2023 IEEE International Conference on Robotics and Automation (ICRA 2023)
May 29 - June 2, 2023. London, UK

# Safeguarding Learning-Based Planners Under Motion and Sensing Uncertainties Using Reachability Analysis

Akshay Shetty[1], Adam Dai[2], Alexandros Tzikas[1] and Grace Gao[1]

***Abstract*— Learning-based trajectory planners in robotics have attracted growing interest given their ability to plan for complex tasks. These planners are typically trained in simulation under nominal conditions before being implemented on real robots. However, in real settings, the presence of motion and sensing uncertainties causes the robot to deviate from planned reference trajectories potentially leading to unsafe outcomes such as collisions. In this paper we present a reachability analysis to predict such deviations and to evaluate robot safety along reference trajectories. We then use the reachability analysis to safeguard a learning-based planner. Finally, we demonstrate the applicability of our safeguarding algorithm for learning-based planners via multiple simulations and real robot experiments.**

## I. Introduction

There has been growing interest in applying learning-based planners for challenging ground and aerial robotic applications, given their ability to learn complex relationships between the robot and its environment [1]–[7]. These planners commonly use neural networks which require a large number of training samples, and hence are typically trained in simulation before being implemented on a real robot. A key research question is how to ensure safety while implementing these learning-based planners on a real robot, where safety refers to avoiding a set of unsafe states (positions, velocities, orientations, etc.), such as obstacles for collision avoidance.

For real robots, planning safe trajectories is challenging due to the presence of motion and sensing uncertainties. Here, motion uncertainties refer to errors between the actual and the expected robot motion arising due to unmodeled effects such as wheel slips or wind disturbances. Sensing uncertainties refer to errors in sensor measurements such as in landmark (or feature) detection and tracking for visual sensors (camera, LiDAR) and signal reflections for Global Positioning System (GPS). These uncertainties cause the robot to have errors in its state estimate, and consequently cause it to deviate from a planned reference trajectory. Thus, it is important to account for these uncertainties while implementing learning-based planners on a real robot.

Recent works [8]–[13] have explored methods to safeguard learning-based planners. However, these works either do not explicitly account for uncertainties [8], [9] or focus solely on motion uncertainties and do not account for sensing uncertainties and state estimation [10]–[13]. One approach to evaluating safety under uncertainties is to predict deviations from reference trajectories. In our previous work [14] we present a reachability analysis where the reachable sets capture possible deviations from a reference trajectory while accounting for both motion and sensing uncertainties along with state estimation. However, the reachable set at any time instant is computed non-recursively as a function of all past uncertainties along the reference trajectory [14, Eqn. (31)], resulting in a higher computational load as the trajectory length grows. Thus, the reachability analysis in [14] is not suitable for safeguarding planners on real robots which have real-time computational constraints.

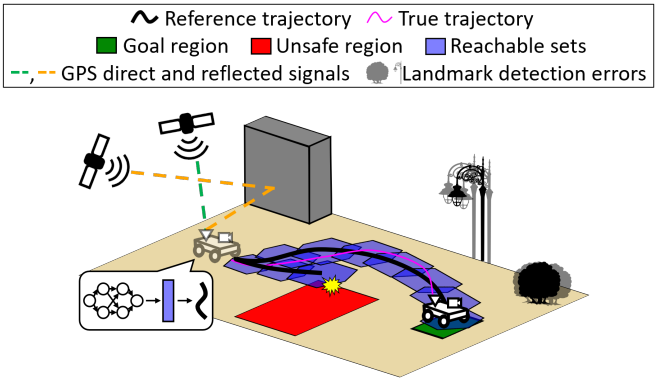


Fig. 1: Presence of motion uncertainties (such as wheel slips) and sensing uncertainties (such as visual landmark detection errors and GPS signal reflections) cause a robot to deviate from reference trajectories. In order to safeguard a learning-based planner against these uncertainties, we present a reachability analysis to evaluate safety of reference trajectories, and attempt to find alternate safe trajectories nearby.

In this paper we present a recursive reachability analysis and use it to safeguard a learning-based planner under motion and sensing uncertainties. Our main contributions are:

1) We first derive a reachability analysis to capture possible robot deviations from a reference trajectory while accounting for motion and sensing uncertainties along with state estimation. As opposed to [14], the reachable sets are recursively computed as a function of quantities only from the current and previous time indices.
2) Next, we integrate the above reachability analysis with a safeguarding algorithm to ensure safety while implementing learning-based planners.
3) Finally, we perform multiple simulations and real robot experiments to validate the performance of the safeguarding algorithm.

[1]Department of Aeronautics and Astronautics, Stanford University, CA 94305, USA {ashetty2, alextzik, gracegao}@stanford.edu
[2]Department of Electrical Engineering, Stanford University, CA 94305, USA addai@stanford.edu

## II. Related Work

**Reachability analysis** is a commonly used tool to predict robot deviations from reference trajectories. In [15]–[18], the authors present methods to compute reachable sets offline that are used to evaluate safety during execution. However, these methods focus on robot deviations due to differences between a deterministic high-fidelity motion model and a low-fidelity planning model and do not explicitly account for sensing uncertainties. Similarly, the work in [19] computes reachable sets while accounting for motion uncertainties, but does not account for sensing uncertainties. The authors in [20], [21] build on [19] to include sensing uncertainties. However, these works do not explicitly model a state estimation process, and thus do not model the effect of the motion and sensing uncertainties on the state estimation errors. [22] and [23] present methods to evaluate safety of actions using control barrier functions and contraction theory respectively. However, these methods focus only on a single camera sensor and require at least a subset of the state to be invertible from the sensed measurement, which is not always feasible (e.g., time instants with a single ranging measurement). Additionally, these methods [22], [23] consider the worst-case (bounded) motion and sensing errors which leads to conservativeness. In contrast, our framework allows for modeling of stochastic errors. In our previous work [14] we present a reachability analysis that accounts for both motion and sensing uncertainties, along with the commonly used Extended Kalman Filter (EKF) [24] for state estimation. However, as discussed in Sec. I, the reachability analysis in [14] is not recursive, making it unsuitable for real robots which typically have real-time constraints.

The concept of **safeguarding planners** has been explored in literature: evaluate safety of a reference trajectory suggested by the planner, and if it is unsafe, apply a fail-safe maneuver [25]. Given the growing interest in learning-based planners, recent works [8]–[13] have applied safeguarding in this domain. In [8] and [9], the authors safeguard learning-based planners using reachability analysis based on [15] and [18] respectively. However, as mentioned above, these methods do not account for sensing uncertainties. In [10]–[13], the authors present methods to safeguard a learning-based planner by ensuring that a safe backup trajectory always exists. This is done by checking for a backup trajectory using a model-predictive-control based optimization method [10], [11], or by verifying safety of backup trajectories using a sampling-based method [12], [13]. However, these works do not explicitly account for robot sensing uncertainties and state estimation, and focus solely on the motion uncertainties. [26] evaluates the safety of a neural network-based planner by predicting deviations from reference trajectories. However, their method requires the network to have a feedforward architecture with specifically ReLU activations, and additionally does not plan alternate trajectories if the network trajectory is unsafe. In contrast to these works, our safeguarding method accounts for deviations due to motion and sensing uncertainties, and does not require the learning-based planner to have any specific network architecture.

## III. Problem Formulation

### A. Robot Setup

We assume that the robot's motion and sensing models are defined as follows:

$$\mathbf{x}_{k+1} = f(\mathbf{x}_k, \mathbf{u}_k) + \mathbf{w}_{k+1}, \tag{1}$$

$$\mathbf{z}_k = h(\mathbf{x}_k) + \mathbf{v}_k, \tag{2}$$

where $\mathbf{x}_k$ is the state vector, $\mathbf{u}_k$ is the control input vector, $\mathbf{z}_k$ is the measurement vector, and $f$ and $h$ are nonlinear functions representing the motion and sensing respectively. Note that here $\mathbf{z}_k$ can be obtained from any sensor such as a localization measurement from visual sensors [27], [28] or ranging measurement from GPS satellites [29]. $\mathbf{w}_k$ is the motion model error vector modeled as a zero-mean Gaussian distribution $\mathbf{w}_k \sim \mathcal{N}(\mathbf{0}, Q_k)$, and $\mathbf{v}_k$ is the sensing model error vector containing two components, i.e., $\mathbf{v}_k = \mathbf{b}_k + \boldsymbol{\epsilon}_k$, where $\boldsymbol{\epsilon}_k$ is modeled as a zero-mean Gaussian distribution $\boldsymbol{\epsilon}_k \sim \mathcal{N}(\mathbf{0}, R_k)$ and $\mathbf{b}_k$ is a bounded additional bias $\mathbf{b}_k \in \left[-\overline{\mathbf{b}}_k, \overline{\mathbf{b}}_k\right]$. Examples of $\mathbf{b}_k$ in sensor measurements include: biases in landmark-based positioning due to landmark detection or misassociation errors [30], biases in GPS measurements due to signal reflections [29], and biases in inertial measurement units (IMU) [31].

The onboard planning and control architecture for the robot is shown in Fig. 2. For tracking the reference trajectory, we assume that the robot implements a state feedback control law [32] along with an EKF, which is commonly used for state estimation. The state feedback controller computes the control input $\mathbf{u}_k$ as $\mathbf{u}_k = \check{\mathbf{u}}_k - K_k(\hat{\mathbf{x}}_k - \check{\mathbf{x}}_k)$, where $\check{\mathbf{x}}_k$ and $\check{\mathbf{u}}_k$ are the reference state and control inputs respectively, $K_k$ is the linear feedback gain which can be obtained from existing control design methods such as linear-quadratic regulator (LQR) [32], and $\hat{\mathbf{x}}_k$ is the state estimate. Here $\check{\mathbf{x}}_k$ and $\check{\mathbf{u}}_k$ adhere to the motion model as $\check{\mathbf{x}}_{k+1} = f(\check{\mathbf{x}}_k, \check{\mathbf{u}}_k)$.

The EKF used for state estimation comprises of a prediction step and a correction step computed as:

$$\begin{aligned}
\bar{\mathbf{x}}_k &= f(\hat{\mathbf{x}}_{k-1}, \mathbf{u}_{k-1}), \\
\bar{P}_k &= A_{k-1} P_{k-1} A_{k-1}^\top + Q_k, \\
L_k &= \bar{P}_k C_k^\top (C_k \bar{P}_k C_k^\top + \hat{R}_k)^{-1}, \\
\hat{\mathbf{x}}_k &= \bar{\mathbf{x}}_k + L_k(\mathbf{z}_k - h(\bar{\mathbf{x}}_k)), \\
P_k &= \bar{P}_k - L_k C_k \bar{P}_k,
\end{aligned} \tag{3}$$

where $\bar{\mathbf{x}}_k$ is the predicted state, $\bar{P}_k$ and $P_k$ are the predicted and the corrected state estimation covariances, $L_k$ is the Kalman gain, and $A_k = \left.\frac{\partial f}{\partial \mathbf{x}}\right|_{\substack{\mathbf{x}=\check{\mathbf{x}}_k \\ \mathbf{u}=\check{\mathbf{u}}_k}}$ and $C_k = \left.\frac{\partial h}{\partial \mathbf{x}}\right|_{\mathbf{x}=\check{\mathbf{x}}_k}$. For the measurement covariance matrix $\hat{R}_k$ in Eqn. (3), we choose an over-bounding scaled approximation [33]–[35] with the $i^{\text{th}}$ diagonal element computed as $\hat{R}_k^{[i,i]} = \left[\left(\overline{\mathbf{b}}_k^{[i]} + m\sqrt{R_k^{[i,i]}}\right)/m\right]^2$, where $m$ denotes a desired $m\sigma$ confidence level for over-bounding [14]. Note that while we assume a state feedback controller along with an EKF, the reachability analysis presented later in Section IV can be potentially extended to other control and state estimation architectures by using the corresponding equations.

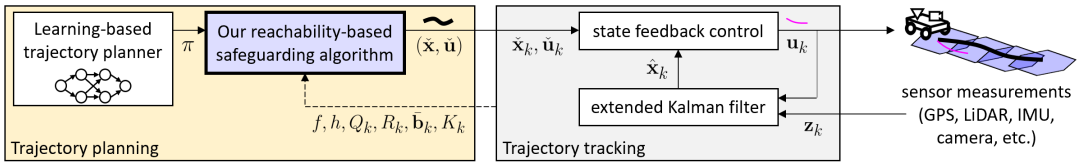


Fig. 2: Our algorithm takes as input a distribution of reference trajectories from a learning-based planner along with the motion and sensing models, uncertainties and feedback gain matrices used by the trajectory tracker. The output is a single reference trajectory that ensures safety while accounting for deviations in the trajectory tracking.

### B. Safeguarding a Learning-based Planner

Given the above robot motion and sensing uncertainties along with the onboard architecture in Fig. 2, the objective of our algorithm is to ensure safety while implementing a learning-based planner. For our problem, we assume that such a planner has already been trained and is available for implementation. Thus, the inputs to our algorithm are: a learning-based planner that outputs a distribution $\pi$ over reference trajectories $(\check{\mathbf{x}}, \check{\mathbf{u}})$; the initial reference state $\check{\mathbf{x}}_0$, state estimate $\hat{\mathbf{x}}_0$ and estimation covariance $P_0$; the motion and sensing models $f$ and $h$, uncertainties $Q_k$, $R_k$ and $\bar{\mathbf{b}}_k$ and the feedback gain $K_k$ used by the trajectory tracker (Fig. 2); and the set of unsafe states $\mathcal{X}_k^{\text{unsafe}}$ to avoid. The desired output from our algorithm is a single reference trajectory $(\check{\mathbf{x}}, \check{\mathbf{u}})$ for the robot to track, such that safety with respect to $\mathcal{X}_k^{\text{unsafe}}$ is ensured.

## IV. Reachability Analysis

In this section we recursively compute reachable sets for a robot along a single reference trajectory. Given a candidate reference trajectory $(\check{\mathbf{x}}, \check{\mathbf{u}})_{k_1:k_2}$ between two time indices $k_1$ and $k_2$, the objective of the computed sets is to predict deviations in the state $\mathbf{x}_k \ \forall \ k \in [k_1, k_2]$.

In order to capture these deviations, we use the probabilistic zonotope [19] set representation which contains two components—a bounded component modeled by a zonotope [36] and a stochastic component modeled by a Gaussian distribution—thus making it suitable to model the uncertainties described in Sec. III-A. A zonotope $\mathscr{Z}$ is defined as follows [19]:

$$\mathscr{Z} = \left\{ \mathbf{x} \in \mathbb{R}^d \mid \mathbf{x} = \mathbf{c} + G\boldsymbol{\beta}, \ \|\boldsymbol{\beta}\|_\infty \leq 1 \right\}, \tag{4}$$

where $d$ is the zonotope dimension, $\mathbf{c}$ is the zonotope center and $G$ is the zonotope generator matrix. We represent a probabilistic zonotope concisely as $\mathcal{P} = \mathcal{Z}(\mathbf{c}, G, \Sigma)$, where $\Sigma$ is the Gaussian covariance of the stochastic component. We use the Minkowski sum $(\oplus)$ and linear transform operations [19] to propagate these sets (similar to how states are propagated) through the closed loop dynamics model, in order to capture possible deviations from the reference trajectory.

We begin computing the reachable sets by linearizing Eqn. (1) about the reference trajectory $(\check{\mathbf{x}}_k, \check{\mathbf{u}}_k)$ and substituting the control input $\mathbf{u}_k$ from Sec. III-A:

$$\begin{aligned} \mathbf{x}_{k+1} = A_k \left(\mathbf{x}_k - \check{\mathbf{x}}_k\right) - B_k K_k \left(\hat{\mathbf{x}}_k - \check{\mathbf{x}}_k\right) \\ + f\left(\check{\mathbf{x}}_k, \check{\mathbf{u}}_k\right) + \mathbf{l}_k^f + \mathbf{w}_{k+1}, \end{aligned} \tag{5}$$

where $A_k$ is defined above in Eqn. (3), $B_k = \frac{\partial f}{\partial \mathbf{u}}\Big|_{\substack{\mathbf{x}=\check{\mathbf{x}}_k \\ \mathbf{u}=\check{\mathbf{u}}_k}}$ and $\mathbf{l}_k^f$ is the corresponding linearization error. Next on substituting Eqn. (2) into the correction step of Eqn. (3), we get:

$$\hat{\mathbf{x}}_{k+1} = \bar{\mathbf{x}}_{k+1} + L_{k+1}\left(h(\mathbf{x}_{k+1}) - h(\bar{\mathbf{x}}_{k+1})\right) + L_{k+1}\mathbf{v}_{k+1}. \tag{6}$$

Here the sensing models can be linearized about the reference trajectory as:

$$h(\mathbf{x}_k) = h(\check{\mathbf{x}}_k) + C_k(\mathbf{x}_k - \check{\mathbf{x}}_k) + \mathbf{l}_k^h, \tag{7}$$

$$h(\bar{\mathbf{x}}_k) = h(\check{\mathbf{x}}_k) + C_k(\bar{\mathbf{x}}_k - \check{\mathbf{x}}_k) + \mathbf{l}_k^{\bar{h}}, \tag{8}$$

where $\mathbf{l}_k^h$ and $\mathbf{l}_k^{\bar{h}}$ are linearization errors. Thus, substituting Eqn. (7) in Eqn. (6), we get:

$$\begin{aligned} \hat{\mathbf{x}}_{k+1} = \bar{\mathbf{x}}_{k+1} + L_{k+1}C_{k+1}\left(\mathbf{x}_{k+1} - \bar{\mathbf{x}}_{k+1}\right) \\ + L_{k+1}\left(\mathbf{l}_k^h - \mathbf{l}_k^{\bar{h}}\right) + L_{k+1}\mathbf{v}_{k+1}. \end{aligned} \tag{9}$$

Here the predicted state $\bar{\mathbf{x}}_k$ can be linearized about the reference trajectory as:

$$\bar{\mathbf{x}}_{k+1} = A_k(\hat{\mathbf{x}}_k - \check{\mathbf{x}}_k) + B_k(\mathbf{u}_k - \check{\mathbf{u}}_k) + f(\check{\mathbf{x}}_k, \check{\mathbf{u}}_k) + \mathbf{l}_k^{\bar{f}}, \tag{10}$$

where $\mathbf{l}_k^{\bar{f}}$ is the linearization error. Thus, substituting Eqns. (5) and (10) in Eqn. (9), we get:

$$\begin{aligned} \hat{\mathbf{x}}_{k+1} = \ & \left(A_k - B_k K_k\right)\left(\hat{\mathbf{x}}_k - \check{\mathbf{x}}_k\right) + f\left(\check{\mathbf{x}}_k, \check{\mathbf{u}}_k\right) + \mathbf{l}_k^{\bar{f}} \\ & + L_{k+1}C_{k+1}\Big(A_k\left(\mathbf{x}_k - \hat{\mathbf{x}}_k\right) + \mathbf{l}_k^f - \mathbf{l}_k^{\bar{f}} \\ & + \mathbf{w}_{k+1}\Big) + L_{k+1}\Big(\mathbf{l}_k^h - \mathbf{l}_k^{\bar{h}}\Big) + L_{k+1}\mathbf{v}_{k+1}. \end{aligned} \tag{11}$$

Combining Eqns. (5) and (11) along with motion model from Sec. III-A, the state $\mathbf{x}_k$ and state estimate $\hat{\mathbf{x}}_k$ can be jointly written as:

$$\begin{aligned} \begin{bmatrix} \mathbf{x}_{k+1} \\ \hat{\mathbf{x}}_{k+1} \end{bmatrix} = & \begin{bmatrix} \check{\mathbf{x}}_{k+1} \\ \check{\mathbf{x}}_{k+1} \end{bmatrix} + \Phi_k \begin{bmatrix} \mathbf{x}_k - \check{\mathbf{x}}_k \\ \hat{\mathbf{x}}_k - \check{\mathbf{x}}_k \end{bmatrix} + \begin{bmatrix} I \\ L_{k+1}C_{k+1} \end{bmatrix} \mathbf{w}_{k+1} \\ & + \begin{bmatrix} O \\ L_{k+1} \end{bmatrix} \mathbf{v}_{k+1} + \begin{bmatrix} I \\ L_{k+1}C_{k+1} \end{bmatrix} \mathbf{l}_k^f \\ & + \begin{bmatrix} O \\ I - L_{k+1}C_{k+1} \end{bmatrix} \mathbf{l}_k^{\bar{f}} + \begin{bmatrix} O \\ L_{k+1} \end{bmatrix} \mathbf{l}_k^h + \begin{bmatrix} O \\ -L_{k+1} \end{bmatrix} \mathbf{l}_k^{\bar{h}}, \end{aligned} \tag{12}$$

where $\Phi_k = \begin{bmatrix} A_k & -B_k K_k \\ L_{k+1}C_{k+1}A_k & A_k - B_k K_k - L_{k+1}C_{k+1}A_k \end{bmatrix}$, and $I$ and $O$ represent identity and zero matrices of appropriate sizes. The above equation governs the growth of a single state $\mathbf{x}_k$ and state estimate $\hat{\mathbf{x}}_k$ given values for the motion error, the sensing error and the linearization errors. However, during execution the exact error values are not known. Thus, we replace these error values by the sets to which they belong, and hence compute the set of states $\mathcal{X}_k$ and state

estimates $\hat{\mathcal{X}}_k$ as:

$$\begin{bmatrix}\mathcal{X}_{k+1}\\ \hat{\mathcal{X}}_{k+1}\end{bmatrix} = \begin{bmatrix}\check{\mathbf{x}}_{k+1}\\ \check{\mathbf{x}}_{k+1}\end{bmatrix} \oplus \Phi_k \begin{bmatrix}\mathcal{X}_k - \check{\mathbf{x}}_k\\ \hat{\mathcal{X}}_k - \check{\mathbf{x}}_k\end{bmatrix} \oplus \begin{bmatrix}I\\ L_{k+1}C_{k+1}\end{bmatrix}\mathcal{W}_{k+1} \oplus \begin{bmatrix}O\\ L_{k+1}\end{bmatrix}\mathcal{V}_{k+1} \oplus \begin{bmatrix}I\\ L_{k+1}C_{k+1}\end{bmatrix}\mathcal{L}_k^f \oplus \begin{bmatrix}O\\ I - L_{k+1}C_{k+1}\end{bmatrix}\mathcal{L}_k^{\bar{f}} \oplus \begin{bmatrix}O\\ L_{k+1}\end{bmatrix}\mathcal{L}_k^h \oplus \begin{bmatrix}O\\ -L_{k+1}\end{bmatrix}\mathcal{L}_k^{\bar{h}}, \tag{13}$$

where $\oplus$ is the Minkowski sum operation [19], $\mathcal{W}_k$ and $\mathcal{V}_k$ are sets representing the motion and sensing uncertainties respectively, and $\mathcal{L}$ represent the corresponding sets of linearization errors. Given the robot setup in Sec. III-A, $\mathcal{W}_k$ and $\mathcal{V}_k$ can be written as probabilistic zonotopes $\mathcal{W}_k = \mathcal{Z}(\mathbf{0}, O, Q_k)$ and $\mathcal{V}_k = \mathcal{Z}(\mathbf{0}, \text{diag}(\overline{\mathbf{b}}_k), R_k)$. For the sets of linearization errors $\mathcal{L}$, we approximate them as zero-mean Gaussian distributions as done in [37, Sec.V.C]. In Eqn. (13) note that $\mathcal{X}_{k+1}$ is only a function of quantities from time indices $k+1$ and $k$, as opposed to quantities from all time indices $k \in [0, k+1]$ [14, Eqn. (31)]. Thus, initializing the sets as $\mathcal{X}_0 = \mathcal{Z}(\check{\mathbf{x}}_0, O, P_0)$ and $\hat{\mathcal{X}}_0 = \mathcal{Z}(\check{\mathbf{x}}_0, O, O)$, and using Eqn. (13) allows us to recursively compute $\mathcal{X}_k$ $\forall\ k \in [k_1, k_2]$, capturing possible deviations in the robot state $\mathbf{x}_k$ from the reference trajectory $(\check{\mathbf{x}}, \check{\mathbf{u}})_{k_1:k_2}$.

## V. Safeguarding a Learning-Based Planner

This section describes our algorithm for safeguarding a learning-based planner. Alg. 1 lists the steps for the algorithm and Fig. 3 provides an illustration. Here we discuss the implementation for a general segment-wise planner (plan trajectory in segments; execute current segment while planning the next one), which can be extended to an offline planner (plan entire trajectory before execution) or an instantaneous planner (plan for only the next time instant) if desired. Note that the reachable sets (computed using Eqn. (13) are not a function of the segment time-horizon.

In general for planning the $n^{\text{th}}$ trajectory segment, we first obtain a distribution of reference trajectories $\pi_n$ from the learning-based planner. We then extract the maximum likelihood reference trajectory $(\check{\mathbf{x}}, \check{\mathbf{u}})^0_{k_n^s:k_n^e}$, where $k_n^s$ and $k_n^e$ are time indices for the start and end of the $n^{\text{th}}$ segment trajectory ($k_n^s = k_{n-1}^e$). Next, the max. likelihood trajectory is appended by a fail-safe maneuver to obtain $(\check{\mathbf{x}}, \check{\mathbf{u}})^0_{k_n^s:k_n^{f,0}}$ (i.e. $\boldsymbol{\tau}_n^0$ in Alg. 1), where $k_n^{f,0}$ is the time index for the end of the maneuver. Here we assume that a fail-safe maneuver such as braking to a stop for ground robots or hovering for aerial robots exists. We then use our reachability analysis from Sec. IV to evaluate robot safety along the reference trajectory, $\psi_n^i = \text{isReachSafe}\big((\check{\mathbf{x}}, \check{\mathbf{u}})^i_{k_n^s:k_n^{f,i}}\big)$. Here we obtain $\psi_n^i$ as:

$$\psi_n^i = \begin{cases}\text{True} & \big(\mathcal{X}_k^{m\sigma} \cap \mathcal{X}_k^{\text{unsafe}}\big) = \emptyset\ \forall\ k \in \big[k_n^s : k_n^{f,i}\big]\\ \text{False} & \text{otherwise}\end{cases}, \tag{14}$$

where $i = 0$ for the max. likelihood reference trajectory, $\mathcal{X}_k^{m\sigma}$ represents $m\sigma$ confidence sets along the reference trajectory obtained from $\mathcal{X}_k$, and $\mathcal{X}_k^{\text{unsafe}}$ represents the set of unsafe states to avoid (could potentially be dynamic where the set changes with time index $k$). Note that larger values of $m$ result in larger confidence sets $\mathcal{X}_k^{m\sigma}$ leading to more conservative planned trajectories. If $\psi_n^0$ is *True*, we select $(\check{\mathbf{x}}, \check{\mathbf{u}})^0_{k_n^s:k_n^e}$ as the reference trajectory for the $n^{\text{th}}$ segment along with $(\check{\mathbf{x}}, \check{\mathbf{u}})^0_{k_n^e:k_n^{f,0}}$ as the fail-safe maneuver. However if $\psi_n^0$ is *False*, we sample additional reference trajectories from $\pi_n$, append a fail-safe maneuver to them (obtaining $\boldsymbol{\tau}_n^i$ in Alg. 1), and evaluate their safety $\psi_n^i$. The sampling step is repeated till a specified maximum segment planning time is reached. If at least one of $\psi_n^i$ is *True*, then we select the safe sampled trajectory that is closest to the max. likelihood reference trajectory along with the corresponding fail-safe maneuver. If none of $\psi_n^i$ are *True*, then we apply the previously selected fail-safe maneuver $(\check{\mathbf{x}}, \check{\mathbf{u}})^i_{k_{n-1}^e:k_{n-1}^{f,i}}$. Note that this fail-safe maneuver has been previously checked using Eqn. (14), and thus ensures safety in the event that no trajectory is found. As the robot executes the planned trajectory for the $n^{\text{th}}$ segment, it begins planning for the $(n+1)^{\text{th}}$ segment. Our algorithm stops once a fail-safe maneuver has been applied or the goal has been reached.

**Algorithm 1** Safeguarding a Learning-based Planner

**Input**: $\check{\mathbf{x}}_0$, $\hat{\mathbf{x}}_0 = \check{\mathbf{x}}_0$, $P_0$, $f$, $h$, $Q_k$, $R_k$, $\overline{\mathbf{b}}_k$, $K_k$ $\mathcal{X}_k^{\text{unsafe}}$
Initialize segment number $n = 1$
**while** *goal not reached* **and** *fail-safe maneuver not applied* **do**
  Get distribution of reference trajectories $\pi_n$ for $n^{\text{th}}$ segment from learning-based planner
  Get max. likelihood reference trajectory from $\pi_n$ and append with fail-safe maneuver $\rightarrow \boldsymbol{\tau}_n^0$
  Check if $\boldsymbol{\tau}_n^0$ is safe $\rightarrow \psi_n^0$
  **if** $\psi_n^0$ *is True* **then**
    Select $\boldsymbol{\tau}_n^0$ as output for $n^{\text{th}}$ segment
  **else**
    **while** *max segment planning time is not reached* **do**
      Sample $i^{\text{th}}$ reference trajectory from $\pi_n$ and append with fail-safe maneuver $\rightarrow \boldsymbol{\tau}_n^i$
      Check if $\boldsymbol{\tau}_n^i$ is safe $\rightarrow \psi_n^i$
    **if** *at least one* $\psi_n^i$ *is True* **then**
      Find $\boldsymbol{\tau}_n^i$ closest to $\boldsymbol{\tau}_n^0$ such that $\psi_n^i$ is True
      Select $\boldsymbol{\tau}_n^i$ as output for $n^{\text{th}}$ segment
    **else**
      Apply fail-safe maneuver previously selected for $(n-1)^{\text{th}}$ segment
  Proceed to plan for next segment, $n = n + 1$.

## VI. Results

In order to validate the safeguarding algorithm, we present simulations and experiments for a ground rover (can be applied to other robot models if desired). The rover is modeled as a Dubins vehicle [38] with the following motion and sensing models:

$$\underbrace{\begin{bmatrix}x_k\\ y_k\\ \theta_k\\ v_k\end{bmatrix}}_{\mathbf{x}_k} = \underbrace{\begin{bmatrix}x_{k-1}\\ y_{k-1}\\ \theta_{k-1}\\ v_{k-1}\end{bmatrix} + \begin{bmatrix}v_{k-1}\cos(\theta_{k-1})\Delta k\\ v_{k-1}\sin(\theta_{k-1})\Delta k\\ \omega_{k-1}\Delta k\\ a_{k-1}\Delta k\end{bmatrix}}_{f(\mathbf{x}_{k-1}, \mathbf{u}_{k-1})} + \mathbf{w}_k, \quad \mathbf{z}_k = \underbrace{\begin{bmatrix}x_k\\ y_k\\ \theta_k\end{bmatrix}}_{h(\mathbf{x}_k)} + \mathbf{v}_k,$$

where $(x_k, y_k)$ is 2D position, $\theta_k$ is heading, $v_k$ is longitudi-

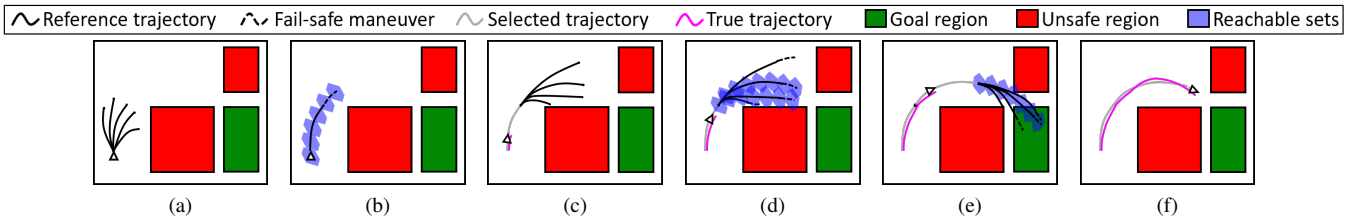


Fig. 3: (a) We begin by obtaining a distribution of reference trajectories from a learning-based planner, and (b) evaluating safety of the max. likelihood reference trajectory appended by a fail-safe maneuver. (c) If it is safe we proceed to execute the trajectory and begin planning for the next trajectory segment. (d) If the max. likelihood reference trajectory is unsafe, we sample trajectories from the learning-based planner distribution, append them with fail-safe maneuvers and evaluate their safety. (e) We proceed to execute the nearest safe sampled trajectory and begin planning for the next trajectory segment. (f) If none of the sampled trajectories are found to be safe, we proceed to execute the previously checked fail-safe maneuver.

nal velocity, $\omega_k$ is angular velocity input, $a_k$ is longitudinal acceleration input and $\Delta k$ is the discrete time-step.

### A. Learning-based Planner Setup

We first train a learning-based planner in simulation without uncertainty, i.e., $\mathbf{w}_k = \mathbf{0}$, $\mathbf{v}_k = \mathbf{0}$ and $\hat{\mathbf{x}}_k = \mathbf{x}_k$. We train the planner for a simple task of navigating to a fixed goal region while avoiding two obstacles. Note that, if desired, our algorithm can be used for arbitrary planners performing more complex tasks. A fully connected network with hidden layers of sizes $[64, 64, 64, 32]$ and LeakyReLU activations is trained using proximal policy optimization [39]. The inputs to the network are the state $\mathbf{x}_k$ and the obstacle configurations, and the output is a distribution over desired angular and longitudinal velocities (between $\pm 0.5\,\mathrm{rads}^{-1}$ and $\pm 0.5\,\mathrm{ms}^{-1}$ respectively) for a trajectory segment of time-horizon 3 s. We use a trajectory parameterization [18] to convert the network output to a distribution $\pi_n$ over reference trajectories $(\check{\mathbf{x}}, \check{\mathbf{u}})_{k_n^s : k_n^e}^0$. The time-step was set to $\Delta k = 0.2\,\mathrm{s}$, which worked well for our experiments in Sec. VI-C.

### B. Simulation Results

Once the learning-based planner is trained, we simulate its implementation on the ground rover under motion and sensing uncertainties. For our simulations, we set the base uncertainties to be $Q_k = 10^{-4} \times \mathrm{diag}(1\,\mathrm{m}^2,\ 1\,\mathrm{m}^2,\ 5\,\mathrm{rad}^2,\ 1\,\mathrm{m}^2\mathrm{s}^{-2})$, $R_k = \mathrm{diag}(0.01\,\mathrm{m}^2,\ 0.01\,\mathrm{m}^2,\ 0.001\,\mathrm{rad}^2)$ and $\overline{\mathbf{b}}_k = [0.1\,\mathrm{m},\ 0.1\,\mathrm{m},\ 0\,\mathrm{rad}]^\top$. We then create 27 different uncertainty scenarios by multiplying each of $Q_k$, $R_k$ and $\overline{\mathbf{b}}_k$ by factors of $[1, 2, 3]$. For the confidence level in the safety evaluation in Eqn. (14), we set $m = 3$. Since the true state $\mathbf{x}_k$ is not known under uncertainty, we use the reference state $\check{\mathbf{x}}_k$ as input to the network. Fig. 4 shows our simulation results, where we observe that directly using the max. likelihood reference trajectories from the learning-based planner results in a significantly lower number of safe trajectories for larger uncertainty scenarios. On the other hand, while using our safeguarding algorithm, we observe that our reachability analysis is able to capture deviations in the rollouts and thus is able to ensure safety for all 27 scenarios. Additionally, we also compare the average time taken to compute reachable sets for a trajectory segment using the presented reachability analysis (Sec. IV) versus the analysis in the previous work [14]. As shown in Table I, our computation time remains low, whereas it increases for [14] as the trajectory grows longer and has more segments.

### C. Real-world Experiments

For experiments on a real robot, we setup a ground rover equipped with a Velodyne VLP-16 LiDAR for 2D landmark-based positioning, and a Xsens MTi-30 IMU for heading measurements. An i7 Intel NUC runs the planning and control architecture (outlined in Fig. 2) entirely onboard[1]. We set the motion and sensing uncertainties $Q_k$, $R_k$ and $\overline{\mathbf{b}}_k$ to the base uncertainty values mentioned in Sec. VI-B. The chosen $\overline{\mathbf{b}}_k$ corresponds to a real bias observed in the landmark-based positioning, which relies on detecting a cylindrical-shaped landmark with known position. Since the LiDAR detects points on the landmark surface, and the landmark radius is approximately $0.1\,\mathrm{m}$, this results in a $\pm 0.1\,\mathrm{m}$ bias in the position measurement. Fig. 5 shows our experimental results. We setup two different environments and conduct 20 runs in each environment: 10 directly using the max. likelihood reference trajectories obtained from the learning-based planner, and 10 using our safeguarding algorithm. We observe that directly implementing the learning-based planner leads to multiple unsafe runs, whereas our safeguarding algorithm results in all 20 runs being safe.

## VII. Conclusions

In this paper we presented an algorithm for safeguarding learning-based planners under motion and sensing uncertainties. We derived a recursive reachability analysis and then integrated it with a safeguarding algorithm to ensure safety using the computed reachable sets. Our algorithm was validated via multiple simulations and real-world experiments.

For our future work, we plan to address some limitations in this paper. Currently our reachability analysis specifically assumes that the robot uses a state feedback control and an EKF; we plan to derive the analysis for additional control and estimation architectures. For the safeguarding algorithm, instead of randomly sampling alternate trajectories, we plan to explore adaptive sampling strategies. Finally, we also plan to test our algorithm on additional robots (such as quadrotors) with more complex learning-based planners operating in cluttered environments with dynamic obstacles.

[1]Code available at: https://github.com/Stanford-NavLab/rover_ros_ws

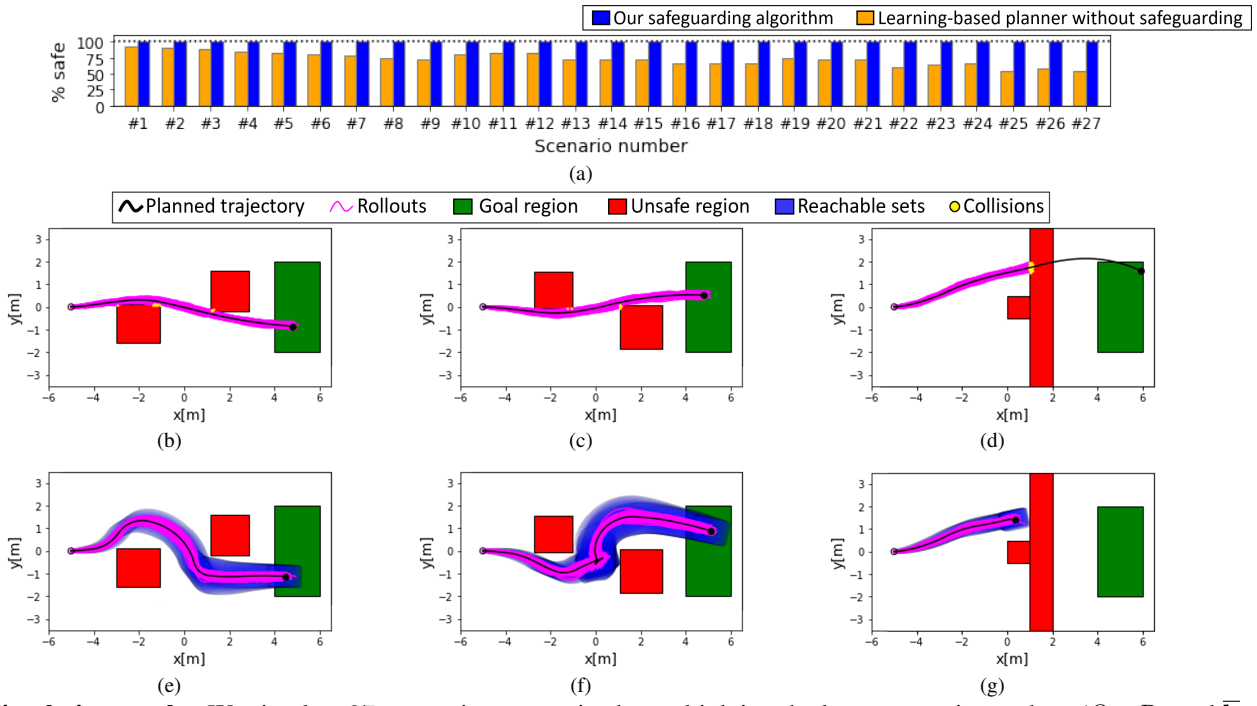


Fig. 4: **Simulation results.** We simulate 27 uncertainty scenarios by multiplying the base uncertainty values ($Q_k$, $R_k$ and $\overline{\mathbf{b}}_k$ in Section VI-B) by factors of $[1, 2, 3]$. For each uncertainty scenario we generate 50 random environments where we plan a trajectory directly using the max. likelihood reference trajectory from the learning-based planner and using our safeguarding algorithm. Then for each environment we simulate 100 trajectory rollouts tracking both the planned trajectories. (a) We observe that as the amount of uncertainty grows, the number of safe rollouts decreases significantly while implementing the learning-based planner directly. On the other hand, our planning algorithm ensures that all rollouts remain safe in each of the 27 scenarios. (b)-(c) Two example environments and (d) an out-of-distribution environment where implementing the learning-based planner directly leads to multiple unsafe rollouts. (e)-(f) While our algorithm results in longer trajectories, it safely guides the rollouts to the goal and (g) resorts to a fail-safe maneuver when a safe trajectory could not be found.

| Trajectory segment number | #1 | #5 | #10 | #15 | #20 |
|---|---|---|---|---|---|
| **Our reachability analysis (Sec. IV)** | 18.9 ms | 18.8 ms | 18.7 ms | 18.6 ms | 18.4 ms |
| [14] | 44.7 ms | 216.0 ms | 436.7 ms | 656.5 ms | 891.2 ms |

TABLE I: As the trajectory grows longer (more segments), our computation time remains low whereas it increases for [14].

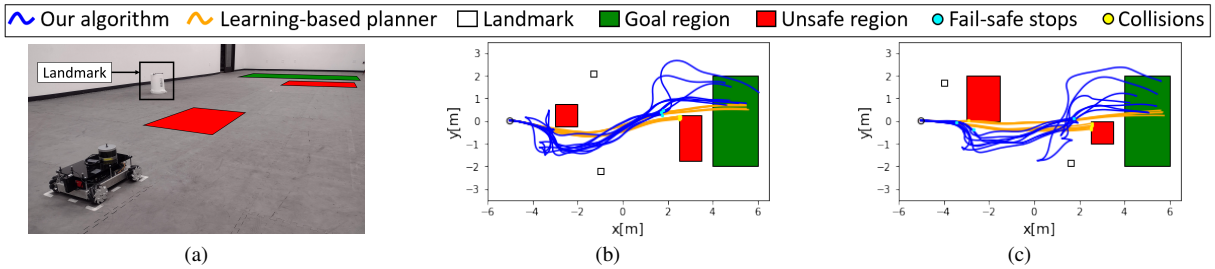


Fig. 5: **Real-world experiments**. (a) We conduct experiments on a ground rover which uses a LiDAR for landmark-based positioning and an IMU for heading measurements. (b)-(c) We setup two different environments and conduct 20 runs in each: 10 runs directly using the max. likelihood reference trajectories from the learning-based planner versus 10 runs using our safeguarding algorithm. The landmark was placed in one of the marked locations for half of the runs, and in the other marked location for the other half. We observe that using the learning-based planner directly gives shorter trajectories, but results in 5 and 7 unsafe runs in the two environments respectively. On the other hand, our safeguarding algorithm results in all 10 runs in both the environments being safe, with 7 runs reaching the goal region in both the environments and the rest applying a fail-safe maneuver. Note that our algorithm involves randomly sampling trajectories from the learning-based planner distribution (Alg. 1, line 10), which leads to different planned trajectories for each run resulting in some back-tracking motion and some fail-safe maneuvers.

## REFERENCES


[1] M. Bansal, A. Krizhevsky, and A. Ogale, "Chauffeurnet: Learning to Drive by Imitating the Best and Synthesizing the Worst," in *Robotics: Science and Systems*, 2019.

[2] B. Ichter, J. Harrison, and M. Pavone, "Learning Sampling Distributions for Robot Motion Planning," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 7087–7094.

[3] L. Yu, X. Shao, Y. Wei, and K. Zhou, "Intelligent Land-vehicle Model Transfer Trajectory Planning Method Based on Deep Reinforcement Learning," *Sensors*, vol. 18, no. 9, p. 2905, 2018.

[4] J. Ngiam, V. Vasudevan, B. Caine, Z. Zhang, H.-T. L. Chiang, J. Ling, R. Roelofs, A. Bewley, C. Liu, A. Venugopal *et al.*, "Scene Transformer: A Unified Architecture for Predicting Future Trajectories of Multiple Agents," in *International Conference on Learning Representations*, 2021.

[5] M. Samir, C. Assi, S. Sharafeddine, D. Ebrahimi, and A. Ghrayeb, "Age of Information Aware Trajectory Planning of UAVs in Intelligent Transportation Systems: A Deep Learning Approach," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 11, pp. 12 382–12 395, 2020.

[6] Y. Song and D. Scaramuzza, "Policy Search for Model Predictive Control With Application to Agile Drone Flight," *IEEE Transactions on Robotics*, 2022.

[7] G. Tong, N. Jiang, L. Biyue, Z. Xi, W. Ya, and D. Wenbo, "UAV Navigation in High Dynamic Environments: A Deep Reinforcement Learning Approach," *Chinese Journal of Aeronautics*, vol. 34, no. 2, pp. 479–489, 2021.

[8] J. F. Fisac, A. K. Akametalu, M. N. Zeilinger, S. Kaynama, J. Gillula, and C. J. Tomlin, "A General Safety Framework for Learning-based Control in Uncertain Robotic Systems," *IEEE Transactions on Automatic Control*, vol. 64, no. 7, pp. 2737–2752, 2018.

[9] Y. S. Shao, C. Chen, S. Kousik, and R. Vasudevan, "Reachability-based Trajectory Safeguard (RTS): A Safe and Fast Reinforcement Learning Safety Layer for Continuous Control," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 3663–3670, 2021.

[10] L. Hewing, K. P. Wabersich, M. Menner, and M. N. Zeilinger, "Learning-based Model Predictive Control: Toward Safe Learning in Control," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 3, pp. 269–296, 2020.

[11] K. P. Wabersich, L. Hewing, A. Carron, and M. N. Zeilinger, "Probabilistic Model Predictive Safety Certification for Learning-based Control," *IEEE Transactions on Automatic Control*, vol. 67, no. 1, pp. 176–188, 2021.

[12] S. Li and O. Bastani, "Robust Model Predictive Shielding for Safe Reinforcement Learning with Stochastic Dynamics," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 7166–7172.

[13] O. Bastani, S. Li, and A. Xu, "Safe Reinforcement Learning via Statistical Model Predictive Shielding," in *Robotics: Science and Systems*, 2021.

[14] A. Shetty and G. Gao, "Trajectory Planning Under Stochastic and Bounded Sensing Uncertainties Using Stochastic Reachability," in *Proceedings of the 33rd International Technical Meeting of The Satellite Division of the Institute of Navigation (ION GNSS+ 2020), St. Louis, MO, USA*, 2020.

[15] I. M. Mitchell, A. M. Bayen, and C. J. Tomlin, "A Time-dependent Hamilton-Jacobi Formulation of Reachable Sets for Continuous Dynamic Games," *IEEE Transactions on automatic control*, vol. 50, no. 7, pp. 947–957, 2005.

[16] M. Chen, S. Bansal, J. F. Fisac, and C. J. Tomlin, "Robust Sequential Trajectory Planning Under Disturbances and Adversarial Intruder," *IEEE Transactions on Control Systems Technology*, vol. 27, no. 4, pp. 1566–1582, 2018.

[17] H. Seo, D. Lee, C. Y. Son, C. J. Tomlin, and H. J. Kim, "Robust Trajectory Planning for a Multirotor Against Disturbance Based on Hamilton-Jacobi Reachability Analysis," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2019, pp. 3150–3157.

[18] S. Kousik, S. Vaskov, F. Bu, M. Johnson-Roberson, and R. Vasudevan, "Bridging the Gap Between Safety and Real-time Performance in Receding-horizon Trajectory Design for Mobile Robots," *The International Journal of Robotics Research*, vol. 39, no. 12, pp. 1419–1469, 2020.

[19] M. Althoff, "Reachability Analysis and its Application to the Safety Assessment of Autonomous Cars," Ph.D. dissertation, Technische Universität München, 2010.

[20] B. Schürmann, D. Heß, J. Eilbrecht, O. Stursberg, F. Köster, and M. Althoff, "Ensuring Drivability of Planned Motions Using Formal Methods," in *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2017, pp. 1–8.

[21] B. Schürmann, N. Kochdumper, and M. Althoff, "Reachset Model Predictive Control for Disturbed Nonlinear Systems," in *2018 IEEE Conference on Decision and Control (CDC)*. IEEE, 2018, pp. 3463–3470.

[22] R. K. Cosner, A. W. Singletary, A. J. Taylor, T. G. Molnar, K. L. Bouman, and A. D. Ames, "Measurement-robust Control Barrier Functions: Certainty in Safety with Uncertainty in State," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2021, pp. 6286–6291.

[23] G. Chou, N. Ozay, and D. Berenson, "Safe Output Feedback Motion Planning from Images via Learned Perception Modules and Contraction Theory," *arXiv preprint arXiv:2206.06553*, 2022.

[24] G. Welch, G. Bishop *et al.*, *An Introduction to The Kalman Filter*. Chapel Hill, NC, USA, 1995.

[25] S. Magdici and M. Althoff, "Fail-safe Motion Planning of Autonomous Vehicles," in *2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2016, pp. 452–458.

[26] M. Everett, G. Habibi, C. Sun, and J. P. How, "Reachability Analysis of Neural Feedback Loops," *IEEE Access*, vol. 9, pp. 163 938–163 953, 2021.

[27] J. Zhang and S. Singh, "LOAM: Lidar Odometry and Mapping in Real-time," in *Robotics: Science and Systems*, vol. 2, no. 9. Berkeley, CA, 2014, pp. 1–9.

[28] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardos, "ORB-SLAM: A Versatile and Accurate Monocular SLAM System," *IEEE transactions on robotics*, vol. 31, no. 5, pp. 1147–1163, 2015.

[29] Y. J. Morton, F. van Diggelen, J. J. Spilker Jr, B. W. Parkinson, S. Lo, and G. Gao, *Position, Navigation, and Timing Technologies in the 21st Century: Integrated Satellite Navigation, Sensor Systems, and Civil Applications*. John Wiley & Sons, 2021.

[30] G. Duenas Arana, O. Abdul Hafez, M. Joerger, and M. Spenko, "Integrity Monitoring for Kalman Filter-based Localization," *The International Journal of Robotics Research*, vol. 39, no. 13, pp. 1503–1524, 2020.

[31] N. Metni, J.-M. Pflimlin, T. Hamel, and P. Soueres, "Attitude and Gyro Bias Estimation for a VTOL UAV," *Control Engineering Practice*, vol. 14, no. 12, pp. 1511–1520, 2006.

[32] S. Skogestad and I. Postlethwaite, *Multivariable Feedback Control: Analysis and Design*. Citeseer, 2007, vol. 2.

[33] M. Susi, M. Andreotti, M. Aquino, and A. Dodson, "Tuning a Kalman Filter Carrier Tracking Algorithm in the Presence of Ionospheric Scintillation," *GPS Solutions*, vol. 21, no. 3, pp. 1149–1160, 2017.

[34] P. N. Raanes, M. Bocquet, and A. Carrassi, "Adaptive Covariance Inflation in the Ensemble Kalman Filter by Gaussian Scale Mixtures," *Quarterly Journal of the Royal Meteorological Society*, vol. 145, no. 718, pp. 53–75, 2019.

[35] Y. Yang and W. Gao, "An Optimal Adaptive Kalman Filter," *Journal of Geodesy*, vol. 80, no. 4, pp. 177–183, 2006.

[36] A. Girard, "Reachability of Uncertain Linear Systems Using Zonotopes," in *International Workshop on Hybrid Systems: Computation and Control*. Springer, 2005, pp. 291–305.

[37] A. Shetty and G. Gao, "Predicting State Uncertainty Bounds Using Non-Linear Stochastic Reachability Analysis for Urban GNSS-Based UAS Navigation," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 9, pp. 5952–5961, 2020.

[38] S. Thrun, W. Burgard, and D. Fox, "Probabilistic Robotics (Intelligent Robotics and Autonomous Agents series), ser. Intelligent Robotics and Autonomous Agents," 2005.

[39] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, "Trust Region Policy Optimization," in *International conference on machine learning*. PMLR, 2015, pp. 1889–1897.