IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

Robust Sensitivity-Aware Chance-Constrained MPC for Efficient Handling of Multiple Uncertainty Sources

James Zhu, Thierry Simeon, Marco Cognetti

***Abstract*—Robust motion planning under uncertainty is critical for unlocking real-world robotics applications. This paper introduces SupeR-MPC, a computationally-efficient, sensitivity-aware, chance-constrained optimization framework that systematically accounts for multiple sources of uncertainty, including state estimation error, model parameter uncertainty, obstacle localization error, and process noise. This approach advances sensitivity-aware robust control by integrating chance-constrained optimization to handle the uncertainty models of Kalman-filtering methods. To demonstrate robustness against multiple uncertainty sources, SupeR-MPC was validated on a range of systems and environments, from a simple 2D example to a multi-agent dynamic obstacle avoidance scenario. Comparisons against existing MPC methods show that SupeR-MPC significantly improves constraint satisfaction and robustness while maintaining real-time computational efficiency. These results highlight the effectiveness of sensitivity-aware chance constraints in enhancing real-world robotic decision-making under uncertainty.**

***Index Terms*—Planning under Uncertainty, Robust/Adaptive Control, Optimization and Optimal Control**

I. Introduction

ADVANCEMENTS in mobile robotics have fueled growing interest in deploying robots for real-world applications such as manufacturing [1], logistics [2], and personal assistance [3]. To operate reliably in these environments, robots must handle a range of uncertainties stemming from factors such as imperfect state estimation, model inaccuracies, and unpredictable agents [4,5]. Modern estimation, mapping, and prediction techniques allow robots to model their environments while characterizing the level of uncertainty arising from sensor bias and variance or inaccuracies in the dynamics model. For instance, the Kalman filter [6] provides a mean estimate and a covariance matrix that approximates uncertainty.

This work investigates how to explicitly account for uncertainties arising from Kalman filter-based estimates of state, model parameters, and obstacle localization within a robust planning framework. Kalman-based estimation can be challenging to integrate into robust control methods because its unbounded Gaussian nature prevents any strict guarantees of safety. However, this estimated information often provides the most accurate prior to plan trajectories that are safe but not overly conservative. This work introduces Super-Robust Model-Predictive Control (**SupeR-MPC**), a novel trajectory planning framework that propagates Gaussian uncertainty estimates to solve a chance-constrained robust control problem.

The authors are with the LAAS-CNRS, Université de Toulouse, CNRS, UPS, Toulouse, France. E-mail: {james.zhu, simeon, marco.cognetti}@laas.fr

This work was supported by the project ANR-20-CE33-0003 “CAMP” and by the Chaire de Professeur Junior Grant ANR-22-CPJ1-0064-01.

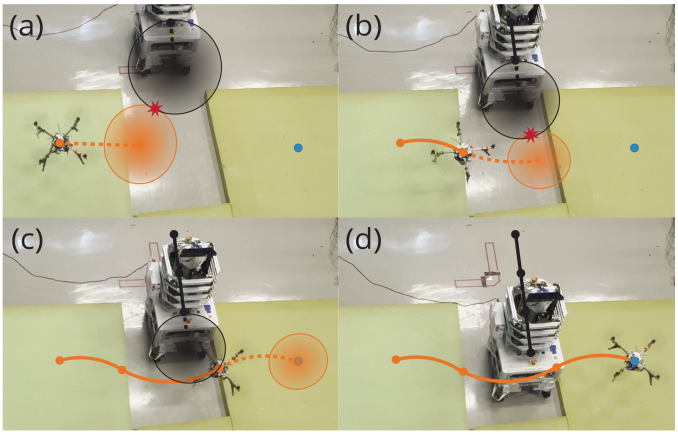

Fig. 1: **SupeR-MPC** enables robust trajectory planning despite uncertainties in the state estimate and mass properties of the ego quadrotor (orange), and the state estimate of the dynamic obstacle (black). The target position is shown in blue. At each iteration, **SupeR-MPC** propagates uncertainty to perform robust collision checks using chance constraints. In (a) and (b), the deterministic MPC trajectory fails these checks (red stars) and **SupeR-MPC** seeks a safer path. When it is safe, as in (c) and (d), it reverts to the optimal deterministic plan. **SupeR-MPC** successfully completed 20/20 trials.

This method draws from estimation data to model uncertainty more accurately compared to approaches that employ predefined uncertainty approximations, while being more effective than sampling-based methods in allowing for real-time onboard deployment on a computationally-limited robot. **SupeR-MPC** provides the following contributions:

1) Draws from real-time estimates of system state, model parameters, process noise, etc., to establish accurate priors of uncertainty.
2) Leverages sensitivity analysis [7] to efficiently propagate initial priors and quantify their effect over time, allowing for dynamic estimation of future uncertainty.
3) Uses a chance-constrained optimization framework [8] to approximate the probability of constraint violations, enabling robots to balance safety with performance.

We present several experiments to validate the utility of this method, including a multi-agent navigation environment where an ego quadrotor must safely avoid a dynamic obstacle under the presence of a mass disturbance and state estimation error of both the ego agent and obstacle, Fig. 1.

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

II. Related Works

In control theory, certain canonical methods have demonstrated robustness to disturbances without requiring an explicit uncertainty model. For example, the PID controller [9] incorporates integral and derivative feedback terms to attenuate low- and high-frequency disturbances, respectively. Model-Predictive Control (MPC) [10] provides robustness by repeatedly solving an optimization problem over a finite horizon, reacting to disturbances in real time. However, neither approach explicitly reasons about uncertainty, limiting their safety in highly uncertain environments.

A more explicit approach to robustness is to incorporate a predefined uncertainty distribution into system dynamics, such as Gaussian noise [11,12], a uniform bounded distribution [13,14], or a discrete set of possible disturbances [15]. While these methods can improve resilience to uncertainty, they assume that the chosen distribution accurately reflects real-world disturbances. In practice, a mismatch between the assumed and actual uncertainty distributions can lead to either insufficient robustness or excessive conservativeness. Instead, adapting uncertainty models in real-time based on sensor measurements and estimation algorithms reduces the need for ad-hoc tuning of uncertainty distributions.

Another challenge for many robust control methods is that different sources of uncertainty affect the robot's future states in different ways, requiring approaches that can reason about them simultaneously. Many works have demonstrated robust planning by propagating uncertainty through system dynamics [7,16,17], but work integrating different uncertainty sources together has been limited. Sampling-based methods [18–21] accurately propagate non-Gaussian distributions through nonlinear dynamics and could be used to propagate any number of uncertainty sources. However, they require a large number of sample trajectories that often leverage GPU-based parallel computing, which may not be feasible for computationally-limited systems like small quadrotors. Additionally, works in adaptive control aim to learn and adapt to unmodeled system dynamics like quadrotor ground effects and wind [22,23]. While adaptive control techniques can improve dynamics models under multiple unmodeled disturbances, such as in [24], they only capture the mean of the uncertainty and not its full distribution.

The proposed Super-Robust MPC (**SupeR-MPC**) method aims to provide a more unified framework for efficiently reasoning about multiple uncertainty sources simultaneously. In particular, we demonstrate its effectiveness in a multi-agent environment, where an ego quadrotor navigates robustly despite uncertainties in state estimation, dynamic obstacle prediction, and model parameters. To the best of our knowledge, this is the first approach that explicitly integrates these diverse uncertainties into a single robust control framework on a computationally-limited system. **SupeR-MPC** maintains robustness without excessive conservativeness and computational demands, effectively bridging gaps in prior robust control approaches.

III. Problem Definition

Consider a discrete dynamical system with state dynamics

$$\boldsymbol{x}_k = \boldsymbol{f}(\boldsymbol{x}_{k-1}, \boldsymbol{u}_{k-1}, \boldsymbol{p}) + \boldsymbol{\omega}_{k-1} \tag{1}$$

where $\boldsymbol{x}_k \in \mathbb{R}^{n_x}$ represents the system state at timestep k, $\boldsymbol{u}_k \in \mathbb{R}^{n_u}$ is the corresponding control input, and $\boldsymbol{p} \in \mathbb{R}^{n_p}$ is a vector of uncertain model parameters assumed to be constant. The process noise $\boldsymbol{\omega}_k$ follows a multivariate Gaussian distribution, $\mathcal{N}(\mathbf{0}, \hat{\boldsymbol{\Sigma}}^Q)$, with zero mean and covariance $\hat{\boldsymbol{\Sigma}}^Q$.

We assume the system is equipped with a Kalman-type observer, such as an unscented Kalman filter (UKF) [25], which estimates both the state $\hat{\boldsymbol{x}}_k \in \mathbb{R}^{n_x}$ and the model parameters $\hat{\boldsymbol{p}}_k \in \mathbb{R}^{n_p}$, along with their associated covariances, $\hat{\boldsymbol{\Sigma}}^x_k \in \mathbb{R}^{n_x \times n_x}$ and $\hat{\boldsymbol{\Sigma}}^p_k \in \mathbb{R}^{n_p \times n_p}$, respectively.

Given a reference state $\tilde{\boldsymbol{x}}_k$ and feedforward input $\tilde{\boldsymbol{u}}_k$, define a tracking controller that acts on the estimated state:

$$\boldsymbol{u}_k = \tilde{\boldsymbol{u}}_k + \boldsymbol{g}(\hat{\boldsymbol{x}}_k, \tilde{\boldsymbol{x}}_k) \tag{2}$$

For a finite time horizon of N timesteps and initial estimates $\hat{\boldsymbol{x}}_0$ and $\hat{\boldsymbol{p}}_0$, the optimal control problem considered in this work is the following:

$$\min_{\tilde{\boldsymbol{x}}_{0:N}, \tilde{\boldsymbol{u}}_{0:N-1}} \quad J = \ell_N(\tilde{\boldsymbol{x}}_N) + \sum_{k=0}^{N-1} \ell_k(\tilde{\boldsymbol{x}}_k, \tilde{\boldsymbol{u}}_k) \tag{3}$$

$$\text{where} \quad \tilde{\boldsymbol{x}}_0 = \hat{\boldsymbol{x}}_0 \tag{4}$$

$$\tilde{\boldsymbol{x}}_k = \boldsymbol{f}(\tilde{\boldsymbol{x}}_{k-1}, \tilde{\boldsymbol{u}}_{k-1}, \hat{\boldsymbol{p}}_0) \quad \forall k \in (0, N] \tag{5}$$

$$\Pr(\boldsymbol{x_k} \notin \mathcal{C}^x_k) \geq 1 - \delta_x \quad \forall k \in [0, N] \tag{6}$$

$$\Pr(\boldsymbol{u_k} \notin \mathcal{C}^u_k) \geq 1 - \delta_u \quad \forall k \in [0, N) \tag{7}$$

where J is the cost function to be minimized and ℓ_N and ℓ_k are the nonlinear terminal and stage costs, respectively. (4) and (5) are the constraints on the initial condition and dynamics, utilizing the state and parameter estimates available at the initial timestep. (6) and (7) are the chance constraints for collision and input saturation, respectively, and are described in detail in Sec. V. As will be discussed, deterministic constraints $\boldsymbol{x_k} \notin \mathcal{C}^x_k$ and $\boldsymbol{u_k} \notin \mathcal{C}^u_k$ can be recovered when $\delta_x = \delta_u = 0.5$. The notation $\tilde{\boldsymbol{x}}_{0:N}$ and $\tilde{\boldsymbol{u}}_{0:N-1}$ indicates that the solution to the optimal control problem generates the reference trajectory and feedforward control inputs to be tracked. As the estimates are updated, MPC will repeatedly solve the optimal control problem.

IV. Sensitivity Analysis

This work aims to analyze how uncertainties in modeling and estimation influence the state and input uncertainty of a planned trajectory. Specifically, we consider three common sources of uncertainty: 1) state estimation errors, 2) parameter variations, and 3) process noise in the dynamics model. We rely on a Gaussian approximation of uncertainties and linearization of the dynamics, which sacrifices some accuracy in favor of computational tractability. We believe this choice is not overly harmful for many real-world applications, as Gaussian-based Kalman-filtering estimation and linearized feedback control are widespread in the field.

Denote $\hat{\boldsymbol{x}}_0$ and $\hat{\boldsymbol{p}}_0$ as the state and parameter estimates provided to the MPC solver at the initialization of an execution

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

to solve (3). $\hat{\boldsymbol{\Sigma}}_0^x$ and $\hat{\boldsymbol{\Sigma}}_0^p$ represent the respective covariance matrices of the estimates. Over a time horizon N, we aim to compute the *trajectory uncertainties* of the state and inputs, $\tilde{\boldsymbol{\Sigma}}_{0:N}^x$ and $\tilde{\boldsymbol{\Sigma}}_{0:N}^u$, which represent the propagation of initial uncertainties over the trajectory. Given a feedforward sequence of inputs $\tilde{\boldsymbol{u}}_{0:N-1}$, define the *reference trajectory* $\tilde{\boldsymbol{x}}_{0:N}$ as the rollout of $\hat{\boldsymbol{x}}_0$ based on $\tilde{\boldsymbol{u}}_{0:N-1}$ and $\hat{\boldsymbol{p}}_0$, such that for any integer $k \in (0, N]$,

$$\tilde{\boldsymbol{x}}_k = \boldsymbol{f}(\tilde{\boldsymbol{x}}_{k-1}, \tilde{\boldsymbol{u}}_{k-1}, \hat{\boldsymbol{p}}_0) \tag{8}$$

where $\tilde{\boldsymbol{x}}_0 = \hat{\boldsymbol{x}}_0$. Note that the effect of future measurements are not considered in the planning stage, so we assume the estimates read at the beginning of the MPC execution are used for the entire time horizon.

In general, $\hat{\boldsymbol{p}}_0$ is an imperfect estimate of the true parameter vector $\boldsymbol{p}$:

$$\boldsymbol{p} = \hat{\boldsymbol{p}}_0 + \delta \boldsymbol{p}_0 \tag{9}$$

Given the true value of $\boldsymbol{p}$, the *estimate trajectory* dynamics are a function of the closed-loop feedback input $\hat{\boldsymbol{u}}_{k-1} = \boldsymbol{g}(\hat{\boldsymbol{x}}_{k-1}, \tilde{\boldsymbol{x}}_{k-1})$:

$$\hat{\boldsymbol{x}}_k = \boldsymbol{f}(\hat{\boldsymbol{x}}_{k-1}, \tilde{\boldsymbol{u}}_{k-1} + \boldsymbol{g}(\hat{\boldsymbol{x}}_{k-1}, \tilde{\boldsymbol{x}}_{k-1}), \hat{\boldsymbol{p}}_0 + \delta \boldsymbol{p}_0) \tag{10}$$

In the case where $\delta \boldsymbol{p}_0 = 0$, $\hat{\boldsymbol{x}}_k = \tilde{\boldsymbol{x}}_k$ for all $k \geq 0$.

If there is also error in the initial state estimation, the true initial state $\boldsymbol{x}_0$ can be expressed as:

$$\boldsymbol{x}_0 = \hat{\boldsymbol{x}}_0 + \delta \boldsymbol{x}_0 \tag{11}$$

The *true trajectory* then has dynamics that evolve according to the true initial state and true parameters:

$$\boldsymbol{x}_k = \boldsymbol{f}(\hat{\boldsymbol{x}}_{k-1} + \delta \boldsymbol{x}_{k-1}, \tilde{\boldsymbol{u}}_{k-1} + \boldsymbol{g}(\hat{\boldsymbol{x}}_{k-1}, \tilde{\boldsymbol{x}}_{k-1}), \hat{\boldsymbol{p}}_0 + \delta \boldsymbol{p}_0) \tag{12}$$

Because the true state is unmeasured and unknown, it is impossible to perform feedback control on the true state. Thus, the feedback actions in (12) and (10) are equivalent:

$$\boldsymbol{u}_{k-1} = \hat{\boldsymbol{u}}_{k-1} = \tilde{\boldsymbol{u}}_{k-1} + \boldsymbol{g}(\hat{\boldsymbol{x}}_{k-1}, \tilde{\boldsymbol{x}}_{k-1}) \tag{13}$$

In order to safely control $\boldsymbol{x}_k$ at every timestep, we aim to understand the sensitivity of $\boldsymbol{x}_k$ with respect to initial variations resulting from state and parameter estimate error:

$$\boldsymbol{\Psi}_k = \frac{\partial \boldsymbol{x}_k}{\partial \boldsymbol{x}_0} \tag{14}$$

$$\boldsymbol{\Pi}_k = \frac{\partial \boldsymbol{x}_k}{\partial \boldsymbol{p}_0} \tag{15}$$

We can approach deriving these matrices with a chain rule expansion. First, regarding (14), note that the dynamics law in (12) does not contain any dependency on $\delta \boldsymbol{x}_{k-1}$ within the terms $\tilde{\boldsymbol{u}}_{k-1} + \boldsymbol{g}(\hat{\boldsymbol{x}}_{k-1}, \tilde{\boldsymbol{x}}_{k-1})$ and $\hat{\boldsymbol{p}}_0 + \delta \boldsymbol{p}_0$. As a result,

$$\frac{\partial \boldsymbol{x}_k}{\partial \boldsymbol{x}_{k-1}} = \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}_{k-1}}\Big|_{(\boldsymbol{x}_{k-1}, \boldsymbol{u}_{k-1}, \boldsymbol{p}) = (\tilde{\boldsymbol{x}}_{k-1}, \tilde{\boldsymbol{u}}_{k-1}, \hat{\boldsymbol{p}}_0)} \tag{16}$$

where the linearization is taken around the zero-variation reference state, input, and parameters $(\tilde{\boldsymbol{x}}_{k-1}, \tilde{\boldsymbol{u}}_{k-1}, \hat{\boldsymbol{p}}_0)$. All derivatives in this section will be linearized around this point, so we will drop the notation for readability.

$\boldsymbol{\Psi_k}$ can then be defined recursively as:

$$\boldsymbol{\Psi}_k = \frac{\partial \boldsymbol{x}_k}{\partial \boldsymbol{x}_0} = \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}_{k-1}} \boldsymbol{\Psi}_{k-1}, \quad \boldsymbol{\Psi}_0 = \mathbf{I} \tag{17}$$

where the initial condition stems from the fact that $\frac{\partial \boldsymbol{x}_0}{\partial \boldsymbol{x}_0} = \boldsymbol{I}$.

Now considering (15), the chain rule expansion is slightly more complex. Consider that

$$\boldsymbol{\Pi}_k = \frac{\partial \boldsymbol{x}_k}{\partial \boldsymbol{p}_0} = \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}_{k-1}} \frac{\partial \boldsymbol{x}_{k-1}}{\partial \boldsymbol{p}_0} + \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{u}_{k-1}} \frac{\partial \boldsymbol{u}_{k-1}}{\partial \boldsymbol{p}_0} + \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{p}_0} \tag{18}$$

where the input sensitivity $\boldsymbol{\Theta}_{k-1} = \frac{\partial \boldsymbol{u}_{k-1}}{\partial \boldsymbol{p}_0}$ has its own chain rule expansion:

$$\boldsymbol{\Theta}_{k-1} = \frac{\partial \boldsymbol{u}_{k-1}}{\partial \boldsymbol{p}_0} = \frac{\partial \boldsymbol{g}}{\partial \boldsymbol{x}_{k-1}} \frac{\partial \boldsymbol{x}_{k-1}}{\partial \boldsymbol{p}_0} \tag{19}$$

Note that under linear feedback control, $\partial \boldsymbol{g} / \partial \boldsymbol{x}_{k-1}$ is exactly the feedback gain matrix $\boldsymbol{K}_{k-1}$ that is used during execution for feedback control of the tracking error.

Some simple algebra yields another recursive equation:

$$\begin{aligned} \boldsymbol{\Pi}_k &= \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}_{k-1}} \boldsymbol{\Pi}_{k-1} + \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{u}_{k-1}} \boldsymbol{\Theta}_{k-1} + \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{p}_0}, \quad \boldsymbol{\Pi}_0 = \mathbf{0} \\ \boldsymbol{\Theta}_k &= \frac{\partial \boldsymbol{g}}{\partial \boldsymbol{x}_k} \boldsymbol{\Pi}_k \end{aligned} \tag{20}$$

The initial condition in this case is derived from $\frac{\partial \boldsymbol{x}_0}{\partial \boldsymbol{p}_0} = \mathbf{0}$. This derivation of the state and input sensitivities with respect to parameter variation mirrors prior work in sensitivity analysis, which has been leveraged to design robust controllers under parametric uncertainty [13,16]. However, to the authors' knowledge, this is the first presentation of state sensitivity with respect to state estimation error.

Finally, process noise uncertainty does not require sensitivity analysis because process noise is modeled as an additive term to the dynamics. At timestep k, the uncertainty in state contributed by process noise is $\hat{\boldsymbol{\Sigma}}_k^Q = k \hat{\boldsymbol{\Sigma}}^Q$.

Now we can compute the trajectory uncertainties $\tilde{\boldsymbol{\Sigma}}_{0:N}^x$ and $\tilde{\boldsymbol{\Sigma}}_{0:N}^u$ as the sum of covariance propagations:

$$\tilde{\boldsymbol{\Sigma}}_k^x = \boldsymbol{\Psi}_k \hat{\boldsymbol{\Sigma}}_0^x \boldsymbol{\Psi}_k^T + \boldsymbol{\Pi}_k \hat{\boldsymbol{\Sigma}}_0^p \boldsymbol{\Pi}_k^T + k \hat{\boldsymbol{\Sigma}}^Q \tag{21}$$

$$\tilde{\boldsymbol{\Sigma}}_k^u = \boldsymbol{\Theta}_k \hat{\boldsymbol{\Sigma}}_0^p \boldsymbol{\Theta}_k^T \tag{22}$$

V. Chance Constraints

Chance-constrained optimization is a well-studied approach in the literature and has been applied to trajectory planning under state estimation errors, dynamic obstacle uncertainty, and more [8,11,26–29]. This work primarily builds on the formulation presented in [8] due to its balance of accuracy and computational simplicity.

In general, a chance constraint imposes a probabilistic bound on constraint satisfaction by ensuring that the random variable $\boldsymbol{x} \sim \mathcal{N}(\hat{\boldsymbol{x}}, \hat{\boldsymbol{\Sigma}})$ lies outside a constraint region $\mathcal{C}$ with at least a specified probability:

$$\Pr(\boldsymbol{x} \notin \mathcal{C}) \geq 1 - \delta \tag{23}$$

where $0 < \delta \leq 1$ is the probability threshold parameter. Note that each chance constraint can be assigned an independently

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

chosen threshold parameter, though throughout this section we will use the generic variable δ.

In this work, we consider two classes of chance constraints. The first is a linear chance constraint, defined as $\Pr(\boldsymbol{a}^T\boldsymbol{x} \geq b) \geq 1-\delta$ where $\boldsymbol{a}$ has the same dimension as $\boldsymbol{x}$ and b is a scalar. The corresponding constraint region is

$$\mathcal{C}^\ell = \{\boldsymbol{x} | \hat{\boldsymbol{a}}^T\boldsymbol{x} < b\} \tag{24}$$

A common approach to handling chance constraints is to convert them into deterministic constraints [30]. Following the derivation in [8], the deterministic equivalent for this class of constraints is

$$\boldsymbol{a}^T\hat{\boldsymbol{x}} - b \geq \text{erf}^{-1}(1-2\delta)\sqrt{2\boldsymbol{a}^T\hat{\boldsymbol{\Sigma}}\boldsymbol{a}} \tag{25}$$

where $\text{erf}(x) = \frac{2}{\sqrt{\pi}}\int_0^x e^{-t^2}dt$ is the standard error function. In the limit case as $\delta \to 0$, $\text{erf}^{-1}(1-2\delta) \to \infty$ due to the unboundedness of the Gaussian distribution. Conversely, as $\delta \to 0.5$, $\text{erf}^{-1}(1-2\delta) \to 0$, collapsing into a standard deterministic constraint on the mean. In robotics applications, linear chance constraints can model input limits and linear obstacles such as walls.

The second class of chance constraints considered in this work models collision avoidance between two circular or spherical bodies. Let the Cartesian position vectors of the two bodies be $\boldsymbol{c}_i \sim \mathcal{N}(\hat{\boldsymbol{c}}_i, \hat{\boldsymbol{\Sigma}}_i)$ and $\boldsymbol{c}_j \sim \mathcal{N}(\hat{\boldsymbol{c}}_j, \hat{\boldsymbol{\Sigma}}_j)$ with respective radii r_1 and r_2. Approximating the collision region as a half-space, define

$$\mathcal{C}^{ij} = \{\boldsymbol{x}_i | \hat{\boldsymbol{a}}^T(\boldsymbol{c}_i - \boldsymbol{c}_j) < r\} \tag{26}$$

where $\hat{\boldsymbol{a}} = (\hat{\boldsymbol{c}}_i - \hat{\boldsymbol{c}}_j)/||\hat{\boldsymbol{c}}_i - \hat{\boldsymbol{c}}_j||$ and $r = r_i + r_j$.

The deterministic formulation of the chance constraint is

$$\hat{\boldsymbol{a}}^T(\hat{\boldsymbol{c}}_i - \hat{\boldsymbol{c}}_j) - r \geq \text{erf}^{-1}(1-2\delta)\sqrt{2\hat{\boldsymbol{a}}^T(\hat{\boldsymbol{\Sigma}}_i + \hat{\boldsymbol{\Sigma}}_j)\hat{\boldsymbol{a}}} \tag{27}$$

VI. Super-Robust MPC

In this section, we introduce a novel framework for robust trajectory planning under multiple sources of probabilistic uncertainty called Super-Robust MPC, or **SupeR-MPC**. Our approach extends the safe planning capabilities of Sensitivity-Aware Tube MPC [14], which handles bounded perturbations in model parameters. However, a bounded parameter distribution is not compatible with Gaussian estimates that are provided by Kalman-filter methods, limiting the utility of [14] in real-world applications. Thus, we improve upon [14] by integrating sensitivity analysis with chance-constrained optimization, enabling robustness not only to model parameter variations but also to general classes of Gaussian uncertainties, such as state estimation errors, dynamic obstacle localization, and process noise.

Broadly speaking, MPC works by constantly replanning trajectories over the course of a deployment. As new measurements are processed by the state estimators, MPC can rapidly adapt its plan to handle unforeseen disturbances [10]. **SupeR-MPC** makes use of this structure to solve the sensitivity-aware, chance-constrained trajectory planning problem while maintaining real-time computational performance with strategies like warm-starting. Algorithm 1 outlines the structure of the **SupeR-MPC** method, which is further detailed below. For readability, trajectory-wise subscripts such as $0:N$ and $0:N-1$ are omitted in Algorithm 1 but are included in the subsequent discussion.

Algorithm 1 SupeR-MPC

$\hat{\boldsymbol{x}}_0, \hat{\boldsymbol{p}}_0, \hat{\boldsymbol{\Sigma}}_0^x, \hat{\boldsymbol{\Sigma}}_0^p \leftarrow$ UpdateEstimate()
$\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{u}}, \tilde{\boldsymbol{\lambda}}, \tilde{\boldsymbol{\mu}} \leftarrow$ SolveDeterministic($\hat{\boldsymbol{x}}_0$,$\hat{\boldsymbol{p}}_0$)
$\tilde{\boldsymbol{K}}^x \leftarrow$ ComputeGains($\tilde{\boldsymbol{x}}$, $\tilde{\boldsymbol{u}}$, $\tilde{\boldsymbol{\lambda}}$, $\tilde{\boldsymbol{\mu}}$)
$\tilde{\boldsymbol{\Sigma}}^x, \tilde{\boldsymbol{\Sigma}}^u \leftarrow$ ComputeSensitivities($\tilde{\boldsymbol{x}}$, $\tilde{\boldsymbol{u}}$, $\hat{\boldsymbol{\Sigma}}_0^x$, $\hat{\boldsymbol{\Sigma}}_0^p$, $\tilde{\boldsymbol{K}}^x$)
$\boldsymbol{W} \leftarrow$ ComputeDerivatives($\tilde{\boldsymbol{x}}$, $\tilde{\boldsymbol{u}}$, $\tilde{\boldsymbol{\Sigma}}^x$, $\tilde{\boldsymbol{\Sigma}}^u$)
while *running* **do**
  $\hat{\boldsymbol{x}}_0, \hat{\boldsymbol{p}}_0, \hat{\boldsymbol{\Sigma}}_0^x, \hat{\boldsymbol{\Sigma}}_0^p \leftarrow$ UpdateEstimate()
  $\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{u}}, \tilde{\boldsymbol{\lambda}}, \tilde{\boldsymbol{\mu}} \leftarrow$ Solve($\hat{\boldsymbol{x}}_0$, $\hat{\boldsymbol{p}}_0$, $\boldsymbol{W}$)
  SendCommand($\tilde{\boldsymbol{u}}_0$)
  $\tilde{\boldsymbol{x}}^*, \tilde{\boldsymbol{u}}^*, \tilde{\boldsymbol{\lambda}}^*, \tilde{\boldsymbol{\mu}}^* \leftarrow$ ShiftSolution($\tilde{\boldsymbol{x}}$, $\tilde{\boldsymbol{u}}$, $\tilde{\boldsymbol{\lambda}}$, $\tilde{\boldsymbol{\mu}}$)
  $\tilde{\boldsymbol{K}}^x \leftarrow$ ComputeGains($\tilde{\boldsymbol{x}}^*$, $\tilde{\boldsymbol{u}}^*$, $\tilde{\boldsymbol{\lambda}}^*$, $\tilde{\boldsymbol{\mu}}^*$)
  $\tilde{\boldsymbol{\Sigma}}^x, \tilde{\boldsymbol{\Sigma}}^u \leftarrow$ ComputeSensitivities($\tilde{\boldsymbol{x}}^*$, $\tilde{\boldsymbol{u}}^*$, $\hat{\boldsymbol{\Sigma}}_0^x$, $\hat{\boldsymbol{\Sigma}}_0^p$, $\tilde{\boldsymbol{K}}^x$)
  $\boldsymbol{W} \leftarrow$ ComputeDerivatives($\tilde{\boldsymbol{x}}^*$, $\tilde{\boldsymbol{u}}^*$, $\tilde{\boldsymbol{\Sigma}}^x$, $\tilde{\boldsymbol{\Sigma}}^u$)

Prior to task execution, the algorithm initializes by generating a warm-start trajectory to feed to the online MPC. Warm-starting is a common strategy to make MPC more efficient and reliable by leveraging the fact that successive trajectory plans are typically similar, apart from a time shift [31]. By warm-starting each MPC execution with the previous solution shifted forward, solver convergence can be significantly accelerated. Many MPC optimization packages natively incorporate the warm-starting method, such as Prox-QP [32], which is utilized in this work.

The initial warm-start procedure begins with the function UpdateEstimate retrieving mean and covariance estimates for the state and parameters from the estimator. Then, SolveDeterministic solves (3) deterministically, disregarding sensitivity analysis. As discussed in Sec. V, this is equivalent to setting $\delta_x = \delta_u = 0.5$. This step produces a nominal trajectory of states $\tilde{\boldsymbol{x}}_{0:N}$, inputs $\tilde{\boldsymbol{u}}_{0:N-1}$, primal variables $\tilde{\boldsymbol{\lambda}}_{0:N}$, and dual variables $\tilde{\boldsymbol{\mu}}_{0:N}$.

Next, the algorithm computes trajectory uncertainties (21) and (22). As mentioned in Sec. IV, in order to accomplish this, MPC must first estimate the feedback gains $\tilde{\boldsymbol{K}}^x_{0:N-1}$, equivalent to $\partial \boldsymbol{g}/\partial \boldsymbol{x}_{k-1}$ in (19). These can be efficiently solved by leveraging the primal-dual structure of modern optimization solvers. A solver like Prox-QP ensures optimality by operating on both primal variables $\tilde{\boldsymbol{\lambda}}$ and dual variables $\tilde{\boldsymbol{\mu}}$, which can also be utilized to compute future feedback gains, as derived in [14]. This procedure is performed by ComputeGains which returns $\tilde{\boldsymbol{K}}^x_{0:N-1}$.

Then, ComputeSensitivities calculates $\tilde{\boldsymbol{\Sigma}}^x_{0:N}$ and $\tilde{\boldsymbol{\Sigma}}^u_{0:N-1}$ according to (21) and (22), based on the derivation in Sec. IV. Finally, ComputeDerivatives seeds the solver with the proper search directions to efficiently find an optimal solution. In this work, the derivatives were computed with the autodiff library [33]. These derivatives, along with the trajectory variables $\tilde{\boldsymbol{x}}_{0:N}$, $\tilde{\boldsymbol{u}}_{0:N-1}$, $\tilde{\boldsymbol{\lambda}}_{0:N}$, $\tilde{\boldsymbol{\mu}}_{0:N}$, $\tilde{\boldsymbol{\Sigma}}^x_{0:N}$, and $\tilde{\boldsymbol{\Sigma}}^u_{0:N-1}$ are stored in the data structure $\boldsymbol{W}$ for warm-starting.

Once this initialization process is complete, the robot begins its navigation task. At each iteration of the MPC loop, updated state and parameter estimates are read, and the robust chance-

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

constrained optimization problem in (3) is solved by incorporating the previously computed sensitivity covariances, $\tilde{\boldsymbol{\Sigma}}^{x}_{0:N}$ and $\tilde{\boldsymbol{\Sigma}}^{u}_{0:N-1}$, into (25) and (27). Once the solver converges to a robust solution, the computed control input is applied to the robot via SENDCOMMAND. Following this, the warm-start process is reinitialized. SHIFTSOLUTION shifts the current solution forward by one timestep. Formally, $\tilde{\boldsymbol{x}}^{*} = [\tilde{\boldsymbol{x}}^{*}_{0:N-1}, \tilde{\boldsymbol{x}}^{*}_{N}]$, with $\tilde{\boldsymbol{x}}^{*}_{0:N-1} = \tilde{\boldsymbol{x}}_{1:N}$ and the final value $\tilde{\boldsymbol{x}}^{*}_{N}$ is estimated by simulating one additional timestep forward. Similar shifts are performed for $\tilde{\boldsymbol{u}}$, $\tilde{\boldsymbol{\lambda}}$, and $\tilde{\boldsymbol{\mu}}$, obtaining $\tilde{\boldsymbol{u}}^{*}$, $\tilde{\boldsymbol{\lambda}}^{*}$, and $\tilde{\boldsymbol{\mu}}^{*}$, respectively. The warm-start procedure is completed as before, invoking COMPUTEGAINS, COMPUTESENSITIVITIES, and COMPUTEDERIVATIVES to store the data structure $\boldsymbol{W}$.

VII. EXPERIMENTS & RESULTS

The proposed method was validated on two types of systems. First, a simple 2D toy example illustrates how the sensitivity-aware chance-constrained approach improves robustness to multiple sources of uncertainty. Next, **SupeR-MPC** is deployed on a quadrotor system, demonstrating the improved performance of **SupeR-MPC** in practical robotic deployments compared to state-of-the-art baselines.

A. 2D Toy Example

To illustrate the sensitivity-aware chance-constrained approach, we present a simple 2D open-loop kinematic system with the following dynamics:

$$\begin{bmatrix}\dot{x}\\ \dot{y}\end{bmatrix} = \begin{bmatrix}0.1 & 0\\ 0 & 0.2\end{bmatrix}\begin{bmatrix}x\\ y\end{bmatrix} + \begin{bmatrix}1 & 0.1+p_1\\ 0.1 & 1+p_2\end{bmatrix}\begin{bmatrix}u_1\\ u_2\end{bmatrix} \tag{28}$$

A trajectory optimization problem is formulated to represent a single execution of the MPC process. Because the persistent replanning of MPC provides a layer of robustness on top of the sensitivity framework, this example isolates the contributions of the sensitivity-aware chance constraints.

The system starts at an initial state of $\boldsymbol{x}_0 = [0, 0.1]^T$ and is tasked with reaching a target state of $\boldsymbol{x}_T = [1, 0.1]^T$. The y-coordinate is constrained to be $0 \leq y \leq 0.2$. Additionally, a circular obstacle is placed at $\boldsymbol{x}^c = [0.5, 0.2]^T$ with radius $r^c = 0.175$. The default parameters are $[p_1, p_2] = [0.1, 0.25]$.

The optimal control problem follows the structure of (3) – (7). In particular, the objective function (3) is given as

$$J = (\tilde{\boldsymbol{x}}_N - \boldsymbol{x}_T)^T \boldsymbol{Q}_T (\tilde{\boldsymbol{x}}_N - \boldsymbol{x}_T) + \sum_{k=0}^{N-1} (\tilde{\boldsymbol{u}}_k^T \boldsymbol{R}_k \tilde{\boldsymbol{u}}_k) \tag{29}$$

The dynamics constraints in (5) are enforced with the discretization of (28). The two collision chance constraints for the y-position are transcribed following (25) where, for instance, the constraint $y \leq 0.2$ maps to $\hat{\boldsymbol{a}} = [0, 1]^T$ and $b = 0.2$. The collision chance constraint for the circular obstacle follows (27), as defined in Sec. V. Input limits (7) are enforced as linear chance constraints (25) with $-6 \leq \boldsymbol{u}_k \leq 6$. While each constraint could be assigned an independent threshold parameter, we use a constant parameter δ for all collision and input constraints. The optimization is executed over a time horizon of $N = 20$ with a timestep interval of $\Delta t = 0.01$s. The weighting matrices are set as $\boldsymbol{Q}_T = 500\boldsymbol{I}$, $\boldsymbol{R}_k = 5e^{-3}\boldsymbol{I}$.

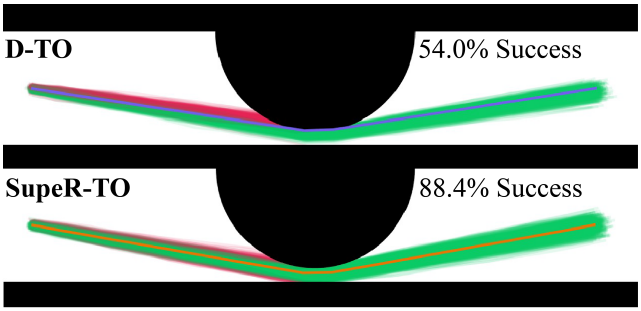

Fig. 2: The robust obstacle avoidance task is shown comparing **D-TO** (top) with the proposed **SupeR-TO** at a $\delta = 0.1$ threshold (bottom). Green trajectories successfully avoided the obstacle, while red trajectories experienced a collision. Nominal planned trajectories are shown in purple (for **D-TO**) and orange (for **SupeR-TO**), respectively.

In this experiment, we compared the sensitivity-aware chance-constrained **SupeR-TO** with a standard deterministic trajectory optimization (**D-TO**) formulation, which only ensures constraint satisfaction of the mean estimate.

To assess robustness, uncertainty in initial state estimation, model parameters, and process noise were introduced using the following covariance estimates, selected to create a challenging yet feasible scenario: $\hat{\boldsymbol{\Sigma}}^{x}_0 = 2e^{-5}\boldsymbol{I}$, $\hat{\boldsymbol{\Sigma}}^{p}_0 = 2e^{-3}\boldsymbol{I}$, $\hat{\boldsymbol{\Sigma}}^{Q} = 2e^{-6}\boldsymbol{I}$. For each formulation, 1000 randomly perturbed trajectories were simulated, with initial state errors, parameter variations, and process noise sampled from zero-mean distributions with these covariances.

The **D-TO** trajectory achieves cost minimization and nominally satisfies constraints in the deterministic setting. However, nearly half of the perturbed trajectories fail, primarily due to collisions with the obstacle. Evaluating **SupeR-TO** requires tuning the threshold parameter δ, where lower δ values yield more robust trajectories but make finding feasible solutions harder. We found $\delta = 0.048$ to be the lowest feasible value, achieving a 90.4% success rate in simulations. However, a change in uncertainty distributions can cause this parameter value to produce infeasible chance constraints, so it is more practical to choose a less strict value of δ. Here, we chose $\delta = 0.1$, which successfully navigated the environment 88.4% of the time, a significant improvement from **D-TO** and nearly the rate of the empirically optimal $\delta = 0.048$ value. The comparison of **D-TO** and **SupeR-TO** with $\delta = 0.1$ is illustrated in Fig. 2.

This example highlights the trade-off between safety and performance, where environments with high levels of uncertainty can not always be safely navigated at strict probability thresholds. Proper values of δ are highly system-dependent, and future work could draw from the field of risk-sensitive control [34,35] to adaptively modify the threshold parameter.

B. Quadrotor

Following [14], define a 3D quadrotor model with a geometric center position $\boldsymbol{c} = [x, y, z]^T$ and velocity $\boldsymbol{v} = [\dot{x}, \dot{y}, \dot{z}]^T$ defined with respect to the world inertial frame.

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

Fig. 3: Image of the quadrotor deployed in experiments. In hardware tests, a mass perturbation was applied at the end of one of the quadrotor arms, shown in the blue ellipse.

TABLE I: MPC Parameters for Quadrotor Experiments

Parameter	Value	
	Simulation	Hardware
Freq. (Hz)	100 Hz	50 Hz
N	20	8
Δt (s)	0.025	0.05
δ	0.02	0.02
$\boldsymbol{Q}_c$	$18\boldsymbol{I}$	$10\boldsymbol{I}$
$\boldsymbol{Q}_v$	$0.6\boldsymbol{I}$	$0.6\boldsymbol{I}$
$\boldsymbol{Q}_q$	$diag([0.01, 0.01, 0.5])$	$diag([0.01, 0.01, 0.5])$
$\boldsymbol{Q}_\omega$	$0.01\boldsymbol{I}$	$0.025\boldsymbol{I}$
$\boldsymbol{R}_k$	$\boldsymbol{I}$	$\boldsymbol{I}$

$\boldsymbol{q} = [q_w, q_x, q_y, q_z]^T$ is the unit quaternion representing the orientation of the body, and $\boldsymbol{\omega} = [\omega_x, \omega_y, \omega_z]^T$ are the angular velocities. This gives a 13-dimensional state vector $\boldsymbol{x} = [\boldsymbol{c}, \boldsymbol{v}, \boldsymbol{q}, \boldsymbol{\omega}]^T$. The dynamics of the system are defined in [14], but are omitted here for brevity. We consider the uncertain variables $\boldsymbol{p} = [m, x_m, y_m, z_m]^T$ representing the total mass and 3D center-of-mass deviation.

The quadrotor utilized in these experiments was custom-built based on the MikroKopter platform (Fig. 3) with a nominal total mass of 1.088 kg and nominal center-of-mass co-located with the geometric center. The quadrotor bounds are approximated as a sphere with radius 0.35 m.

SupeR-MPC iteratively solves the optimization problem in (3) – (7). The cost function to be optimized in this scenario is (29), where $\boldsymbol{Q}_T$ is a block diagonal matrix consisting of blocks $\boldsymbol{Q}_c$, $\boldsymbol{Q}_v$, $\boldsymbol{Q}_q$, and $\boldsymbol{Q}_\omega$, whose values are reported in Tab. I. For all of the following experiments, collision chance constraints with walls and obstacles are defined following (27) and input limit chance constraints follow (25), with $960 \leq \boldsymbol{u}_k \leq 6000$ constraining propeller speed in rpm.

1) Simulation Experiments: In simulation, the quadrotor was tasked with tracking an aggressive "racetrack" trajectory starting at position $[0, 0, 1]^T$ and each second passing through target waypoints at $[0, 2, 1]^T$, $[2, 2, 1]^T$, and $[2, 0, 1]^T$ before coming to rest at its initial position. The environment consists of two walls centered at the origin, an inner wall with a radius of 0.55 m and an outer wall with radius 1.65 m, see Fig. 4.

While the positions of the walls were assumed to be perfectly known, online state estimation was performed with an unscented Kalman filter [25] with a simulated measurement noise with covariance $5e^{-3}\boldsymbol{I}$. The parameter uncertainty was not estimated online, but provided with covariance $5e^{-4}\boldsymbol{I}$. Future work will also incorporate online estimation of parameters. MPC parameters are presented in Tab. I.

Three MPC algorithms were compared in this experiment. The first is Deterministic MPC (**D-MPC**) that does not consider any uncertainty. Next is a Parameter-Sensitive MPC (**PS-MPC**) that considers only parametric uncertainty, treating $\tilde{\boldsymbol{\Sigma}}_k^x = \boldsymbol{\Pi}_k \hat{\boldsymbol{\Sigma}}_0^p \boldsymbol{\Pi}_k^T$ in lieu of (21). This is the probabilistic adaptation of [14]. We believe **PS-MPC** is an appropriate baseline because it is also able to leverage the probabilistic Gaussian parameter estimates. Other methods such as scenario-based MPC [15] may struggle to handle Gaussian uncertainties and can be computationally demanding as multiple optimization problems must be solved across a set of disturbances. As a result, we did not consider these methods for our experiments. Finally, the proposed **SupeR-MPC** considers multiple sources of uncertainty. Several values of δ were evaluated for **PS-MPC** and **SupeR-MPC**, with $\delta = 0.02$ empirically determined to perform well.

Each of the three methods was evaluated on 10 sets of randomly sampled model parameters, representing variations in mass and center-of-mass. For each parameter set, 10 trials were simulated with random measurement noise, resulting in 100 total trials per method.

Figure 4 shows the trajectory of the geometric center of the quadrotor for each trial. Without accounting for any uncertainties, **D-MPC** is ineffective at navigating this environment, successfully finishing only 52% of the time. While **PS-MPC** shows an improvement in performance with a 70% success rate, **SupeR-MPC** is definitively superior, achieving a 93% success rate. This comparison highlights the need for robust algorithms capable of integrating multiple sources of uncertainty in order to navigate realistic environments. Our implementation of **SupeR-MPC** ran at 100 Hz on an Intel Core i7 CPU, with an average iteration time of 8.48 ms. The sensitivity propagation required on average 4.21 ms. This makes up a significant portion of the computation, but is still easily fast enough for real-time control at 100 Hz.

2) Hardware Experiments: Real-world experiments were conducted using a physical quadrotor, shown in Fig. 3. An unmodeled 46 g mass was attached to the end of one of the quadrotor arms by a rope approximately 0.12 m in length and approximately 0.23 m distally from the robot's geometric center. The parameter covariance was set to be $5e^{-4}\boldsymbol{I}$.

All computations ran on-board the quadrotor, which was equipped with an Odroid XU4 computer. Because of the limited power of this CPU, the MPC horizon was reduced to 8 timesteps, with the timestep length extended to 50 ms, resulting in a 50 Hz update rate. These limitations emphasize the need for computationally efficient robust planning methods. The MPC parameters are provided in Tab. I.

To evaluate the performance of **D-MPC**, **PS-MPC**, and **SupeR-MPC** on hardware, the quadrotor was tasked with tracking a 1.5 m long straight-line trajectory while avoiding a cylindrical obstacle with radius 0.3 m. To prevent damage to the quadrotor, the obstacle was represented by a small dummy object. State measurements for both the quadrotor and the dummy were obtained using a motion capture system with injected random noise with covariance $5e^{-3}\boldsymbol{I}$.

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

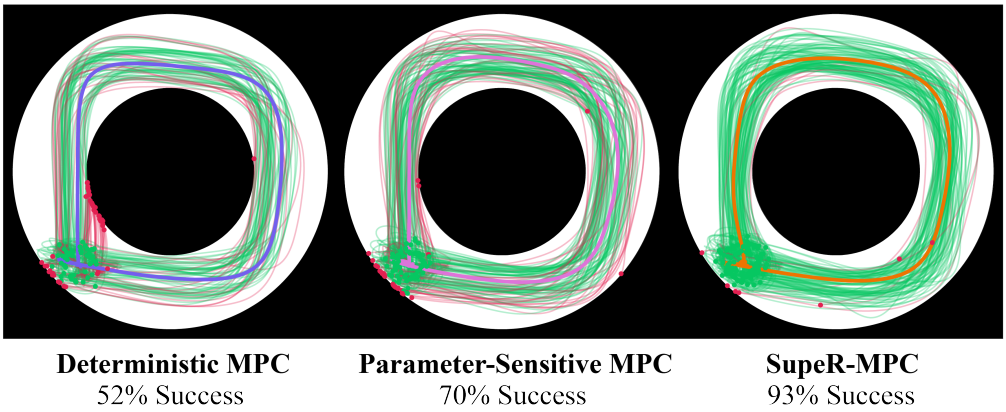

Fig. 4: **D-MPC** struggles with navigating a circular track in the presence of parameter and state estimation error, only achieving 52% success. **PS-MPC** demonstrates significant improvements, but without considering state estimation error, it is limited to 70% success. Synthesizing both sources of uncertainty allows **SupeR-MPC** to reach 93% success. Green trajectories successfully navigated the environment while red trajectories experienced a collision. The bold trajectories (purple for **D-MPC**, pink for **PS-MPC**, and orange for **SupeR-MPC**) represent the nominal trajectories under perfect estimation.

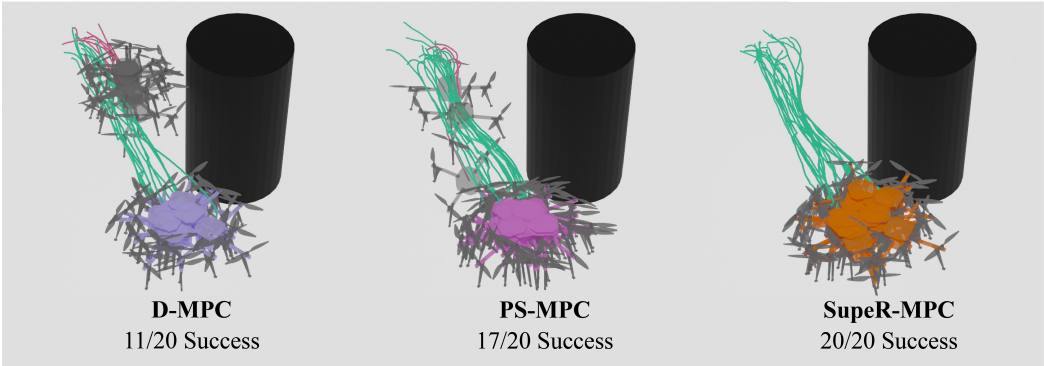

Fig. 5: In this hardware experiment, the quadrotor is tasked with robustly avoiding a static obstacle. Again, **PS-MPC** can improve upon **D-MPC**, which struggles greatly. However, **SupeR-MPC** displays superior robustness performance with a 100% success rate. Green trajectories successfully navigated around the obstacle while trajectories in red experienced a collision. The final position of successful trajectories is visualized with colored quadrotors (purple for **D-MPC**, pink for **PS-MPC**, and orange for **SupeR-MPC**), and gray quadrotors show the robot state at the moment of failure.

To assess robustness to real-world uncertainties, 20 trials of each algorithm were conducted, evaluating performance under mass parameter and state estimation uncertainty of the ego quadrotor and the obstacle. Due to real-world randomness, the initial state of the quadrotor varied across trials.

Since the cylindrical obstacle was not physically present, Fig. 5 renders each trial with the obstacle digitally visualized. Consistent with results from the simulation experiments, **D-MPC** struggled with safe obstacle avoidance, successfully completing only 11 out of 20 trials and colliding with the obstacle 9 times. **PS-MPC** demonstrated improved performance, successfully completing 17 out of 20 trials. However, in two instances, uncertainty caused the quadrotor to violate the obstacle chance-constraint, and in one case, MPC failed to converge due to a violation of the input chance-constraint, resulting in a loss of control and a crash. In contrast, **SupeR-MPC** succeeded in all 20 trials, underscoring the importance of jointly considering multiple sources of uncertainty for safe and robust trajectory planning. On the Odroid CPU, the average **SupeR-MPC** iteration required 13.57 ms, of which 6.32 ms was spent on sensitivity propagation. This experiment shows that **SupeR-MPC** is efficient enough to run in real time with limited computation power. To emphasize the challenge with implementing sampling-based methods, simulating a single timestep on the on-board CPU took on average 0.063 ms. For a short 8 timestep horizon, sampling a trajectory would take approximately 0.504 ms, yielding a maximum of 39 samples per iteration running at 50 Hz. Without more powerful parallel processing capabilities, this is not sufficient for MPPI or similar sampling-based methods.

To further evaluate the capabilities of **SupeR-MPC**, a final experiment introduced a dynamic obstacle in the form of a wheeled robot. Its state was estimated using a motion capture setup with injected sensor noise, and its motion was predicted

using a constant-velocity model. The quadrotor, still subject to the unknown mass disturbance, was tasked with following the same straight trajectory while robustly avoiding the moving obstacle at a perpendicular crossing, as shown in Fig. 1. Given that the previous hardware experiment suggested that neither **D-MPC** nor **PS-MPC** could reliably handle dynamic obstacle avoidance, this experiment was conducted exclusively with **SupeR-MPC**. The algorithm successfully completed all 20 trials, further emphasizing its effectiveness in real-world, uncertainty-rich environments.

VIII. Conclusion

Uncertainty is inherent in real-world robotic deployments, arising from state estimation errors, model parameter variations, unmodeled dynamics, and obstacle localization inaccuracies. Fortunately, these uncertainties can generally be effectively modeled using Gaussian approximations.

This work introduces **SupeR-MPC**, a robust trajectory planning framework that synthesizes multiple sources of Gaussian uncertainty to enhance probabilistic safety without excessive conservatism. Compared to Deterministic MPC and a state-of-the-art Parameter-Sensitive MPC, **SupeR-MPC** demonstrates significant performance improvements in both simulated and real-world quadrotor experiments.

Building on these contributions, future work will explore integrating **SupeR-MPC** into an adaptive control framework to better estimate unmodeled dynamics, such as quadrotor ground effects [22,23], and to accelerate computations through learned sensitivity matrices [13]. Additionally, we aim to extend this approach to hybrid systems, such as legged robots, by incorporating the saltation matrix [36].

References

[1] Cobot, "Building tomorrow, together with Cobot," accessed: 2025-02-27. [Online]. Available: https://www.co.bot/

[2] Zipline, "Welcome to the best delivery experience not on earth." accessed: 2025-02-27. [Online]. Available: https://www.flyzipline.com/

[3] Weave Robotics, "We're making Isaac," accessed: 2025-02-27. [Online]. Available: https://www.weaverobots.com/

[4] S. Thrun, *Probabilistic Robotics*. The MIT Press, 2005.

[5] B. Paden, M. Čáp, S. Yong, D. Yershov *et al.*, "A survey of motion planning and control techniques for self-driving urban vehicles," *IEEE Trans. on Intelligent Vehicles*, vol. 1, no. 1, pp. 33–55, 2016.

[6] R. E. Kalman, "A new approach to linear filtering and prediction problems," *Journal of Basic Engineering*, pp. 35–45, 1960.

[7] A. Ansari and T. Murphey, "Minimum sensitivity control for planning with parametric and hybrid uncertainty," *The Int. Journal of Robotics Research*, vol. 35, no. 7, pp. 823–839, 2016.

[8] H. Zhu and J. Alonso-Mora, "Chance-constrained collision avoidance for mavs in dynamic environments," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 776–783, 2019.

[9] K. J. Åström and T. Hägglund, *PID Controllers*. Instrument Society of America, 1995.

[10] J. B. Rawlings, D. Q. Mayne, and M. M. Diehl, *Model Predictive Control: Theory, Computation, and Design*. Nob Hill Publishing, 2017.

[11] T. Lew, R. Bonalli, and M. Pavone, "Chance-constrained sequential convex programming for robust trajectory optimization," *European Control Conf.*, 2020.

[12] J. Nubert, J. Köhler, V. Berenz, F. Allgöwer *et al.*, "Safe and fast tracking on a robot manipulator: Robust MPC and neural network control," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 3050–3057, 2020.

[13] S. Wasiela, M. Cognetti, P. Robuffo Giordano, J. Cortés *et al.*, "Robust motion planning with accuracy optimization based on learned sensitivity metrics," *IEEE Robotics and Automation Letters*, vol. 9, no. 11, pp. 10 113–10 120, 2024.

[14] T. Belvedere, M. Cognetti, G. Oriolo, and P. Robuffo Giordano, "Sensitivity-aware model predictive control for robots with parametric uncertainty," *IEEE Trans. on Robotics*, pp. 1–20, 2025.

[15] D. Bernardini and A. Bemporad, "Scenario-based model predictive control of stochastic constrained linear systems," *IEEE Conf. on Decision and Control*, pp. 6333–6338, 2009.

[16] P. Robuffo Giordano, Q. Delamare, and A. Franchi, "Trajectory generation for minimum closed-loop state sensitivity," *IEEE Int. Conf. on Robotics and Automation*, pp. 286–293, 2018.

[17] J. Zhu, J. J. Payne, and A. M. Johnson, "Convergent iLQR for safe trajectory planning and control of legged robots," *IEEE Int. Conf. on Robotics and Automation*, 2024.

[18] G. Williams, P. Drews, B. Goldfain, J. M. Rehg *et al.*, "Aggressive driving with model predictive path integral control," *IEEE Int. Conf. on Robotics and Automation*, pp. 1433–1440, 2016.

[19] J. Yin, Z. Zhang, and P. Tsiotras, "Risk-aware model predictive path integral control using conditional value-at-risk," *IEEE Int. Conf. on Robotics and Automation*, pp. 7937–7943, 2023.

[20] X. Shen, Y. Wang, K. Hashimoto, Y. Wu *et al.*, "Probabilistic reachable sets of stochastic nonlinear systems with contextual uncertainties," *Automatica*, vol. 176, p. 112237, 2025.

[21] Z. Wang, O. So, K. Lee, and E. A. Theodorou, "Adaptive risk sensitive model predictive control with stochastic search," *Conf. on Learning for Dynamics and Control*, vol. 144, pp. 510–522, 2021.

[22] R. Sinha, J. Harrison, S. M. Richards, and M. Pavone, "Adaptive robust model predictive control with matched and unmatched uncertainty," *American Control Conf.*, pp. 906–913, 2022.

[23] M. Bujarbaruah, X. Zhang, M. Tanaskovic, and F. Borrelli, "Adaptive stochastic MPC under time-varying uncertainty," *IEEE Trans. on Automatic Control*, vol. 66, no. 6, pp. 2840–2845, 2020.

[24] V. R. Desaraju, A. E. Spitzer, C. O'Meadhra, L. Lieu *et al.*, "Leveraging experience for robust, adaptive nonlinear mpc on computationally constrained systems with time-varying state uncertainty," *The International Journal of Robotics Research*, vol. 37, no. 13-14, pp. 1690–1712, 2018.

[25] E. A. Wan and R. Van Der Merwe, "The unscented Kalman filter for nonlinear estimation," *IEEE Adaptive Systems for Signal Processing, Communications, and Control Symposium*, pp. 153–158, 2000.

[26] T. Wakabayashi, Y. Suzuki, and S. Suzuki, "Dynamic obstacle avoidance for multi-rotor UAV using chance-constraints based on obstacle velocity," *Robotics and Auton. Systems*, vol. 160, p. 104320, 2023.

[27] J. A. Paulson, E. A. Buehler, R. D. Braatz, and A. Mesbah, "Stochastic model predictive control with joint chance constraints," *Int. Journal of Control*, vol. 93, no. 1, pp. 126–139, 2020.

[28] Y. K. Nakka and S.-J. Chung, "Trajectory optimization of chance-constrained nonlinear stochastic systems for motion planning under uncertainty," *IEEE Trans. on Robotics*, vol. 39, no. 1, pp. 203–222, 2023.

[29] K. Ryu and N. Mehr, "Integrating predictive motion uncertainties with distributionally robust risk-aware control for safe robot navigation in crowds," *IEEE Int. Conf. on Robotics and Automation*, pp. 2410–2417, 2024.

[30] M. Farina, L. Giulioni, and R. Scattolini, "Stochastic linear model predictive control with chance constraints – a review," *Journal of Process Control*, vol. 44, pp. 53–67, 2016.

[31] J. Gondzio, "Warm start of the primal-dual method applied in the cutting-plane scheme," *Mathematical Programming*, vol. 83, no. 1, pp. 125–143, 1998.

[32] A. Bambade, S. El-Kazdadi, A. Taylor, and J. Carpentier, "PROX-QP: Yet another quadratic programming solver for robotics and beyond," *Robotics: Science and Systems*, 2022.

[33] A. M. M. Leal, "autodiff, a modern, fast and expressive C++ library for automatic differentiation," 2018. [Online]. Available: https://autodiff.github.io

[34] H. Zhu, J. Hale, and E. Zhou, "Simulation optimization of risk measures with adaptive risk levels," *Journal of Global Optimization*, vol. 70, pp. 783–809, 2018.

[35] M. Saida, J. R. Medina, and S. Hirche, "Adaptive attitude design with risk-sensitive optimal feedback control in physical human-robot interaction," *IEEE Int. Symposium on Robot and Human Interactive Communication*, pp. 955–961, 2012.

[36] N. J. Kong, J. J. Payne, J. Zhu, and A. M. Johnson, "Saltation matrices: The essential tool for linearizing hybrid dynamical systems," *Proceedings of the IEEE*, vol. 112, no. 6, pp. 585–608, 2024.