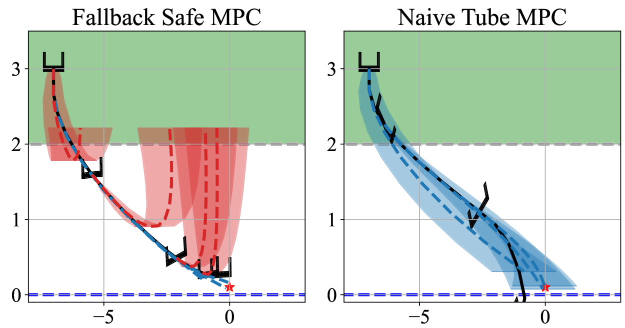

Fig. 3. Trajectories of the planar quadrotor in the xy-plane. The realized quadrotor trajectories are in black, and the icons show the orientation of the quadrotor at every $k=7$ time steps. The safe recovery region is highlighted in green. The blue-dashed line indicate the state constraint. Left: In 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.

an unsafe ground region, such as the busy road in the example in Fig. 1, specified as the region of states with $|x|<1,\ y\leq 0$. Therefore, we take the recovery region $\mathcal{X}_R$ as all landed states with $|x|\geq 1$. Clearly, $\mathcal{X}_R$ is a safe recovery region for $\pi_R(\boldsymbol{y})=0$, under the true dynamics (1).[1] For the drone to cross the road, we need to maintain recoverability with respect to either of the two disjoint recovery regions. We compare our approach with another naive baseline, which we label the Unsafe Recovery 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 region, slows down, and then switches to the leftmost recovery region once a feasible trajectory is found. In contrast, the Unsafe Recovery 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.

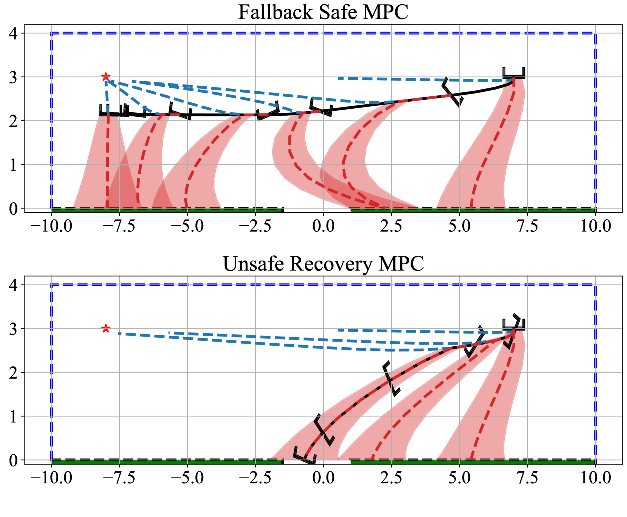

Fig. 4. Trajectories of the planar quadrotor in the xy-plane. The disjoint safe recovery regions are highlighted in green, the unsafe ground region is the unmarked region $|x|<1,\ y\leq 0$. Both the top and bottom figure follow the layout in Fig. 3 (left).

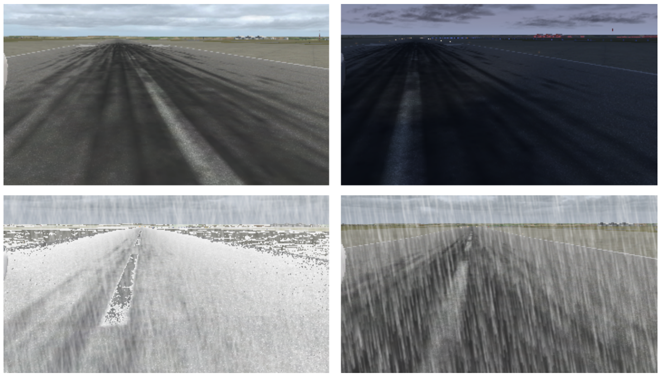

Fig. 5. Simulated environments in the X-Plane 11 simulator. Top left: Morning, no weather. Top right: Night, no weather. Bottom left: Afternoon, snowing. Bottom right: Afternoon, raining.

X-Plane Aircraft Simulator: Finally, we evaluate the conformal prediction Algorithm 2 and the end-to-end safety 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 in the deployment context 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[1] under $\pi_R(\boldsymbol{y}) :=$

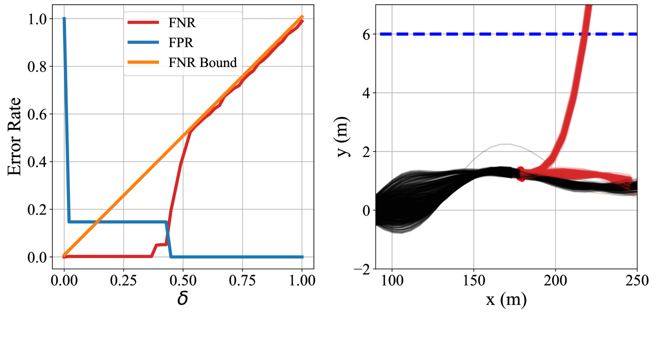

Fig. 6. Left: FNR Bound indicates the $\delta+1/(|\mathcal{A}|+1)$ bound on the FNR from Lemma 2. The FPR indicates the empirical rate at which we trigger the fallback without a perception fault ever occurring. The FNR indicates the empirical rate at which a perception fault occurs before we trigger 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.

[1] 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 $\hat{x}$ 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 (7) is infeasible, a simple fix first proposed in [40]. We did not observe recursive feasibility issues in the simulations.

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 (7) 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 using and 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 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 region, a subset of the state space that we can make invariant without full knowledge of the state. We have demonstrated that recovery regions may be readily identified, and 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 our monitors to 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," 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 from images via learned perception modules and contraction 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," 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," 2023, Available at https://stanfordasl.github.io/wp-content/papercite-data/pdf/Sinha.Pavone.CDC23.pdf.

[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.

[41] B. Recht, R. Roelofs, L. Schmidt *et al.*, “Do ImageNet classifiers generalize to ImageNet?” in *Proceedings of the 36th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, K. Chaudhuri and R. Salakhutdinov, Eds., 09–15 Jun 2019.

[42] Y. Ovadia, E. Fertig, J. Ren *et al.*, “Can you trust your model's uncertainty? evaluating predictive uncertainty under dataset shift,” in *Proceedings of the 33rd International Conference on Neural Information Processing Systems*, 2019.

[43] D. Hendrycks and T. Dietterich, “Benchmarking neural network robustness to common corruptions and perturbations,” *Proceedings of the International Conference on Learning Representations*, 2019.

[44] A. Sharma, N. Azizan, and M. Pavone, “Sketching curvature for efficient out-of-distribution detection for deep neural networks,” in *Proceedings of the Thirty-Seventh Conference on Uncertainty in Artificial Intelligence*, ser. Proceedings of Machine Learning Research, C. de Campos and M. H. Maathuis, Eds., 27–30 Jul 2021.

[45] P. W. Koh *et al.*, “Wilds: A benchmark of in-the-wild distribution shifts,” in *Proceedings of the 38th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, M. Meila and T. Zhang, Eds., 18–24 Jul 2021.

[46] J. Yang, K. Zhou, Y. Li *et al.*, “Generalized out-of-distribution detection: A survey,” *arXiv preprint arXiv:2110.11334*, 2021.

[47] B. Lakshminarayanan, A. Pritzel, and C. Blundell, “Simple and scalable predictive uncertainty estimation using deep ensembles,” in *Advances in Neural Information Processing Systems*, I. Guyon, U. V. Luxburg, S. Bengio *et al.*, Eds., 2017.

[48] Y. Gal and Z. Ghahramani, “Dropout as a bayesian approximation: Representing model uncertainty in deep learning,” in *Proceedings of The 33rd International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, M. F. Balcan and K. Q. Weinberger, Eds., 20–22 Jun 2016.

[49] M. Abdar, F. Pourpanah, S. Hussain *et al.*, “A review of uncertainty quantification in deep learning: Techniques, applications and challenges,” *Information Fusion*, 2021.

[50] A. Amini, W. Schwarting, A. Soleimany *et al.*, “Deep evidential regression,” in *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell *et al.*, Eds., 2020.

[51] I. Osband, Z. Wen, S. M. Asghari *et al.*, “Epistemic neural networks,” 2023.

[52] J. Z. Liu, Z. Lin, S. Padhy *et al.*, “Simple and principled uncertainty estimation with deterministic deep learning via distance awareness,” 2020.

[53] Z. Fang, Y. Li, J. Lu *et al.*, “Is out-of-distribution detection learnable?” in *Advances in Neural Information Processing Systems*, 2022.

[54] P. Antonante, D. I. Spivak, and L. Carlone, “Monitoring and diagnosability of perception systems,” in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2021.

[55] J. M. Manzano, D. Limon, D. Muñoz de la Peña *et al.*, “Robust learning-based mpc for nonlinear constrained systems,” *Automatica*, 2020.

[56] J. Schulman, S. Levine, P. Abbeel *et al.*, “Trust region policy optimization,” in *Proceedings of the 32nd International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, F. Bach and D. Blei, Eds., 07–09 Jul 2015.

[57] S. Levine, A. Kumar, G. Tucker *et al.*, “Offline reinforcement learning: Tutorial, review, and perspectives on open problems,” 2020.

[58] L. Hewing, J. Kabzan, and M. N. Zeilinger, “Cautious model predictive control using gaussian process regression,” *IEEE Transactions on Control Systems Technology*, 2020.

[59] S. Levine, C. Finn, T. Darrell *et al.*, “End-to-end training of deep visuomotor policies,” *Journal of Machine Learning Research*, 2016.

[60] B. Thananjeyan, A. Balakrishna, S. Nair *et al.*, “Recovery rl: Safe reinforcement learning with learned recovery zones,” *IEEE Robotics and Automation Letters*, 2021.

[61] S. M. Katz, A. L. Corso, C. A. Strong *et al.*, “Verification of image-based neural network controllers using generative models,” *Journal of Aerospace Information Systems*, 2022.

APPENDIX I
GLOSSARY

A glossary of all the notational conventions and symbols used in this paper is included below.

-	Symbol	Description
Conventions and Notation	x	unless explicitly defined otherwise, scalar variables are lowercase
	$\mathbf{x}$	vectors are boldfaced
	$\mathcal{X}$	sets are caligraphic
	x_t	time-varying quantities are indexed with a subscript $t\in\mathbb{N}_{\geq 0}$
	$\mathbf{R}$	matrices are uppercase and boldfaced
	$\text{Prob}(x\geq 0)$	probability of the event $x\geq 0$
	$\lambda,\delta,\epsilon,\theta$	hyperparameters (regardless of their type) are lowercase Greek characters
	$\boldsymbol{x}_{t+k\vert t}$	Predicted quantities at k time steps into the future computed at time step t. Read $x_{t+k\vert t}$ as "the predicted value of x at time $t+k$ given time t."
Variables		

APPENDIX II
EXTENDED RELATED WORKS

The fact that modern deep learning models behave poorly on OOD data has been extensively documented in recent years. For perception algorithms, this results in models failing in ways that are difficult to anticipate because it is impossible to derive or intuit what makes the pixel values of high-dimensional input image "different" to training data from first-principles [2], [3], [41]. For example, researchers have shown that vision models often confidently make incorrect classifications on images with unseen classes [42] or minor corruptions or perturbations [43] and that they greatly deteriorate in performance when the input domain subtly changes due to e.g., differences in lighting (such as day-night shifts), weather, and background scenery (e.g., across neighboring countries or even between rural and urban scenes) [2], [44], [45]. In the ML community, this has primarily been studied from a model-centric view by constructing benchmarks to study model degradation and by proposing algorithms that improve domain generalization and domain adaptation algorithms that adapt models in response to data from shifted distributions (see [1], and the references therein for an overview). However, few platforms exist to benchmark the degradation of a closed-loop control system that repeatedly applies a learned model in feedback. Moreover, it is a basic fact that we cannot anticipate or robustify against all OOD failure modes [1], [4]. Therefore, we implement several common OOD scenarios in the photorealistic X-Plane simulator to test our approach, and open-source our benchmark platform as a community resource.

In addition, a large number of OOD detection and runtime monitoring algorithms have been proposed to detect when a perception model is operating outside of its competence [5]. These methods commonly train an additional network to flag inputs that are dissimilar from the training data in a variety of ways, for example, by directly modeling the input distribution, measuring distances in latent-spaces, analyzing autoencoder reconstruction errors, or directly classifying whether inputs are OOD with one-class classification losses (see [6], [7], [46] for an overview). Other methods construct improved uncertainty scores produced by the perception model by approximating the Bayesian posterior through e.g., ensembling [47], Monte-Carlo dropout [48], choices of architecture or loss functions [49]–[52], or LaPlace approximations [44]. In general, these algorithms are heuristics that correlate with perception faults, but do not provide formal guarantees of correctness. In fact, recent work proved that the OOD detection problem is not PAC (provably approximately correct) learnable in a number of settings [53]. Moreover, even algorithms that certifiably detect perception faults in certain settings (e.g., by checking consistency in outputs between sensing modalities and across time e.g., see [5], [54]) first require us to define what constitutes a perception fault (i.e., "how bad is too bad"), because state estimates are never exactly correct. Therefore, we jointly design the runtime monitor and control stack by specifying an error bound on nominal perception errors in the control design, and then construct a runtime monitor that uses an OOD detector to predict when that bound is violated.

We take this approach because it is difficult to apply existing work on output-feedback to a perception-enabled system. In particular, a significant body of work constructs controllers that satisfy state and input constraints for all time despite partial observability using robust model predictive control (MPC) (e.g., see [20]–[25], [55]). A common approach is use knowledge of the dynamics and measurement model with assumptions on system observability to construct state estimators that persistently satisfy known error bounds. They then 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 black-box neural networks to extract scene information, which may exhibit arbitrarily poor error behavior on OOD inputs. Moreover, in our setting, the measurements that remain reliable after the perception fails are insufficient to recover the state (e.g., i.e., fallback system is not observable). 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.

In addition, our approach takes inspiration from existing work on *safety filters* that modify an ML-based black-box policy's actions to maintain safety [14]. Such methods minimally modify nominal decisions to ensure invariance of a safe region 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 [17], [18]. However, our method differs from such approaches in two important ways: First, existing safety filters typically operate under assumptions of perfect state knowledge, or that an estimate of sufficient accuracy is always available [19]. In contrast, we must ensure the fallback strategy satisfies full state constraints only using the last accurate estimate and any remaining partial state information. To account for these discrete information modes, our insight is that we can often specify *recovery regions* to fallback into, safe regions of the state space that we can make invariant without full knowledge of the state. For example, the drone in Fig. 1 does not need an obstacle detector to avoid 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 model outputs. We consider a setting where ML perception is critical to achieve the task, and therefore such an approach is much too conservative. Instead, we recognize that ML-enabled components are generally reliable, but leverage OOD detection to transition to a fallback strategy in rare failure modes.

Rather than filtering a black-box learned system's outputs to remain safe, another major line of research has been to construct controllers that ensure learned models operate within their domain of competence [19]. This includes work on policy optimization in (model-based, offline) reinforcement learning (RL) [56], [57], where model updates are typically constrained to trust regions to keep trajectories close to existing data, and learning-based control [19], [58], where MPC planning is typically combined with Bayesian model-learning to avoid states with high uncertainty in the dynamics. RL algorithms have been extensively applied to vision-based tasks [56], [59], [60]. In addition, some recent approaches in learning-based control propose to learn the error behavior of a vision system as a function of the state and then verify closed-loop properties [61] or robustly plan while taking these error bounds into account [26]–[28], for example, by making smoothness assumptions on the vision's error behavior [26]. However, these algorithms require the environment to remain fixed (i.e., that the mapping from state to image is constant), such that degradation in model quality is solely a result of visiting unseen states due to changes in the control policy. Instead, many of the subtle failure modes of perception systems are caused by environmental changes beyond the control of the robot, like weather changes. Therefore, the robot cannot act to retain confidence in the perception system, so we consider triggering a fallback the only reasonable alternative.

We take inspiration from 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. Most closely related to our approach are several applied works that trigger a fallback controller by thresholding an OOD detector in the robotics field [8]–[10]. These works use fallbacks that are domain-specific or assumed to be safe, assume the ML models function perfectly nominally, and rely on OOD detectors without end-to-end guarantees of system safety. We formalize the design of fallback strategy through the definition of recovery sets and make end-to-end guarantees.

To make an end-to-end guarantee, we leverage recent results in conformal prediction. This is because traditional methods for fault detection, such as model-based approaches, are difficult to apply to vision failures. Instead, we learn how to rely on a heuristic OOD detector through a conformal inference procedure (see [29] for an overview). Conformal methods are attractive in this setting because they produce strong guarantees on the correctness of predictions with very little data and those guarantees are distribution-free – that is, they do not depend on assumptions on the data-generating distribution [29], [30]. However, such methods have not generally been proposed for making high-probability guarantees jointly for all time in a sequential prediction setting, where inputs are correlated over time. While some recent work has aimed to move beyond the i.i.d. setting [31], [32], or making sequentially valid predictions across i.i.d. trajectories [33], these cannot be applied sequentially over correlated observations within a trajectory. Instead, we adapt an existing algorithm, [34], to the sequential prediction setting by noticing we only require a guarantee on the first detection of a fault to ensure the end-to-end safety of our framework.

APPENDIX III
PROOF OF LEMMA 1

Proof: First, we prove that $\hat{\boldsymbol{x}}_{t+k} \in \widehat{\mathcal{R}}_k$ for all $k \in \{0,...T+1\}$. We proceed by induction. Here, we drop arguments to $\mathcal{R}$ for notational simplicity (i.e., $\mathcal{R}_0 := \mathcal{R}_0(\hat{\boldsymbol{x}}_t, \boldsymbol{u}_{1:T})$). As the base case, note that $\hat{\boldsymbol{x}}_t \in \widehat{\mathcal{R}}_0$ by definition. For the inductive step, assume $\hat{\boldsymbol{x}}_{t+k} \in \widehat{\mathcal{R}}_k$ for some $k \in \{0,...,T\}$. Since 1) we assume $\boldsymbol{e}_{t:t+T+1} \subset \mathcal{E}_\theta$ and 2) the estimated state follows the dynamics in (6), that is, that $\hat{\boldsymbol{x}}_{t+k+1} = \boldsymbol{f}(\hat{\boldsymbol{x}}_{t+k} + \boldsymbol{e}_{t+k}, \boldsymbol{u}_k^F(\boldsymbol{g}(\hat{\boldsymbol{x}}_{t+k} + \boldsymbol{e}_{t+k})), \boldsymbol{w}_{t+k}) - \boldsymbol{e}_{t+k+1}$, it holds that $\hat{\boldsymbol{x}}_{t+k+1} \in \widehat{\mathcal{R}}_{k+1}$ by construction. Therefore, $\hat{\boldsymbol{x}}_{t+k} \in \widehat{\mathcal{R}}_{t+k}$ for all $k \in \{0,...,T+1\}$.

Next, we prove that $\widehat{\mathcal{R}}_k(\hat{\boldsymbol{x}}_{t+1}, \boldsymbol{u}_{2:T}^F) \subseteq \widehat{\mathcal{R}}_{k+1}(\hat{\boldsymbol{x}}_t, \boldsymbol{u}_{1:T}^F)$ for $k \in \{0,...,T\}$ by induction. As the base case, note that $\hat{\boldsymbol{x}}_{t+1} \in \hat{\mathcal{R}}_1(\hat{\boldsymbol{x}}_t, \boldsymbol{u}_{1:T}^F)$. Therefore, $\{\hat{\boldsymbol{x}}_{t+1}\} = \widehat{\mathcal{R}}_0(\hat{\boldsymbol{x}}_{t+1}, \boldsymbol{u}_{2:T}^F) \subseteq \widehat{\mathcal{R}}_1(\hat{\boldsymbol{x}}_t, \boldsymbol{u}_{1:T}^F)$. For the inductive step, assume that $\widehat{\mathcal{R}}_k(\hat{\boldsymbol{x}}_{t+1}, \boldsymbol{u}_{2:T}^F) \subseteq \widehat{\mathcal{R}}_{k+1}(\hat{\boldsymbol{x}}_t, \boldsymbol{u}_{1:T}^F)$ for some $k \in \{0,...,T-1\}$. Suppose $\hat{\boldsymbol{x}}' \in \widehat{\mathcal{R}}_{k+1}(\hat{\boldsymbol{x}}_{t+1}, \boldsymbol{u}_{2:T}^F)$. Then, by definition, $\hat{\boldsymbol{x}}' = \boldsymbol{f}(\hat{\boldsymbol{x}} + \boldsymbol{e}, \boldsymbol{u}_{k+1}^F(\boldsymbol{g}(\hat{\boldsymbol{x}} + \boldsymbol{e})), \boldsymbol{w}) - \boldsymbol{e}'$ for some $\hat{\boldsymbol{x}} \in \widehat{\mathcal{R}}_k(\hat{\boldsymbol{x}}_{t+1}, \boldsymbol{u}_{2:T}^F)$, $\boldsymbol{w} \in \mathcal{W}$, and $\boldsymbol{e}$, $\boldsymbol{e}' \in \mathcal{E}_\theta$. This immediately implies that $\hat{\boldsymbol{x}}' \in \widehat{\mathcal{R}}_{k+2}(\hat{\boldsymbol{x}}_t, \boldsymbol{u}_{1:T}^F)$, since we assume $\widehat{\mathcal{R}}_k(\hat{\boldsymbol{x}}_{t+1}, \boldsymbol{u}_{2:T}^F) \subset \widehat{\mathcal{R}}_{k+1}(\hat{\boldsymbol{x}}_t, \boldsymbol{u}_{1:T}^F)$. Therefore, $\widehat{\mathcal{R}}_{k+1}(\hat{\boldsymbol{x}}_{t+1}, \boldsymbol{u}_{2:T}^F) \subseteq \widehat{\mathcal{R}}_{k+2}(\hat{\boldsymbol{x}}_t, \boldsymbol{u}_{1:T}^F)$. Therefore,

$$\widehat{\mathcal{R}}_k(\hat{\boldsymbol{x}}_{t+1}, \boldsymbol{u}_{2:T}^F) \subseteq \widehat{\mathcal{R}}_{k+1}(\hat{\boldsymbol{x}}_t, \boldsymbol{u}_{1:T}^F), \quad \forall k \in \{0,...,T\} \tag{10}$$

by induction.

Finally, (10) immediately implies that $\mathcal{R}_k(\hat{\boldsymbol{x}}_{t+1}, \boldsymbol{u}_{2:T}^F) \subseteq \mathcal{R}_{k+1}(\hat{\boldsymbol{x}}_t, \boldsymbol{u}_{1:T}^F)$ for all $k \in \{0,...,T\}$. Moreover, since $0 \in \mathcal{E}_\theta$, we have that $\widehat{\mathcal{R}}_k(\hat{\boldsymbol{x}}_t, \boldsymbol{u}_{1:T}^F) \subseteq \mathcal{R}_k(\hat{\boldsymbol{x}}_t, \boldsymbol{u}_{1:T}^F)$ for $k \in \{0,...,T+1\}$. In addition, since 1) we proved $\hat{x}_{t+k} \in \widehat{\mathcal{R}}_k(\hat{\boldsymbol{x}}_t, \boldsymbol{u}_{1:T}^F)$ and 2) we assume $\boldsymbol{e}_{t:t+T+1} \subset \mathcal{E}_\theta$, we have that $\boldsymbol{x}_{t+k} \in \mathcal{R}_k(\hat{\boldsymbol{x}}_t, \boldsymbol{u}_{1:T}^F)$ for all $k \in \{0,...,T+1\}$.

APPENDIX IV
PROOF OF THEOREM 1

As a shorthand in the theorem statement and the following proofs, we use the notation t_{fail} to denote the first time step that the runtime monitor raises an alarm, i.e., that $w(a_t)=0$ for all $t<t_{\text{fail}}$ and that $w(a_{t_{\text{fail}}})=1$ if $t_{\text{fail}}<\infty$. To prove Theorem 1, we first prove the recursive feasibility of the MPC (7) in Algorithm 2 up to the failure time t_{fail}.

Lemma 3. *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 the runtime monitor w 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 $t<t_{\text{fail}}$. Then, if the Fallback-Safe MPC problem (7) is feasible at $t=0$ and $w(a_0)=0$, we have that 1) the MPC problem (7) 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 $t<t_{\text{fail}}$.*

Proof: Since we assume $w(a_0)=0$, we have that $t_{\text{fail}}\geq 1$. Therefore, suppose the MPC (7) is feasible at some time $t\in\{0,...,t_{\text{fail}}-1\}$ with optimal fallback policy sequence $\boldsymbol{u}^{F,\star}_{t:t+T|t}$. Then, $\boldsymbol{x}_t \in \mathcal{X}$, since $\boldsymbol{x}_t \in \mathcal{R}_0(\hat{\boldsymbol{x}}, \boldsymbol{u}^{F,\star}_{t:t+T|t}) \subseteq \mathcal{X}$ by Lemma 1 and the assumption that $\boldsymbol{e}_{0:t_{\text{fail}}-1} \subseteq \mathcal{E}_\theta$. In addition, we have that Algorithm 1 applies the input $\pi(\hat{\boldsymbol{x}}_t, \boldsymbol{y}_t) = \boldsymbol{u}^\star_{t|t}(\hat{\boldsymbol{x}}_t) \in \mathcal{U}$, since we enforce $\boldsymbol{u}^\star_{t|t}(\hat{\boldsymbol{x}}_t) = \boldsymbol{u}^{F,\star}_{t|t}(\boldsymbol{y}_t)$ and $\boldsymbol{u}^{F,\star}_{t|t}(\boldsymbol{y}_t) \in \mathcal{U}$ by construction in (7).

Next, consider the candidate fallback policy sequence $\boldsymbol{u}^F_{t+1:t+T+1|t} := \{\boldsymbol{u}^{F,\star}_{t+1|t}, ..., \boldsymbol{u}^{F,\star}_{t+T|t}, \pi_R\}$ for problem (7) at time $t+1$. If $t+1<t_{\text{fail}}$, Lemma 1 gives us that $\mathcal{R}_k(\hat{\boldsymbol{x}}_{t+1}, \boldsymbol{u}^F_{t+1:t+T|t}) \subseteq \mathcal{R}_{k+1}(\hat{\boldsymbol{x}}_t, \boldsymbol{u}^{F,\star}_{t:t+T|t})$ for $k\in\{0,...,T\}$, since we assume $\boldsymbol{e}_{0:t_{\text{fail}-1}} \subset \mathcal{E}_\theta$. Moreover, since $\mathcal{R}_{T+1}(\hat{\boldsymbol{x}}_t, \boldsymbol{u}^{F,\star}_{t:t+T}) \subseteq \mathcal{X}_R$, we have that $\mathcal{R}_{T+2}(\hat{\boldsymbol{x}}_{t+1}, \boldsymbol{u}^F_{t+1:t+T+1|t}) \subseteq \mathcal{X}_R$ by Assumption 4 and Lemma 1. Therefore, the candidate fallback policy sequence $\boldsymbol{u}^F_{t+1:t+T+1|t}$ satisfies the state and input constraints in (7) at $t+1$. Since we assume there always exists a $\boldsymbol{u}\in\mathcal{P}_N$ such that $\boldsymbol{u}(\hat{\boldsymbol{x}}_{t+1}) = \boldsymbol{u}^{F,\star}_{t+1|t}(\boldsymbol{y}_{t+1})$, problem (7) is therefore feasible at time $t+1$. The lemma then holds by induction, since we assume (7) is feasible at $t=0$ and that $w(a_0)=0$.

The proof of Theorem 2 then follows by combining Lemma 3 with the logic that triggers the fallback in Algorithm 1.

Proof of Theorem 1:

Proof: By Lemma 3, feasibility of the MPC (7) at $t=0$ and $w(a_0)=0$ implies that the MPC is feasible for all $0\leq t\leq t_{\text{fail}}-1$ and that the system satisfies state and input constraints for $t<t_{\text{fail}}$. Moreover, since a) (7) is feasible at time step $t_{\text{fail}}-1$, b) $\boldsymbol{u}^\star_{t_{\text{fail}}-1|t_{\text{fail}}-1}(\hat{\boldsymbol{x}}_{t_{\text{fail}}-1}) = \boldsymbol{u}^{F,\star}_{t_{\text{fail}}-1|t_{\text{fail}}-1}(\boldsymbol{y}_{t_{\text{fail}}-1})$ by construction, and c) Algorithm 1 applies the fallback strategy $\boldsymbol{u}^{F,\star}_{t|t_{\text{fail}}-1}$ for $t\in\{t_{\text{fail}},...,t_{\text{fail}}+T-1\}$, it holds that $\boldsymbol{x}_t \in \mathcal{X}$ and $\boldsymbol{u}_t \in \mathcal{U}$ for all $t\in\{t_{\text{fail}},...,t_{\text{fail}}+T-1\}$ by Lemma 1. Moreover, feasibility of the MPC (7) at $t_{\text{fail}}-1$ also implies that $\boldsymbol{x}_{t_{\text{fail}}+T} \in \mathcal{X}_R$. Furthermore, Assumption 4 and $0\in\mathcal{E}_\theta$ give us that the application of π_R for all $t\geq t_{\text{fail}}+T$ ensures that $\boldsymbol{x}_t \in \mathcal{X}_R \subseteq \mathcal{X}$ and $\boldsymbol{u}_t \in \mathcal{U}$ for all $t\geq t_{\text{fail}}+T$.

APPENDIX V
TRACTABLE REFORMULATION OF (7) FOR LINEAR-QUADRATIC SYSTEMS

We consider linear systems of the form

$$\begin{aligned} \boldsymbol{x}_{t+1} &= A\boldsymbol{x}_t + B\boldsymbol{u}_t + \boldsymbol{w}_t \\ \boldsymbol{y}_t &= C\boldsymbol{x}_t \end{aligned}, \tag{11}$$

subject to polytopic constraints on states and inputs,

$$\mathcal{X} = \{\boldsymbol{x} \,:\, H_x\boldsymbol{x} \leq h_x\}, \quad \mathcal{U} = \{\boldsymbol{u} \,:\, H_u\boldsymbol{u} \leq \boldsymbol{h}_u\},$$

and a polytopic disturbance and perception error bound of the form

$$\mathcal{W} = \{\boldsymbol{w} \,:\, H_w\boldsymbol{w} \leq h_w\}, \quad \mathcal{E}_\theta = \{\boldsymbol{e} \,:\, H_e\boldsymbol{e} \leq h_e\}.$$

Let $\bar{\boldsymbol{x}}_t = \hat{\boldsymbol{x}}_t$ be the nominal state of the fallback trajectory at time t and define the nominal fallback dynamics as

$$\begin{aligned} \bar{\boldsymbol{x}}_{t+1} &= A\bar{\boldsymbol{x}}_t + B\bar{\boldsymbol{u}}^F_t \\ \bar{\boldsymbol{y}}_t &= C\bar{\boldsymbol{x}}_t \end{aligned}$$

for $t\in\{t, t+T+1\}$ under the open-loop input sequence $\bar{\boldsymbol{u}}^F_{t:t+T} \subset \mathbb{R}^m$. As is normative in robust MPC, we consider affine fallback policy sequences that satisfy

$$\boldsymbol{u}^F_{t+k}(\boldsymbol{y}_t) = \bar{\boldsymbol{u}}^F_{t+k} + K(\boldsymbol{y}_{t+k} - \bar{\boldsymbol{y}}_{t+k}) \tag{12}$$

for $k\in\{0,...,T\}$, where $K\in\mathbb{R}^{m\times r}$ is a fixed feedback gain specified by a designer. Under this fallback policy sequence, it holds that

$$\begin{aligned} \hat{\boldsymbol{x}}_{t+1} - \bar{\boldsymbol{x}}_{t+1} &= (A\boldsymbol{x}_t + B\boldsymbol{u}^F_t(\boldsymbol{y}_t) + \boldsymbol{w}_t - \boldsymbol{e}_{t+1}) - (A\bar{\boldsymbol{x}}_t + B\bar{\boldsymbol{u}}^F_t) \\ &= A(\hat{\boldsymbol{x}}_t + \boldsymbol{e}_t) + B(\bar{\boldsymbol{u}}^F_t + K(\boldsymbol{y}_t - \bar{\boldsymbol{y}}_t)) + \boldsymbol{w}_t - \boldsymbol{e}_{t+1} - (A\bar{\boldsymbol{x}}_t + B\bar{\boldsymbol{u}}^F_t) \\ &= A(\hat{\boldsymbol{x}}_t + \boldsymbol{e}_t) + BKC(\hat{\boldsymbol{x}}_t + \boldsymbol{e}_t - \bar{\boldsymbol{x}}_t) + \boldsymbol{w}_t - \boldsymbol{e}_{t+1} - A\bar{\boldsymbol{x}}_t \\ &= (A+BKC)(\hat{\boldsymbol{x}}_t - \bar{\boldsymbol{x}}_t) + (A+BKC)\boldsymbol{e}_t + \boldsymbol{w}_t - \boldsymbol{e}_{t+1}. \end{aligned}$$

Therefore, by recursively defining the sets

$$\begin{aligned}\mathcal{F}_0 &:= \{0\}\\ \mathcal{F}_{k+1} &:= (A+BKC)\mathcal{F}_k \oplus (A+BKC)\mathcal{E}_\theta \oplus \mathcal{W} \oplus \mathcal{E}_\theta\end{aligned}$$

for $k \in \{0,...,T\}$ and noting that $\mathcal{E}_\theta$ is symmetric, we then have that the $k-$step reachable sets under $\boldsymbol{u}^F_{t:t+T}$ are given as

$$\begin{aligned}\widehat{\mathcal{R}}_k(\hat{\boldsymbol{x}}_t, \boldsymbol{u}^F_{t:t+T}) &= \{\bar{\boldsymbol{x}}_{t+k}\} \oplus \mathcal{F}_k\\ \mathcal{R}_k(\hat{\boldsymbol{x}}_t, \boldsymbol{u}^F_{t:t+T}) &= \{\bar{\boldsymbol{x}}_{t+k}\} \oplus \mathcal{F}_k \oplus \mathcal{E}_\theta\end{aligned}$$

for $k \in \{0,...,T+1\}$. Therefore, for a linear system, we reformulate (7) as

$$\begin{aligned}\underset{\substack{\bar{\boldsymbol{u}}^F_{t:t+T|t} \subset \mathbb{R}^m,\\ \boldsymbol{u}_{t:t+T|t} \subset \mathcal{U}}}{\text{minimize}} \quad & C(\hat{\boldsymbol{x}}_t, \boldsymbol{u}_{t:t+T|t}, \bar{\boldsymbol{u}}^F_{t:t+T|t})\\ \text{subject to} \quad & \bar{\boldsymbol{x}}_{t+k+1|t} = A\bar{\boldsymbol{x}}_{t+k|t} + B\bar{\boldsymbol{u}}^F_{t+k|t} \qquad \forall k \in \{0,...,T\},\\ & \bar{\boldsymbol{x}}_{t+k|t} \in \mathcal{X} \ominus (\mathcal{F}_k \oplus \mathcal{E}_\theta) \qquad \forall k \in \{0,...,T\},\\ & \bar{\boldsymbol{u}}^F_{t+k|t} \in \mathcal{U} \ominus KC(\mathcal{F}_k \oplus \mathcal{E}_\theta) \qquad \forall k \in \{0,...,T\},\\ & \bar{\boldsymbol{x}}_{t+T+1|t} \in \mathcal{X}_R \ominus (\mathcal{F}_{T+1} \oplus \mathcal{E}_\theta),\\ & \boldsymbol{u}_{t|t} = \bar{\boldsymbol{u}}^F_{t|t}, \quad \bar{\boldsymbol{x}}_{t|t} = \hat{\boldsymbol{x}}_t.\end{aligned} \tag{13}$$

In (13), we specify an objective of the form

$$C(\hat{\boldsymbol{x}}_t, \boldsymbol{u}_{t:t+T|t}, \bar{\boldsymbol{u}}^F_{t:t+T|t}) := \sum_{k=0}^{T} h_N(\bar{\boldsymbol{x}}^N_{t+k|t}, \boldsymbol{u}_{t+k|t}) + V_N(\bar{\boldsymbol{x}}^N_{t+T+1|t}),$$

where we define the nominal state evolution as following $\bar{\boldsymbol{x}}^N_{t|t} := \hat{\boldsymbol{x}}_t$ and $\bar{\boldsymbol{x}}^N_{t+k+1} = A\bar{\boldsymbol{x}}^N_{t+k} + B\boldsymbol{u}_{t+k|t}$ for a positve (semi) definite quadratic stage cost h and terminal cost V. Moreover, since we assume $\mathcal{X}$, $\mathcal{U}$, $\mathcal{W}$, and $\mathcal{E}_\theta$ are polytopes, we can compute the constraint tightening in (13) (that is, $\mathcal{X} \ominus (\mathcal{F}_k \oplus \mathcal{E}_\theta)$, $\mathcal{U} \ominus KC(\mathcal{F}_k \oplus \mathcal{E}_\theta)$, $\mathcal{X}_R \ominus (\mathcal{F}_{T+1} \oplus \mathcal{E}_\theta)$) via linear programming, rendering (13) a convex quadratic program (QP) [25]. In addition, we use [36, Alg. 10.5] to compute the robust invariant set $\mathcal{X}_R$ given a recovery policy.

APPENDIX VI
APPROXIMATE SOLUTION APPROACH TO (7) FOR NONLINEAR SYSTEMS WITH PARTICLE MPC [37]

For nonlinear systems, we propose the use of the Particle MPC (PMPC) algorithm [37] to approximate a solution to (7). In lieu of explicitly computing the $k-$step reachable sets, the PMPC algorithm accounts for uncertainties by sampling $M \in \mathbb{N}_{>0}$ trajectories from an initial belief state and a given control input sequence, and enforcing the state and input constraints along the M sampled trajectories. Therefore, when we apply the PMPC algorithm, we consider both open-loop nominal input sequences $\boldsymbol{u}_{t:t+T|t} \subset \mathbb{R}^m$ and open-loop fallback input sequences $\boldsymbol{u}^F_{t:t+T|t} \subset \mathbb{R}^m$ for simplicity. First, we define a shorthand dynamics term $\boldsymbol{h}$ to approximate the dynamics of the perception estimate in (6) as

$$\boldsymbol{h}(\bar{\boldsymbol{x}}, \boldsymbol{u}, \boldsymbol{w}, \boldsymbol{e}, \boldsymbol{e}') := \boldsymbol{f}(\bar{\boldsymbol{x}} + \boldsymbol{e}, \boldsymbol{u}, \boldsymbol{w}) - \boldsymbol{e}'.$$

Then, at time t, we sample M disturbance sequences as $\{\boldsymbol{w}^j_{t+k|t}, \boldsymbol{e}^j_{t+k|t}, \boldsymbol{e}'^j_{t+k|t}\}_{j=0}^M$ from $\mathcal{W}$ and $\mathcal{E}_\theta$ so that we can approximate the reachable sets $\widehat{\mathcal{R}}_k$. We then approximate (7) as

$$\begin{aligned}\underset{\substack{\boldsymbol{u}^F_{t:t+T|t} \subset \mathcal{U},\\ \boldsymbol{u}_{t:t+T|t} \subset \mathcal{U}}}{\text{minimize}} \quad & \frac{1}{M}\sum_{j=0}^{M} C_j(\hat{\boldsymbol{x}}_t, \boldsymbol{u}_{t:t+T|t}, \boldsymbol{u}^F_{t:t+T|t})\\ \text{subject to} \quad & \bar{\boldsymbol{x}}^j_{t+k+1|t} = \boldsymbol{h}(\bar{\boldsymbol{x}}^j_{t+k|t}, \boldsymbol{u}^F_{t+k|t}, \boldsymbol{w}_{t+k|t}, \boldsymbol{e}^j_{t+k|t}, \boldsymbol{e}'^j_{t+k|t}) \quad \forall k \in \{0,...,T\},\ j \in \{0,...,M\},\\ & \bar{\boldsymbol{x}}^j_{t+k|t} \in \mathcal{X} \ominus \mathcal{E}_\theta, \quad \bar{\boldsymbol{x}}^j_{t|t} = \hat{\boldsymbol{x}}_t \quad \forall k \in \{0,...,T\},\ j \in \{0,...,M\},\\ & \bar{\boldsymbol{x}}^j_{t+T+1|t} \in \mathcal{X}_R \ominus \mathcal{E}_\theta, \quad \forall j \in \{0,...,M\},\\ & \boldsymbol{u}_{t|t} = \boldsymbol{u}^F_{t|t},\end{aligned} \tag{14}$$

on which the PMPC algorithm applies sequential convex programming (SCP) to yield a locally optimal solution. In (14), we have tightened the state constraints further to account for the perception error, i.e., the difference between $\mathcal{R}$ and $\widehat{\mathcal{R}}$. Moreover, in (14), we optimize the sample average of cost functions C_j. We draw the cost functions C_j as a sum of stage costs h over a sampled trajectory under the nominal inputs $\boldsymbol{u}_{t:t+T|t}$, i.e., as

$$C_j(\hat{\boldsymbol{x}}_t, \boldsymbol{u}_{t:t+T|t}, \boldsymbol{u}^F_{t:t+T|t}) = \sum_{k=0}^{T} h(\bar{\boldsymbol{x}}^{N,j}_{t+k|t}, \boldsymbol{u}_{t+k|t}) + V(\bar{\boldsymbol{x}}^{N,j}_{t+T+1|t}), \tag{15}$$

where $\bar{\boldsymbol{x}}^{N,j}_{t|t} := \hat{\boldsymbol{x}}_t$ and $\bar{\boldsymbol{x}}^{N,j}_{t+k+t} = \boldsymbol{h}(\bar{\boldsymbol{x}}^j_{t+k|t}, \boldsymbol{u}^F_{t+k|t}, \boldsymbol{w}_{t+k|t}, \boldsymbol{e}^j_{t+k|t}, \boldsymbol{e}^{'j}_{t+k|t})$ for $k \in \{0,...,T\}$. We emphasize that the disturbance sequences drawn to compute the cost functions are independent from those we use to evaluate the constraints in (14), i.e., we sample a total of $2M$ disturbance sequences at each time step, but keep notation minimal in (15).

APPENDIX VII
PROOF OF LEMMA 2

A. Proof of Lemma 2

Proof: First, we note that the false negative rate is bounded by the probability of a missed detection at t_{stop}, conditioned on the event that Algorithm 2 does not raise an alarm at any time $t < t_{\text{stop}}(\tau)$, i.e., that is, that $w(a_t) = 0$ for all $t \in \{0,...,t_{\text{stop}}(\tau) - 1\}$. As in the statement and proof of Theorem 1, we use the shorthand $t_{\text{fail}} \geq t_{\text{stop}}$ to denote this event, so that

$$\begin{aligned}
\text{Prob}\big(\text{False Negative}\big) &:= \text{Prob}\big(w(a_t) = 0 \ \forall t \in [0, t_{\text{stop}}(\tau)] \mid \boldsymbol{e}_{t_{\text{stop}}(\tau)} \notin \mathcal{E}_\theta\big) \\
&= \text{Prob}\big(w(a_{t_{\text{stop}}(\tau)}) = 0 \mid \boldsymbol{e}_{t_{\text{stop}}(\tau)} \notin \mathcal{E}_\theta,\ t_{\text{fail}} \geq t_{\text{stop}}\big)\text{Prob}\big(t_{\text{fail}} \geq t_{\text{stop}} \mid \boldsymbol{e}_{t_{\text{stop}}(\tau)} \notin \mathcal{E}_\theta\big) \\
&\leq \text{Prob}\big(w(a_{t_{\text{stop}}(\tau)}) = 0 \mid \boldsymbol{e}_{t_{\text{stop}}(\tau)} \notin \mathcal{E}_\theta,\ t_{\text{fail}} \geq t_{\text{stop}}\big).
\end{aligned} \tag{16}$$

Next, suppose the deployed trajectory τ conditioned on $t_{\text{fail}} \geq t_{\text{stop}}$ follows the training trajectory P so that $(\tau \mid t_{\text{fail}} \geq t_{\text{stop}}),\ \tau_1, \ldots, \tau_N \overset{\text{iid}}{\sim} P$, and let Prob_{iid} denote the probability of an event under this assumption. That is, $\text{Prob}_{\text{iid}}(\text{False Negative})$ denotes the probability of a false negative when $(\tau \mid t_{\text{fail}} \geq t_{\text{stop}}),\ \tau_1,...,\tau_N \overset{\text{iid}}{\sim} P$.

Under the i.i.d. assumption, it follows that $(\boldsymbol{x}_{t_{\text{stop}}(\tau)}, \hat{\boldsymbol{x}}_{t_{\text{stop}}(\tau)}, a_{t_{\text{stop}}(\tau)})$ and the elements of $\mathcal{D}_{\text{stop}}$ are also i.i.d. Therefore, the sequence $(\boldsymbol{e}_{t_{\text{stop}}(\tau)}, a_{t_{\text{stop}}(\tau)})$, $(\boldsymbol{e}^1_{t_{\text{stop}}(\tau_1)}, a^1_{t_{\text{stop}}(\tau_1)})$, ..., $(\boldsymbol{e}^N_{t_{\text{stop}}(\tau_N)}, a^N_{t_{\text{stop}}(\tau_N)})$ is exchangeable. By [34, Proposition 1], it then follows that

$$\text{Prob}_{\text{iid}}\big(w(a_{t_{\text{stop}}(\tau)}) = 0 \mid \boldsymbol{e}_{t_{\text{stop}}(\tau)} \notin \mathcal{E}_\theta,\ t_{\text{fail}} \geq t_{\text{stop}}\big) \leq \delta + \frac{1}{|\mathcal{A}| + 1}, \tag{17}$$

because Algorithm 2 implements [34, Algorithm 1] using $\mathcal{D}_{\text{stop}}$ as the calibration dataset, $-a_{t_{\text{stop}}}$ as the *surrogate safety score*, and *true safety score* $f(\boldsymbol{e}) := \mathbf{1}\{\boldsymbol{e} \in \mathcal{E}_\theta\}$.

Finally, we bound the difference between the false negative rate under the assumption that $(\tau \mid t_{\text{fail}} \geq t_{\text{stop}}),\ \tau_1,...,\tau_N \overset{\text{iid}}{\sim} P$ and the true false negative rate as

$$\begin{aligned}
&\text{Prob}\big(w(a_{t_{\text{stop}}(\tau)}) = 0 \mid \boldsymbol{e}_{t_{\text{stop}}(\tau)} \notin \mathcal{E}_\theta,\ t_{\text{fail}} \geq t_{\text{stop}}\big) - \text{Prob}_{\text{iid}}\big(w(a_{t_{\text{stop}}(\tau)}) = 0 \mid \boldsymbol{e}_{t_{\text{stop}}(\tau)} \notin \mathcal{E}_\theta,\ t_{\text{fail}} \geq t_{\text{stop}}\big) \\
&\leq \Big|\text{Prob}\big(w(a_{t_{\text{stop}}(\tau)}) = 0 \mid \boldsymbol{e}_{t_{\text{stop}}(\tau)} \notin \mathcal{E}_\theta,\ t_{\text{fail}} \geq t_{\text{stop}}\big) - \text{Prob}_{\text{iid}}\big(w(a_{t_{\text{stop}}(\tau)}) = 0 \mid \boldsymbol{e}_{t_{\text{stop}}(\tau)} \notin \mathcal{E}_\theta,\ t_{\text{fail}} \geq t_{\text{stop}}\big)\Big| \\
&\leq \text{TV}(P_{t_{\text{stop}}|\text{fault}}, P'_{t_{\text{stop}}|\text{fault}}).
\end{aligned} \tag{18}$$

Combining (16) with (17) and (18) then yields

$$\begin{aligned}
\text{Prob}\big(\text{False Negative}\big) &\leq \text{Prob}\big(w(a_{t_{\text{stop}}(\tau)}) = 0 \mid \boldsymbol{e}_{t_{\text{stop}}(\tau)} \notin \mathcal{E}_\theta,\ t_{\text{fail}} \geq t_{\text{stop}}\big) \\
&= \text{Prob}_{\text{iid}}\big(w(a_{t_{\text{stop}}(\tau)}) = 0 \mid \boldsymbol{e}_{t_{\text{stop}}(\tau)} \notin \mathcal{E}_\theta,\ t_{\text{fail}} \geq t_{\text{stop}}\big) \\
&\quad + \text{Prob}\big(w(a_{t_{\text{stop}}(\tau)}) = 0 \mid \boldsymbol{e}_{t_{\text{stop}}(\tau)} \notin \mathcal{E}_\theta,\ t_{\text{fail}} \geq t_{\text{stop}}\big) - \text{Prob}_{\text{iid}}\big(w(a_{t_{\text{stop}}(\tau)}) = 0 \mid \boldsymbol{e}_{t_{\text{stop}}(\tau)} \notin \mathcal{E}_\theta,\ t_{\text{fail}} \geq t_{\text{stop}}\big) \\
&\leq \delta + \frac{1}{|\mathcal{A}| + 1} + \text{TV}(P_{t_{\text{stop}}|\text{fault}}, P'_{t_{\text{stop}}|\text{fault}}).
\end{aligned}$$

APPENDIX VIII
PROOF OF THEOREM 2

Proof: First, we note that Lemma 2 bounds the false negative rate of a runtime monitor constructed using Algorithm 2. Therefore, we use Lemma 2 to lower-bound the probability that the runtime monitor does not miss a detection of a perception fault as

$$\begin{aligned}
\text{Prob}(\boldsymbol{e}_t \in \mathcal{E}_\theta \ \forall t < t_{\text{fail}}) &= 1 - \text{Prob}(\exists t < t_{\text{fail}} \text{ s.t. } \boldsymbol{e}_t \notin \mathcal{E}_\theta) \\
&= 1 - \text{Prob}(w(a_t) = 0 \ \forall t \in [0, t_{\text{stop}}] \cap \boldsymbol{e}_{t_{\text{stop}}} \notin \mathcal{E}_\theta) \\
&= 1 - \text{Prob}(w(a_t) = 0 \ \forall t \in [0, t_{\text{stop}}] \mid \boldsymbol{e}_{t_{\text{stop}}} \notin \mathcal{E}_\theta)\text{Prob}(\boldsymbol{e}_{t_{\text{stop}}} \notin \mathcal{E}_\theta) \\
&\geq 1 - \text{Prob}(w(a_t) = 0 \ \forall t \in [0, t_{\text{stop}}] \mid \boldsymbol{e}_{t_{\text{stop}}} \notin \mathcal{E}_\theta) \\
&\geq 1 - \delta - \frac{1}{|\mathcal{A}| + 1} - \text{TV}(P_{t_{\text{stop}}|\text{fault}}, P'_{t_{\text{stop}}|\text{fault}}).
\end{aligned}$$

Moreover, Theorem 1 gives us that

$$\text{Prob}(\boldsymbol{x}_t \in \mathcal{X},\ \boldsymbol{u}_t \in \mathcal{U} \ \forall t \in [0, t_{\text{lim}}] \mid \boldsymbol{e}_t \in \mathcal{E}_\theta \ \forall t < t_{\text{fail}}) = 1.$$

As a shorthand, let SAFE denote the event that $\boldsymbol{x}_t \in \mathcal{X},\ \boldsymbol{u}_t \in \mathcal{U}\ \forall t \in [0,t_{\mathrm{lim}}]$. By applying the law of total probability, it then follows that

$$\begin{aligned}\mathrm{Prob}(\mathrm{SAFE}) &\geq \mathrm{Prob}(\mathrm{SAFE} \mid \boldsymbol{e}_t \in \mathcal{E}_\theta\ \forall t < t_{\mathrm{fail}})\mathrm{Prob}(\boldsymbol{e}_t \in \mathcal{E}_\theta\ \forall t < t_{\mathrm{fail}}) \\ &= \mathrm{Prob}(\boldsymbol{e}_t \in \mathcal{E}_\theta\ \forall t < t_{\mathrm{fail}}) \\ &\geq 1-\delta-\frac{1}{|\mathcal{A}|+1}-\mathrm{TV}(P_{t_{\mathrm{stop}}|\mathrm{fault}}, P'_{t_{\mathrm{stop}}|\mathrm{fault}}).\end{aligned}$$

Appendix IX
Experiment Details

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 $\mathrm{dt} = 0.15s$. The drone is subject to wind disturbances in the set $\mathcal{W} = \{\boldsymbol{w} : (-0.05, -0.02, -1e-3, -1e-3, -1e-3, -1e-3) \leq \boldsymbol{w} \leq (0.05, 0.02, 1e-3, 1e-3, 1e-3, 1e-3)\}$, 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 simulate the nominal perception errors as bounded within the set $\mathcal{E} = \{\boldsymbol{e} = \boldsymbol{x} - \hat{\boldsymbol{x}}\ :\ \hat{\dot{\boldsymbol{p}}} = \boldsymbol{p},\ \hat{\theta} = \theta,\ \|[\hat{x},\hat{y}] - [x,y]\|_\infty \leq 0.05\}$, so that the perception system estimates the $xy-$position within a $10\mathrm{cm}$-wide box around the true position nominally, and perfectly outputs $\boldsymbol{y}$. When the vision system fails, we randomly sample the $xy-$position within the range $(-10,10)$. We set the time limit $t_{\mathrm{lim}} = 8\mathrm{s}$, and the controller horizon to $T = 10$. In these simulations we give the Fallback-Safe MPC a perfect runtime monitor, so that $w(\cdot) = 1 - \mathbb{1}\{\boldsymbol{e}_t \in \mathcal{E}\}$. We implement the Fallback-Safe MPC using the tube MPC formulation in Appendix V, where we place a quadratic distance penalty $P = I_{6\times 6}$ to a goal location on the nominal trajectory and a quadratic cost $R = I_{2\times 2}$ on each input.

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: $\mathcal{X} := \{\boldsymbol{x}\ :\ y \geq 0\}$. The fallback is for the drone to abort the landing and fly away at a sufficient altitude. So, we take the recovery set as $\mathcal{X}_R = \{\boldsymbol{x}\ :\ \boldsymbol{y} \geq 2\}$. We note that under the recovery policy $\pi_R(\boldsymbol{y}) := \epsilon + K\boldsymbol{y}$, where K stabilizes the orientation θ around 0 and ϵ is a small offset to continually fly upward, the recovery set is invariant under both the estimator dynamics (6) 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 does not anticipate vision failures. 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 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 the in-air xy goal location $x_g = [-8,3]$ the box state constraint $\mathcal{X} = \{\boldsymbol{x}\ :\ -10 \leq x \leq 10,\ 0 \leq y \leq 4\}$. 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 $|\theta| \leq \pi/12$ and $\|\dot{p}\|_\infty \leq .1$. 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 $|x| < 1,\ y \leq 0$. Therefore, we take the recovery region $\mathcal{X}_R$ as all landed states with $|x| \geq 1$.

Clearly, $\mathcal{X}_R$ is a safe recovery region under $\pi_R(\boldsymbol{y}) = 0$ under the true dynamics (1).[2] For the drone to cross the road, we need to maintain recoverability with two disjoint recovery regions. Rather than add a mixed-integer constraint to (7), we solve multiple versions of (7) at each time step, one for each recovery region, and select the feasible input with lowest cost. We compare our approach with another naive baseline, which we label the Unsafe Recovery 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 region, slows down, and then switches to the leftmost recovery region once a feasible trajectory is found. In contrast, the Unsafe Recovery 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 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 $\rho := (\text{weather type}, t_{\mathrm{start}}, \text{severity})$ as a triplet indicating the weather type, severity level, and starting time from which the visibility starts to degrade, so that in the deployment context P_ρ, we randomly sample an environment

[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 $\hat{x}$ 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 (7) is infeasible, a simple fix first proposed in [40]. We did not observe recursive feasibility issues in the simulations.

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, so that the recovery set is given as $\mathcal{X}_R = \{\boldsymbol{x} \,:\, v = 0,\ |y| \leq 6\}$, which is invariant[1] 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 as $\mathcal{E} = \{\boldsymbol{e} = \hat{\boldsymbol{x}} - \boldsymbol{x} \,:\, |\hat{\theta} - \theta| \leq 7^\circ,\ |\hat{y} - y| \leq 1.3\text{m}\}$, to include at most 7 degree HE and 1.3m CTE.

We record 100 training trajectories using the Fallback-Safe MPC (7) and a ground-truth supervisor $w(\cdot) = 1 - \mathbb{1}\{\boldsymbol{e}_t \in \mathcal{E}\}$ to calibrate Algorithm 2, and then evaluate on 900 test trajectories with environments sampled i.i.d. from P_ρ using and Algorithms 1-2. This ensures we satisfy Definition 1 by corollary 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. 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 shows that over 80% would have led to an aircraft failure had we not interfered. This shows that our framework is effective at avoiding robot failures, and experiences few unnecessary interruptions with an effective OOD detection heuristic like the autoencoder reconstruction loss.