Predictive Runtime Monitoring for Linear Stochastic Systems and Applications to Geofence Enforcement for UAVs

Hansol Yoon[1], Yi Chou[1], Xin Chen[2], Eric Frew[1], and Sriram Sankaranarayanan[1]

[1] University of Colorado, Boulder, USA
{hansol.yoon, yi.chou, eric.frew, srirams} @colorado.edu
[2] University of Dayton, Dayton, USA
xchen4@udayton.edu

Abstract. We propose a predictive runtime monitoring approach for linear systems with stochastic disturbances. The goal of the monitor is to decide if there exists a possible sequence of control inputs over a given time horizon to ensure that a safety property is maintained with a sufficiently high probability. We derive an efficient algorithm for performing the predictive monitoring in real time, specifically for linear time invariant (LTI) systems driven by stochastic disturbances. The algorithm implicitly defines a control envelope set such that if the current control input to the system lies in this set, there exists a future strategy over a time horizon consisting of the next N steps to guarantee the safety property of interest. As a result, the proposed monitor is oblivious of the actual controller, and therefore, applicable even in the presence of complex control systems including highly adaptive controllers. Furthermore, we apply our proposed approach to monitor whether a UAV will respect a "geofence" defined by a geographical region over which the vehicle may operate. To achieve this, we construct a data-driven linear model of the UAVs dynamics, while carefully modeling the uncertainties due to wind, GPS errors and modeling errors as time-varying disturbances. Using realistic data obtained from flight tests, we demonstrate the advantages and drawbacks of the predictive monitoring approach.

1 Introduction

We present efficient algorithms for the problem of monitoring viability in linear stochastic systems, and apply our approach to monitoring *geofencing* for UAVs. As UAVs become increasingly prevalent, the issue of such UAVs straying into critical infrastructure such as airports [3], residential buildings, military installations and other areas has gained critical importance. Geofences are virtual areas defined by a air traffic management authority inside which a UAV is permitted to operate [24]. However, breaches of these geofences, intentional or otherwise, need to be monitored. Furthermore, monitors need to provide advance warning to an operator that a breach is impending: such warnings can provide the

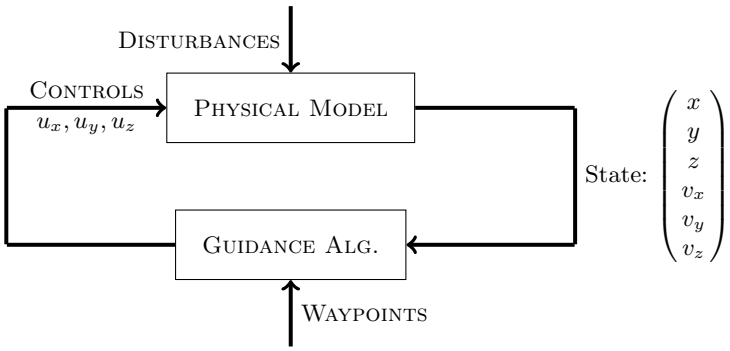

Fig. 1. Schematic diagram of the closed loop system showing the plant and the controller.

traffic manager valuable time to undertake possible defensive measures, or even help with post-incident investigations. Formally, a *geofence* is defined by a set $F \subseteq \mathbb{R}^3$, specifying legal (x, y, z) positions of the UAV.

Definition 1 (Geofence Monitoring Problem). *Given the current* state *of the aircraft (assume current time is $t = 0$), will its position $(x(t), y(t), z(t))$ remain inside a geofence F over a time interval $[0, T]$ in the future.*

As such, geofences are safety properties and monitoring them over a finite time horizon requires over-approximating the set of possible reachable states $\mathcal{R}_T$ over the time interval $[0, T]$ of interest, enabling us to check the condition $\mathcal{R}_T \subseteq F$. Naturally, for runtime monitoring, the reachable state estimate must be computed and checked against the specifications during the deployment. However, reachability analysis is a difficult computation that is complicated by multiple factors. As shown in Figure 1, the UAV system is a closed-loop consisting of the physical dynamics of the aircraft and the on-board navigation/guidance controller.

1. The physical dynamics are uncertain and influenced by unknown, stochastic disturbances such as the wind.
2. The navigation and guidance component are often nonlinear, involving proprietary autopilot systems and influenced by *waypoints* or other mission specifications provided by the operator.

Nonlinear reachability analysis for hybrid systems is an active area of research [7, 9, 2, 12]. However, a rapid real-time stochastic reachability analysis of the closed loop in Fig. 1 to check geofence violations is beyond the capability of even the most sophisticated tools, at the time of writing. In this paper, we *sidestep* these issues to derive an efficient solution suitable for online monitoring:

1. We use data-driven linear forecast models for the UAVs that includes the effect of wind uncertainty and unmodeled dynamics.
2. Instead of monitoring whether the closed loop will satisfy the safety property, we ask a different question of *viability* rather than *safety*.

Definition 2 (θ-viable). *A state $\boldsymbol{x}(0)$ of the plant is* θ-viable *at a future time T, iff there exists a control strategy over $[0, T]$ such that the probability of safety at time T is at least θ — i.e., $\mathbb{P}(\boldsymbol{x}(T) \in F) \geq \theta$.*

In practice, θ is set to a number close to 1, indicating the desired level of confidence in the system state. Viability monitoring differs from the standard safety monitor in the following manner: rather than ask whether the particular control law can act to prevent a future failure, we ask the question whether there exists *some control strategies* that can prevent failure. In particular, this strategy may differ from the one used by the actual controller. It is possible that a system state is viable but the specific control law employed can lead to failure. On the other hand, if a system is deemed *not viable* for a future time, it is essentially at the mercy of the environment: no controller can guarantee safety for such a system with probability exceeding θ. The advantages of the viability monitoring approach used in this paper include:

(a) It is purely a property of the plant model and does not involve the controller. This allows us to handle systems that are controlled by complex control strategies that may be proprietary, or in general, hard to reason about in real-time. This may include neural networks and learning-based controllers that are now quite popular in autonomous systems.
(b) Finally, we show a sufficient condition that yields a sound monitor for viability for linear stochastic systems. This monitor can be implemented efficiently with an efficient online runtime monitoring strategy.

However, viability monitoring suffers from key disadvantages from the point of view of runtime monitoring for safety:

(a) A viability monitor can miss impending safety violation (false negatives). The closed loop system may, in fact, violate the safety property despite there being a strategy to avoid that violation.
(b) The approach can potentially yield *false alarms*, wherein a failure of viability does not necessarily lead to a property failure. It is entirely possible that the environment does not exhibit the worst case behavior leading to a failure of viability, but at the same time allowing the system to remain safe. Nevertheless, such situations are important to note and fix, lest they lead to an actual violation in a different instance.

The gap between safety monitoring and the notion of viability monitoring presented in this paper can be narrowed by constraining the definition of viability, in order to account for properties of the controller. The key here is to restrict the control strategies used in Def. 2 to a smaller set of strategies that include those employed by the controller. For instance, the range of control inputs used by the actual controller implementation can be used to restrict the strategies considered in Def. 2. More general abstractions of the controller, if available, can also be employed in this manner. In doing so, the drawbacks mentioned above can be partly addressed.

The rest of the paper is organized as follows: Section 2 describes the data-driven modeling approach, Section 3 presents the basic algorithm for monitoring viability efficiently, Section 4 presents how this algorithm is adapted and makes

more efficient specifically for geofencing. Finally, an evaluation based on actual UAV test flight data under windy conditions is presented in Section 5. The evaluation shows that the approach is generally successful in monitoring violations of safety properties sufficiently ahead of time. Very few violations ($< 1\%$) are missed by our monitors, while at the same time the false positive rate is very small ($< 0.3\%$).

1.1 Related Work

The use of real-time monitors to predict and act against imminent property violations forms the basis for runtime assurance using L1-Simplex architectures that switch between a lower performance but formally validated control when an impending failure is predicted [23]. However, the key issue lies in the process of predicting impending failures with high confidence. Often, predicting failures involves computing control invariant sets, or solving reachability problems in real time [14, 8]. Previous work by some of the authors use a game theoretic approach to monitor impending property violations for linear systems [10]. This paper directly extends our previous work to probabilistic models. Additionally, we showcase a realistic application to monitoring geofence violations. Recently, the idea of *shielding* has been proposed for runtime assurance of autonomous systems with human operators and learning-enabled systems. Formally, a shield is a component that interfaces between a human operator or a complex controller and the plant that can modify the control inputs in real time to avoid an erroneous state, or recover from one as quickly as possible [15]. The idea has been applied to safe reinforcement learning wherein the shield restricts the actions of the learner during the training phase and prevents a specification violation during deployment phase [1]. Although the present work focuses on efficient monitoring, our approach described here lends itself easily to the synthesis of a shield component. However, a key difference is that our work focuses on predicting the satisfaction of safety properties over a future state using a data-driven model. Such a prediction is complementary to the process of shield synthesis that focuses on corrective actions to avoid failures, or recover from them.

Recently, data-driven predictions of impending property violations have received much attention. Phan et al. demonstrate the use of neural network classifiers combined with offline statistical model checkers to predict if the current state is likely to violate a property during a future time horizon [20]. Neural networks are black boxes whose predictions must be trusted. Nevertheless, the recent surge of interest in verifying neural networks bodes well for the use of these models in monitoring applications.

Lygeros and Prandini (along with coworkers) have investigated stochastic reachability analysis approaches for detecting and avoiding collisions between aircrafts [22, 16]. Their approach bears many similarities with ours: the use of stochastic models to predict future positions with uncertainties and the use of reachability analysis to estimate probability of collision. However, there are many key differences: first, we use discrete time data-driven models inferred in

real time as we obtain recent historical data. We also monitor viability in order to avoid reasoning about the on-board controllers.

Stevens et al. investigate the problem of monitoring geofences: on one hand their work can monitor nonconvex geofence regions [25]. However, their monitoring is not predictive: to enable prediction they simply narrow the safety region so that a violation of the conservative region indicates a potential impending violation of the original specification. In this work, we tackle the problem of predicting future loss in viability, more systematically, while accounting for wind disturbances.

The work of Moosbrugger et al. presents another key application of data-driven models for the runtime monitoring of UAVs [19]. Their approach uses a combination of a Bayes network to model how various observable events may result in hardware/software failures or security threats during the operation of a UAV. Our approach also uses data-driven models, but to predict future positions and velocities. Also, we monitor viability properties involving future positions of the aircraft. We hope to investigate how ideas from Moosbrugger et al. (ibid) can be incorporated into our framework to fuse observable events on-board UAVs to better predict future positions. Besides predicting collisions, or monitoring onboard health, data-driven models can be applicable to other aspects of flight such as remaining fuel/power. Chati and Balakrishnan present a data-driven Gaussian process model of aircraft fuel consumption, an important prediction target during flight [6].

Vinod et al. consider the problem of finding control inputs to guarantee future safety property of a linear stochastic system using ideas such as dynamic programming, chance constrained optimization and Fourier transform approaches [26, 27]. Our work in contrast attempts to get rid of the stochastic disturbances by finding a robust set to contain the disturbances. This has the advantage of computational speed suitable for real-time monitoring. However, we can only provide a sound rather than a precise solution for the monitoring problem. Quantifying the gap between the two approaches will be performed in our future work.

2 Data-Driven Model

In this section, we review our approach to formulating data-driven models that augment an existing physical model of aircraft dynamics. We first start with the overall structure of the model and explain how various parts of the model are infered as well as the process of modeling the uncertainty.

We start with a simple physical model of an aircraft with current position (x, y, z), velocities (v_x, v_y, v_z) along a static reference frame fixed to the earth and accelerations (u_x, u_y, u_z) that are treated as control inputs. Let δ be a fixed step size. Our experiments use $\delta = 0.4$ seconds based on the data refresh rate

from our UAV platforms.

$$\left[\begin{array}{l} x(t+\delta) = x(t) + \delta v_x(t) + \frac{1}{2}\delta^2 u_x + e_x(t+\delta) \\ y(t+\delta) = y(t) + \delta v_y(t) + \frac{1}{2}\delta^2 u_y + e_y(t+\delta) \\ z(t+\delta) = z(t) + \delta v_z(t) + \frac{1}{2}\delta^2 u_z + e_z(t+\delta) \\ v_x(t+\delta) = v_x(t) + \delta u_x(t) + e_{vx}(t+\delta) \\ v_y(t+\delta) = v_y(t) + \delta u_y(t) + e_{vy}(t+\delta) \\ v_z(t+\delta) = v_z(t) + \delta u_z(t) + e_{vz}(t+\delta) \end{array}\right] \tag{1}$$

We have introduced terms $e_x, e_y, e_z, e_{vx}, e_{vy}, e_{vz}$ model discrepancies between the observed data at time $t+\delta$ and that predicted by a simple Newtonian particle model of the UAV. We will call Eq. (1) as our core model that incorporates the physical knowledge as well as unexplained discrepancies that may arise due to model mismatch as well as the disturbances. The key is to model these discrepancies as a function of the *recent* historical data. We wish to model each error as a function of the past:

$$e_x(t+\delta) = a_0 e_x(t) + \cdots + a_{p-1} e_x(t-(p-1)\delta) + w(t+\delta)\,. \tag{2}$$

Here $a_0, \ldots, a_{p-1}$ are called the *autoregressive* coefficients, p is the history length and w is a random variable drawn from a known probability distribution, and independent of the current state variables. Such a model is called an *autoregressive* (AR) model. Likewise, we formulate AR models for $e_y, e_z, e_{vx}, \ldots, e_{vz}$. It is important to fix the form of this model and the length of the historical data needed.

2.1 Autoregressive Models

In this paper, we will model discrepancies $\boldsymbol{e}$ using a simple yet effective modeling paradigm called *auto-regressive* models with *exogenous inputs* (ARX) models[18, 5]. ARX models are a simple approach to formulating linear models in time-series forecasting. They are widely used in numerous applications especially where simplicity and careful modeling of uncertainties are important. Furthermore, we prefer to explore the capabilities of such a simple approach to motivate whether more complex nonlinear models can be useful. Consider again the form of the discrepancy model in Eq. (2). To infer the coefficients $a_0, \ldots, a_{p-1}$, we first use our data to compute $e_x(t), e_y(t), \ldots, e_{vz}(t)$ for each time $t = \delta, \ldots, N\delta$ by substituting the known data in the core model (1).

Inferring Models (Regression): For discrepancy variables e_x, e_y, e_z, e_{vx}, e_{vy}, and e_{vz}, we set up models for fixed history lengths p as in Eq. (2). The disturbance term $w(t+\delta)$ is removed from the model and estimated later. For each time $t = 0, \delta, \ldots, (N-1)\delta$, we obtain a single equation involving the unknowns $\boldsymbol{c} : (a_0, \ldots, a_{p-1})$. As a result of plugging in the data, we obtain a system of equations of the form: $A\boldsymbol{c} \approx \boldsymbol{b}$, wherein $A, \boldsymbol{b}$ are formulated from the data. However, there will be more equations than there are unknowns. Thus, our goal

is not to find a "perfect" solution to the equations, since one will not exist in all but the rarest of cases. We use linear regression to minimize the residual error $\min(||A\boldsymbol{c}-\boldsymbol{b}||_2^2)$. Often, we wish to minimize the residual error subject to sparsity constraints on the coefficient. One approach to do is called *ridge regression* [13], wherein we minimize $\min(||A\boldsymbol{c}-\boldsymbol{b}||_2^2+\alpha||\boldsymbol{c}||_2^2)$, wherein α is a constant. The objective function represents a tradeoff between achieving low residual errors versus keeping the sizes of the coefficients small. This is adjusted using the parameter α. The regression problem is solved using well-studied approaches from linear algebra such as Cholesky decomposition or conjugate gradient approaches. The resulting solution yields the coefficients $\boldsymbol{c}$ of the ARX model.

Once we finish solving the regression, we estimate the distribution of the residuals $\boldsymbol{w}(t)$, by modeling the distribution of the residual error vector $A\boldsymbol{c}-\boldsymbol{b}$. This can be achieved in one of two ways: (a) carrying the residual vector $A\boldsymbol{c}-\boldsymbol{b}$ and randomly sampling a value from it when needed to obtain a sample; (b) modeling the error as a distribution such as a Gaussian distribution by computing its mean and standard deviation. Statistical tests such as the chi-squared tests can help us estimate how close the residual distribution is to being a Gaussian. We adopt the latter approach since for all our datasets seen in this paper, we obtain excellent fits to the Gaussian distribution. The distribution mean is obtained as the empirical mean of the residuals $A\boldsymbol{c}-\boldsymbol{b}$ and the standard deviation is obtained as the empirical standard deviation. The final form of the ARX model is therefore given by Eq. (2) with $\boldsymbol{w}$ modeled as a sample from a Gaussian distribution with a given mean and standard deviation.

Example 1. The equation below partially illustrates a model for x, y, z with e_x, e_y and e_z. Note that $\delta = 0.4$.

$$\begin{aligned}
x(t+\delta) &= x(t) + \delta v_x(t) + \tfrac{1}{2}\delta^2 u_x + e_x(t)\\
y(t+\delta) &= y(t) + \delta v_y(t) + \tfrac{1}{2}\delta^2 u_y + e_y(t)\\
z(t+\delta) &= z(t) + \delta v_z(t) + \tfrac{1}{2}\delta^2 u_z + e_z(t)\\
e_x(t+\delta) &= 0.57e_x(t) + 0.39e_x(t-\delta) + w_1 \ (\sigma_1: \ 0.13)\\
e_y(t+\delta) &= 0.49e_y(t) + 0.27e_y(t-\delta) + w_2 \ (\sigma_2: \ 0.14)\\
e_z(t+\delta) &= 1.35e_z(t) - 0.39e_z(t-\delta) + w_3 \ (\sigma_3: \ 0.053)
\end{aligned}$$

The terms w_1, w_2, w_3 are Gaussian random variables with 0 mean and variances σ_1, σ_2 and σ_3, respectively, as shown.

Model Updating: We briefly comment on model updating. UAV environments involve changes to aircraft dynamics due to wear and tear, payload variations and fuel loss as well as changes in wind conditions. As a result, it is important to update the model using the "latest" available data. In our experiments, the process of constructing the model from nearly 2000 data points takes less than 0.1 seconds. As a result, it is possible to keep updating the model in real time. Another alternative is to update the distributions of $\boldsymbol{w}(t)$ over time using the residuals computed based on real data. For the rest of the paper, we consider the model to remain fixed for all times. Schemes for updating the model in real time are beyond the scope of the current work. We hope to investigate them as part of future work.

3 Viability Monitoring

We will now present viability monitoring for linear stochastic systems.

Definition 3 (Plant Model). *A plant model $\mathcal{P}$ is given by a tuple $\langle \delta, A, B, C, \mathcal{U}, \mathcal{D} \rangle$ with a time step $\delta > 0$, state vector $\boldsymbol{x}(t) \in \mathbb{R}^n$, a control input vector $\boldsymbol{u}(t) \in \mathbb{R}^m$ and a disturbance input vector $\boldsymbol{w}(t) \in \mathbb{R}^k$. At each step, the state of the plant model is updated according to the matrices $A \in \mathbb{R}^{n \times n}$, $B \in \mathbb{R}^{n \times m}$ and $C \in \mathbb{R}^{n \times k}$.*

$$\boldsymbol{x}(t+\delta) = A\boldsymbol{x}(t) + B\boldsymbol{u}(t) + C\boldsymbol{w}(t)\,,$$

wherein $\boldsymbol{u}(t) \in \mathcal{U}$ and $\boldsymbol{w}(t) \sim \mathcal{D}$, i.e., $\boldsymbol{w}(t)$ is distributed according to $\mathcal{D}$.

We will make the following assumptions on the structure of the plant model.

(1) The set $\mathcal{U}$ is a box wherein each component u_i belongs to an interval $[a_{u_i}, b_{u_i}]$. Later, this assumption will enable us to simplify the overall monitoring algorithm.
(2) The distribution $\mathcal{D}$ is normal wherein each component $w_i(t)$ for $1 \leq i \leq k$ is distributed according to a gaussian random variable with mean 0 and standard deviation σ_i. Furthermore, random variable $w_i(t), w_j(t)$ are pairwise independent for $i \neq j$. Also, random variables $\boldsymbol{w}(t), \boldsymbol{w}(t')$ are independent for $t' \neq t$.

We note that the data-driven model discussed in eqs. (1) and (2) in Section 2 fit the structure of our plant model. The state $\boldsymbol{x}$ consists of the following:

$$\underbrace{x(t), y(t), z(t)}_{\text{Position}}, \underbrace{v_x(t), v_y(t), v_z(t)}_{\text{Velocities}}, \underbrace{e_x(t), \ldots, e_x(t-(p-1)\delta)}_{\text{ARX model state}}, \cdots, \underbrace{e_{v_z}(t), \ldots, e_{v_z}(t-(p-1)\delta)}_{\text{ARX model state}}.$$

Let F be a set of safe states $\boldsymbol{x}$ defined by constraints of the form $P\boldsymbol{x} \leq \boldsymbol{q}$ for a $l \times n$ matrix P and $l \times 1$ vector $\boldsymbol{q}$. For a fixed parameter $\theta \in (0,1)$, we define θ viability formally in terms of the plant model.

Definition 4 (θ-viable). *A state $\boldsymbol{x}$ is said to be θ-viable with respect to a plant model $\mathcal{P}$ and time $T = N\delta$ if an only if*

$$(\exists \boldsymbol{u}(0), \ldots, \boldsymbol{u}((N-1)\delta) \in \mathcal{U}^N)\ \ \mathbb{P}_{\boldsymbol{w}(0)\sim\mathcal{D},\ldots,\boldsymbol{w}((N-1)\delta)\sim\mathcal{D}}\left(\boldsymbol{x}(N\delta) \in F\right)\ \geq\ \theta\,. \quad (3)$$

3.1 Sufficient Condition for θ-Viability

We will now present the derivation for a sufficient condition for θ viability given an initial state $\boldsymbol{x}(0)$. For simplicity, we will assume that there is no uncertainty with respect to the initial state itself. However, such uncertainties can be easily modeled in our framework. Let $\boldsymbol{u}(t)$ denote the control inputs at time $t \in \{0, \ldots, (N-1)\delta\}$, such that $\boldsymbol{u}(t) \in \mathcal{U}$. Let $\boldsymbol{v}_j : \begin{pmatrix} \boldsymbol{u}(0) \\ \vdots \\ \boldsymbol{u}((j-1)\delta) \end{pmatrix}$ for $j \geq 1$, be the vector that collects the control inputs over time points $t \in \{0, \delta, \ldots, (j-1)\delta\}$.

Let $\mathcal{V}_j$ be the set of admissible values of $\boldsymbol{v}_j$. Finally, let $\boldsymbol{z}_j : \begin{pmatrix} \boldsymbol{w}(0) \\ \vdots \\ \boldsymbol{w}((j-1)\delta) \end{pmatrix}$ collect the disturbance inputs over the time points $\{0, \delta, \ldots, (j-1)\delta\}$. We can calculate $\boldsymbol{x}(N\delta)$ as follows:

$$\boldsymbol{x}(N\delta) = A_N \boldsymbol{x}(0) + B_N \boldsymbol{v}_N + C_N \boldsymbol{z}_N \,. \tag{4}$$

The matrix A_N is defined by the recurrence: $A_j = AA_{j-1},\ j \geq 2$, with base case $A_1 = A$. B_N is defined by the recurrence $B_j = \begin{bmatrix} AB_{j-1} & B \end{bmatrix}$ for $j \geq 2$, with base case $B_1 = B$. Likewise, $C_j = \begin{bmatrix} AC_{j-1} & C \end{bmatrix}$ for $j \geq 2$, with base case $C_1 = C$.

Lemma 1. *Given A_N, B_N, C_N and the vectors $\boldsymbol{v}_j, \boldsymbol{z}_j$, as described above, for any initial state $\boldsymbol{x}_0$ at time $t = 0$, $\boldsymbol{x}(N\delta) = A_N \boldsymbol{x}_0 + B_N \boldsymbol{v}_N + C_N \boldsymbol{z}_N$.*

Proof. Proof is by induction on N. For $N = 1$, the formula describes a single step of the plant model. Assume that it holds for $N = j$. We have that

$$\begin{aligned}
\boldsymbol{x}((j+1)\delta) &= A\boldsymbol{x}(j\delta) + B\boldsymbol{u}(j\delta) + C\boldsymbol{w}(j\delta) \\
&= A(A_j \boldsymbol{x}_0 + B_j \boldsymbol{v}_j + C_j \boldsymbol{z}_j) + B\boldsymbol{u}(j\delta) + C\boldsymbol{w}(j\delta) \\
&= AA_j \boldsymbol{x}_0 + (AB_j \boldsymbol{v}_j + B\boldsymbol{u}(j\delta)) + (AC_j \boldsymbol{z}_j + C\boldsymbol{w}(j\delta)) \\
&= A_{j+1} \boldsymbol{x}_0 + \begin{bmatrix} AB_j & B \end{bmatrix} \begin{pmatrix} \boldsymbol{v}_j \\ \boldsymbol{u}(j\delta) \end{pmatrix} + \begin{bmatrix} AC_j & C \end{bmatrix} \begin{pmatrix} \boldsymbol{w}_j \\ \boldsymbol{w}(j\delta) \end{pmatrix} \\
&= A_{j+1} \boldsymbol{x}_0 + B_{j+1} \boldsymbol{v}_{j+1} + C_{j+1} \boldsymbol{z}_{j+1}
\end{aligned}$$

Robust Disturbance Sets: Eq. (3) involves checking an existentially quantified formula involving an integration over $\boldsymbol{z}_N$. Such assertions are called *chance constraints*, and can be quite expensive to verify[26]. We perform a reduction of the chance constraints through a simple trick of replacing the integration with a *forall* quantifier by using a *robust disturbance set*.

Definition 5 (θ-robust set). *Let $\boldsymbol{z} \sim \mathcal{D}$. A set Z_θ is said to be θ-robust for distribution $\mathcal{D}$ if $\mathbb{P}(\boldsymbol{z} \in Z_\theta) \geq \theta$.*

In general, there are many possible choices of θ-robust sets, given the distribution of each disturbance input vector $\boldsymbol{w}(t)$. For instance, if $\boldsymbol{w}(t)$ is a normally distributed random variable $\mathcal{N}(\mathbf{0}, \sigma^2 I_{n\times n})$ with mean $\mathbf{0}$ and co-variance matrix $\Sigma :\ \sigma^2 I_{n\times n}$, then the following hyper-spherical region is θ-robust:

$$Z_{\theta,\Sigma} = \{\boldsymbol{w} \mid \boldsymbol{w}^T \Sigma^{-1} \boldsymbol{w} \leq \chi^2_n(1-\theta)\}\,, \tag{5}$$

wherein $\chi^2_n(1-\theta)$ represents the upper $(1-\theta)$ quantile of the standard chi-squared distribution with n-degrees of freedom, whose value can be looked up from a table. Therefore, to derive a sufficient condition for checking θ-viability, we first select a θ-robust set Z_θ such that $\mathbb{P}(\boldsymbol{z}_N \in Z_\theta) \geq \theta$. Next, we check the assertion:

$$(\exists \boldsymbol{v}_N \in \mathcal{V}_N)\ (\forall \boldsymbol{z}_N \in Z_\theta)\ \boldsymbol{x}(N\delta) \in F \tag{6}$$

Lemma 2. *The condition in Eq.* (6) *implies the viability definition in Eq.* (3).

Proof. Let us assume that $\boldsymbol{x}(0)$ satisfies Eq. (6).

$$\begin{aligned}\mathbb{P}(\boldsymbol{x}(N\delta) \in F) &\geq \mathbb{P}(\boldsymbol{x}(N\delta) \in F | \boldsymbol{z}_N \in Z_\theta)\mathbb{P}(\boldsymbol{z}_N \in Z_\theta) \\ &\geq \mathbb{P}(\boldsymbol{z}_N \in Z_\theta) \ \leftarrow \text{ because } \boldsymbol{x}(N\delta) \in F, \ \forall \ \boldsymbol{z}_N \in Z_\theta \ . \\ &\geq \theta \ \leftarrow \text{ by defn. of } Z_\theta \, .\end{aligned}$$

Therefore, once Z_θ is chosen, the computation reduces to checking (6). However, this involves a single quantifier alternation and thus, computational expensive. However, the structure of the plant model can be exploited as follows:

First, we note that the set of possible states $\boldsymbol{x}(N\delta)$ in Eq. (4) is the sum of three individual components that can each be chosen independently of the others: (a) constant vector $A_N\boldsymbol{x}(0)$ (no real choice here), (b) a vector of the form $B_N\boldsymbol{v}_N$ indicating the contribution from the control strategy, and (c) a disturbance vector chosen from the set:

$$\hat{Z} : \ \{C_N\boldsymbol{z}_N \mid \boldsymbol{z}_N \in Z_\theta\} \tag{7}$$

Thus, the key observation is that the reachable set at time $N\delta$ is a *Minkowski sum* of three sets, as described above. We will now define the operation of *Minkowski difference* of two sets and directly use it to remove the quantifier alternation in Eq. (6).

Definition 6 (Minkowski Difference). *Let $A, B \subseteq \mathbb{R}^n$ be two subsets. The* Minkowski difference *$A \ominus B$ is defined as the set: $A \ominus B : \ \{\boldsymbol{a} \mid (\forall \ \boldsymbol{b} \in B) \ \boldsymbol{a} + \boldsymbol{b} \in A\}$.*

Therefore, returning to Eq. (6), we use the definition of $\hat{Z}$ from Eq. (7), and the notion of Minkowski difference to obtain an equivalent condition:

$$(\exists \ \boldsymbol{v}_N \in \mathcal{V}_N) A_N\boldsymbol{x}_0 + B_N\boldsymbol{v}_N \in (F \ominus \hat{Z}) \tag{8}$$

Lemma 3. *The condition in Eq.* (8) *is equivalent to that in Eq.* (6).

Proof. Let us assume that Eq. (8) holds for some initial state $\boldsymbol{x}_0$. Let $\boldsymbol{v}_N \in \mathcal{V}_N$ be the value of control inputs such that $A_N\boldsymbol{x}_0 + B_N\boldsymbol{v}_N \in (F \ominus \hat{Z})$. Therefore, by definition 6, $(\forall \ \hat{\boldsymbol{z}} \in \hat{Z}) \ A_N\boldsymbol{x}_0 + B_N\boldsymbol{v}_N + \hat{\boldsymbol{z}} \in F$. Recalling definition of $\hat{Z}$ from, Eq. (7), we obtain: $(\forall \ \boldsymbol{z} \in Z_\theta) \ A_N\boldsymbol{x}_0 + B_N\boldsymbol{v}_N + C\boldsymbol{z} \in F$. This is the condition from Eq. (6). The converse is proved by reversing the argument above.

Computing Minkowski Difference: It is essential to compute the Minkowski difference between a polyhedron F given by constraints $P\boldsymbol{x} \leq \boldsymbol{q}$ and a set $\hat{Z}$. We use the following properties to compute this difference efficiently for polyhedral sets.

Lemma 4. *Consider a family of sets A_j for $j = 1, \ldots, l$. $(\bigcap_{j=1}^{l} A_j) \ominus B = \bigcap_{j=1}^{l}(A_j \ominus B)$.*

Proof. Let $\boldsymbol{a} \in (\bigcap_{j=1}^{l} A_j) \ominus B$. Therefore, for any $\boldsymbol{b} \in B$, we have $(\boldsymbol{a} + \boldsymbol{b}) \in \bigcap_{j=1}^{l} A_j$. Thus, forall $\boldsymbol{b} \in B$, we have $(\boldsymbol{a} + \boldsymbol{b}) \in A_j$ for each $j = 1, \ldots, l$. Thus, $\boldsymbol{a} \in A_j \ominus B$, and hence, $\boldsymbol{a} \in \bigcap_{j=1}^{l}(A_j \ominus B)$. Conversely, let $\boldsymbol{a} \in \bigcap_{j=1}^{l}(A_j \ominus B)$. Therefore, $\boldsymbol{a} \in A_j \ominus B$ for each $j = 1, \ldots, l$. Thus forall $\boldsymbol{b} \in B$, we have $\boldsymbol{a}+\boldsymbol{b} \in A_j$. Thus, forall $\boldsymbol{b} \in B$, $\boldsymbol{a} + \boldsymbol{b} \in \bigcap_j A_j$. Hence, $\boldsymbol{a} \in (\bigcap_{j=1}^{l} A_j) \ominus B$.

Lemma 5. *Let A_j be a set denoted by a half-space $\{\boldsymbol{x} \mid \boldsymbol{a}_j \cdot \boldsymbol{x} \geq b_j\}$ and B be a compact set. Let Δ_j be the result of the optimization problem* $\min\ \boldsymbol{a}_j \cdot \boldsymbol{x}$ s.t. $\boldsymbol{x} \in B$. *The set $A_j \ominus B$ is the half-space $\hat{A}_j$ given by $\{\boldsymbol{x} \mid \boldsymbol{a}_j \cdot \boldsymbol{x} \geq b_j - \Delta_j\}$.*

Proof. Choose any $\boldsymbol{b} \in B$ and $\boldsymbol{x} \in \hat{A}_j$. We have $\boldsymbol{a}_j \cdot (\boldsymbol{x} + \boldsymbol{b}) \geq \boldsymbol{a}_j \cdot \boldsymbol{x} + \boldsymbol{a}_j \cdot \boldsymbol{b} \geq (b_j - \Delta_j) + \Delta_j \geq b_j$. Therefore, $\boldsymbol{x} + \boldsymbol{b} \in A_j$. We conclude $\hat{A}_j \subseteq (A_j \ominus B)$.

Furthermore, since B is a compact set, the minimum of the optimization problem is achieved at a point $\boldsymbol{b}^* \in B$, i.e., $\boldsymbol{a}_j \cdot \boldsymbol{b}^* = \Delta_j$. Let $\boldsymbol{x}$ be any point in $A_j \ominus B$. It follows that $\boldsymbol{x} + \boldsymbol{b}^* \in A_j$. I.e., $\boldsymbol{a}_j(\boldsymbol{x} + \boldsymbol{b}^*) \geq b_j$. Therefore, $\boldsymbol{a}_j\boldsymbol{x} \geq b_j - \boldsymbol{a}_j \cdot \boldsymbol{b}^* = b_j - \Delta_j$. Therefore, $\boldsymbol{x} \in \hat{A}_j$. Thus we conclude $\hat{A}_j \supseteq (A_j \ominus B)$.

The lemma above provides us the ingredients for computing $F \ominus \hat{Z}$ for a polyhedron F given by the intersection of $l > 0$ half-spaces, and a set $\hat{Z}$. This involves solving optimization problems of the form ($\min\ \boldsymbol{a} \cdot \boldsymbol{z}$ s.t. $\boldsymbol{z} \in \hat{Z}$). If $\hat{Z}$ is a convex set, then computing $\boldsymbol{\Delta}$ can be performed efficiently using convex optimization solvers[4].

Lemma 6. *The Minkowski difference of a polyhedron $F : P\boldsymbol{x} \leq \boldsymbol{q}$ and a compact set $\hat{Z}$ is given by a polyhedron $G :\ F \ominus \hat{Z}$ of the form $G :\ P\boldsymbol{x} \leq \boldsymbol{q} - \boldsymbol{\Delta}$, wherein $\Delta_j :\ \min\ P_j\boldsymbol{x}$ s.t. $\boldsymbol{x} \in \hat{Z}$.*

Proof. Proof combines the previous two lemmas: a polyhedron is an intersection of half-spaces and we can compute Minkowski difference for each half-space.

3.2 Overall Monitoring Algorithm

We will now present the overall monitoring algorithm as a combination of (a) upfront offline calculations, and (b) the real-time online monitor.

Offline Calculations: The offline calculations are performed given the plant model $\mathcal{P} : \langle A, B, C, \mathcal{U}, \mathcal{D} \rangle$ (Def. 3), and the safety property F as a convex polyhedron $P\boldsymbol{x} \leq \boldsymbol{q}$.

1. Compute matrices A_N, B_N and C_N using $\Theta(N)$ matrix multiplication operations.
2. Compute a θ-robust set Z_θ. Since the disturbance inputs are distributed normally, we use Eq. (5) to choose one Z_θ.
3. Compute the polyhedron for $F \ominus \hat{Z}$, wherein $\hat{Z} : \{C_N\boldsymbol{z} \mid \boldsymbol{z} \in Z_\theta\}$ using Lemma 6. Since Z_θ is a convex quadratic, this is technically a *quadratically constrained quadratic program* (QCQP).

Online Calculations: The results of the offline calculations include matrices (A_N, B_N) and the set $G : F \ominus \hat{Z}$. The online monitor receives the current state estimate $\boldsymbol{x}_0$ and the current controller input $\boldsymbol{u}_0$. Since $\boldsymbol{v}_N = (\boldsymbol{u}(0), \boldsymbol{u}(\delta), \cdots, \boldsymbol{u}((N-1)\delta))^T$. We will set $\boldsymbol{u}(0) = \boldsymbol{u}_0$ and let $\boldsymbol{u}(\delta), \ldots, \boldsymbol{u}((N-1)\delta)$ be unknown decision variables. The monitor checks the following constraint (Eq. (8)):

$$(\exists\ \boldsymbol{u}(\delta) \in \mathcal{U}, \cdots, \boldsymbol{u}((N-1)\delta) \in \mathcal{U})\ A_N \boldsymbol{x}_0 + B_N \begin{pmatrix} \boldsymbol{u}_0 \\ \boldsymbol{u}(\delta) \\ \vdots \\ \boldsymbol{u}((N-1)\delta) \end{pmatrix} \in G \qquad (9)$$

Note that we can use a linear programming (LP) solver to check the condition above. If it is feasible, we conclude that the system is viable. Otherwise, we flag a potential violation of viability. Solving a LP can be performed efficiently in polynomial time [11] and real-time solvers have been pioneered for applications to model-predictive control [17]. However, as we will examine in the subsequent section, it is possible to efficiently monitor a single half-space of the geofence, while completely avoiding the LP solver.

4 Monitoring For Geofence Violations

In this section, we use the implementation of the monitoring approach from Section 3 for checking geofences for UAVs. A *geofence* is defined by a (disjoint union of) polyhedral regions over $\mathbb{R}^3$ that defines the possible (x, y, z) coordinates of an aircraft over time. Let F denote the polyhedral region. We will use a data-driven plant model $\mathcal{P}$ that is inferred from the telemetry data including positions and velocities over time, as described in Section 2. The data is updated with a small time period δ (0.4 seconds). We will choose a time horizon $N\delta$ (typically in the range $5-20$ seconds). The monitoring approach uses the following improvements on top of the base algorithm from Section 3:

1. *Monitoring Single Half-spaces:* We show that the approach in Section 3 can be simplified considerably if we can monitor one half-space at a time. This is natural for geofencing applications, wherein the safety property represents a large geographical region.
2. *Receding horizon monitoring:* We deploy N monitors $\mathcal{M}_1, \ldots, \mathcal{M}_N$ in parallel wherein $\mathcal{M}_j$ monitors the viability for time $j\delta$ into the future.

Monitoring Single Half-Spaces: We will now derive an efficient monitor when the safety property F is defined by a single half-space: $F : \{\boldsymbol{x} \mid \boldsymbol{c} \cdot \boldsymbol{x} \geq d\}$. We will also assume that $\mathcal{U}$, the bounds on the control inputs is a box with each control input $u_i \in [a_i, b_i]$. The restriction to a single half-space can be justified for geofence regions that are large enough so that if they are violated, the violation will occur by crossing a single hyperplane of the polyhedron rather than crossing the intersection of multiple regions simultaneously. We will now derive the monitoring conditions, following the same approach as in Section 3. However, we will do so for the special case when F is a single half-space.

Given $\boldsymbol{x}(N\delta) = A_N\boldsymbol{x}_0 + B_N\boldsymbol{v}_N + C_N\boldsymbol{z}_N$, we have $\boldsymbol{c}\cdot\boldsymbol{x}(N\delta) = (\boldsymbol{c}^T A_N)\boldsymbol{x}_0 + (\boldsymbol{c}^T B_N)\boldsymbol{v}_N + (\boldsymbol{c}^T C_N)\boldsymbol{z}_N$. Therefore, $\boldsymbol{c}\cdot\boldsymbol{x}(N\delta) \geq d$ if and only if there exists $\boldsymbol{v}_N \in \mathcal{V}_N$ such that the following condition holds with probability at least θ:

$$(\boldsymbol{c}^T A_N)\boldsymbol{x}_0 + (\boldsymbol{c}^T B_N)\boldsymbol{v}_N + (\boldsymbol{c}^T C_N)\boldsymbol{z}_N \geq d \tag{10}$$

Note that the disturbance term $(\boldsymbol{c}^T C_N)\boldsymbol{z}_N$ is a scalar normal random variable with 0 mean and whose standard deviation can be computed as a weighted sum of the individual standard deviations of the component random variables. Therefore, let $[-M, M]$ represent an interval such that $\mathbb{P}((\boldsymbol{c}^T C_N)\boldsymbol{z}_N \in [-M, M]) \geq \theta$. In other words, we choose a θ-robust set, that is an interval. Therefore, a sufficient condition for Eq. (10) is as follows:

$$(\exists\ \boldsymbol{v}_N \in \mathcal{V}_N)\ (\boldsymbol{c}^T B_N)\boldsymbol{v}_N \geq d + M - (\boldsymbol{c}^T A_N)\boldsymbol{x}_0 \tag{11}$$

$\boldsymbol{v}_N$ collects all the control inputs $\boldsymbol{u}(0), \ldots, \boldsymbol{u}((N-1)\delta)$. Thus, Eq. (11) is "expanded" as

$$(\exists \boldsymbol{u}(\delta) \in \mathcal{U}, \ldots, \boldsymbol{u}((N-1)\delta) \in \mathcal{U}) \sum_{j=0}^{N-1}\sum_{i=1}^{m} \hat{c}_{i,j}\boldsymbol{u}_i(j\delta) \geq \hat{d}\,, \tag{12}$$

wherein $\hat{c}_{i,j}$ represents the component of $\boldsymbol{c}^T B_N$ corresponding to the control input $\boldsymbol{u}_i(j\delta)$ (the i^{th} component of the control input at time $t = j\delta$) and $\hat{d} = d + M - (\boldsymbol{c}^T A_N)\boldsymbol{x}_0$. Note that the value of $\boldsymbol{u}_i(0)$ is known, and for $j \geq 1$, $\boldsymbol{u}_i(j\delta) \in [a_i, b_i]$. We define $\hat{u}_{i,j}$ as follows:

$$\hat{u}_{i,j} = \begin{cases} b_i & \text{if } \hat{c}_{i,j} \geq 0 \\ a_i & \text{if } \hat{c}_{i,j} < 0 \end{cases}$$

Lemma 7. *The condition* (12) *is satisfiable iff*

$$\sum_{j=1}^{N-1}\sum_{i=1}^{m} \hat{c}_{i,j}\hat{u}_{i,j} \geq \hat{d} - \sum_{i=1}^{m} \hat{c}_{i,0}\boldsymbol{u}_i(0)\,. \tag{13}$$

Proof. Note that $\hat{c}_{i,j}\boldsymbol{u}_i(j\delta) \leq \hat{c}_{i,j}\hat{u}_{i,j}$ for every i, j.

Let us assume that (12) is satisfiable. We conclude the existence of $\boldsymbol{u}_i(j\delta) \in [a_i, b_i]$ such that

$$\sum_{j=0}^{N-1}\sum_{i=1}^{m} \hat{c}_{i,j}\boldsymbol{u}_i(j\delta) \geq \hat{d}\,.$$

Therefore,

$$\sum_{j=1}^{N-1}\sum_{i=1}^{m} \hat{c}_{i,j}\boldsymbol{u}_i(j\delta) \leq \sum_{j=1}^{N-1}\sum_{i=1}^{m} \hat{c}_{i,j}\hat{u}_{i,j}$$

Therefore, we conclude that

$$\sum_{j=0}^{N-1}\sum_{i=1}^{m} \hat{c}_{i,j}\hat{u}_{i,j} \geq \hat{d} - \sum_{i=1}^{m} \hat{c}_{i,0}\boldsymbol{u}_i(0)\,.$$

Conversely, let us assume

$$\sum_{j=1}^{N-1}\sum_{i=1}^{m} \hat{c}_{i,j}\hat{u}_{i,j} \geq \hat{d} - \sum_{i=1}^{m} \hat{c}_{i,0}\boldsymbol{u}_i(0)\,.$$

We set $\boldsymbol{u}_i(j\delta) := \hat{u}_{i,j}$. This is a legal choice since $\hat{u}_{i,j} \in [a_i, b_i]$. Therefore, we have for our choice of $\boldsymbol{u}_i(j\delta)$:

$$\sum_{j=1}^{N-1}\sum_{i=1}^{m} \hat{c}_{i,j}\boldsymbol{u}_i(j\delta) \geq \hat{d} - \sum_{i=1}^{m} \hat{c}_{i,0}\boldsymbol{u}_i(0)\,.$$

In other words, (12) is satisfiable.

In other words, monitoring a single half-space can avoid using LP solvers, and instead, rely on efficient matrix vector multiplication operations.

Finding largest θ value for viability: Rather than fixing a value of θ and checking θ-viability, a simple modification to Eq. (11) allows us to find the largest value of θ for which viability can be guaranteed. To do so, we find a value of M which corresponds to the minimum possible disturbance that can continue to maintain viability. This is convenient since it allows us to compute a *risk measure* rather than a yes/no answer.

5 Evaluation

We now present a preliminary evaluation of the ideas presented, thus far, based on viability monitoring applied to telemetry data collected from a test flight of the Talon UAV running a Pixhawk autopilot [28, 21]. The test flight was carried out over the Pawnee national grassland in the USA during summer 2017 and the data recorded included GPS positions, velocities and accelerations in x, y, z directions. Note that accelerations are treated as the control inputs to our model. The Talon UAV flight data includes about 4500 seconds of flight data with data collected at $\delta = 0.4$ second intervals. We dropped the first 800 seconds that consisted of take off followed by *loitering*. The subsequent 800 seconds of data were used as the training set for inferring a data-driven model. The estimated average wind speed was about 3 m/s. However, detailed wind data was not collected for these experiments.

Data-Driven Model: We used regression to infer AR models for capturing the deviations, as explained in Section 2. The value of the *lookback* (p) was chosen to be $p = 4$, so that the overall standard deviation of the residuals was minimized. The combined model has 30 state variables that include the positions (x, y, z), velocities (v_x, v_y, v_z), and the AR model states for $e_x, e_y, e_z, e_{v_x}, e_{v_y}$, and e_{v_z}. The disturbances were taken to be normal random variable with mean and standard deviations estimated from the residual errors obtained after fitting the AR model.

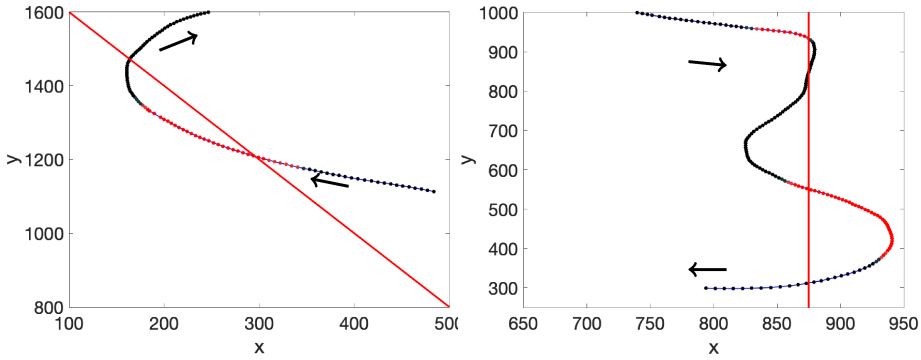

Fig. 2. Sample trajectory segments over time intervals (a) $[1600, 1680]$ seconds and (b) $[4000, 4080]$ seconds from start of flight test. Two geofence boundaries are shown as red lines. The monitor uses a time horizon $N = 15$ (6 seconds). Arrows denote the direction of the UAV's flight. Data points are shaded red if a violation results and blue/black depending on the magnitude of the viability probability $\theta(t)$.

The mean values were very close to 0, lying in the range $[-0.05, 0.05]$ in all cases, and thus taken to be exactly zero. All calculations were performed in Matlab(tm) running on a macbook pro laptop with 3.1 GHz Intel Core i7 and 16 GB RAM. The time taken to perform regression was less than 0.05 seconds. The matrices A, B, C for the plant model are sparse and thus we use sparse matrix manipulations available in Matlab(tm).

To what extent can a viability monitor be used to flag safety violations? As mentioned earlier, viability and safety are rather different. On one hand, the UAV can violate the geofence without causing a failure of viability. This is because, there may always be a N step strategy to keep the violation from happening, whereas the actual controller is unable to implement this strategy. On the other, a loss of viability does not mean that safety will be violated. After all, the environment may not have manifested its worst case behavior. Model mismatch between the linear stochastic data-driven model and the underlying nonlinear model can potentially make the issue of missed violations and false alarms much worse. We will now perform an empirical evaluation of the viability monitor, focusing on its ability to predict an impending violation as well as the false alarm rate.

Figure 2 shows two example scenarios for a fixed geofence property shown, each corresponding to roughly 80 seconds of flying time taken from our data. We defined geofence boundaries and use our monitors with $N = 15$ to check for viability. Note that in both, the viability monitor is able to provide advance warning of an impending violation (shown using red circles). However, the viability monitor differs from a safety monitor: this is clearly seen at points that are shaded blue/black even though the UAV remains in violation of the geofence.

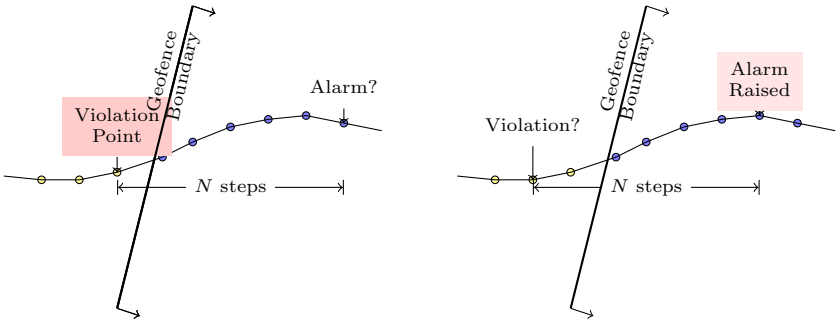

Fig. 3. (Left): For each violation point, is there an alarm raised N steps in the past for various values of N? **(Right):** For each alarm raised by the monitor, is there a violation of the geofence N steps into the future?

This is because the monitor infers the existence of a strategy for the UAV to get back into the geofence within the time horizon.

Empirical Evaluation on Randomly Generated Geofence Specifications: The empirical evaluation is carried out over segments of the data past the initial 800 seconds of data used for training. We defined various randomly generated half-spaces $c_1x + c_2y + c_3z \geq d$ as the geofences to be monitored. For each such geofence, we ran 30 monitors wherein the i^{th} monitor has its time horizon of $N = i$. First, we define *violation points* for the geofence, wherein time t is said to be a violation point iff the position at time t violates the geofence whereas the position at the previous time step $t - \delta$ satisfies the geofence specification (see Fig. 3). We analyze our data in order to answer three questions **Q1-3**, with **Q1, Q2** focusing on missed alarms whereas **Q3** focusing on alarms that do not materialize in a violation.

1. **Q1:** How far ahead of a violation point do we obtain the earliest alarm corresponding to that point?
2. **Q2:** What fraction of the violation points are alarmed by monitor with lookahead time $N = i$ for various values of $i \in [1, 30]$?
3. **Q3:** If a monitor with lookahed of i, raises an alarm at time t, does the UAV position at $t + i\delta$ violate the geofence?

We studied 250 randomly generated geofence specifications and instantiated 30 monitors for each specification with time horizons ranging from $1 - 30$. The offline calculations yield matrices $\boldsymbol{c}^T A_N$, $\boldsymbol{c}^T B_N$ and $\boldsymbol{c}^T C_N$, wherein $\boldsymbol{c}$ represents the normal vector to the hyperplane describing the geofence. The online monitor uses the calculations presented in Section 4 using lower bounds and upper bounds on the accelerations. These were taken to be $\pm 2m/s^2$ for our calculations based on the acceleration inputs observed in the actual data. For each state $\boldsymbol{x}(t)$ and control $\boldsymbol{u}(t)$, we calculate $\theta(t)$ the largest value of θ for which the property of

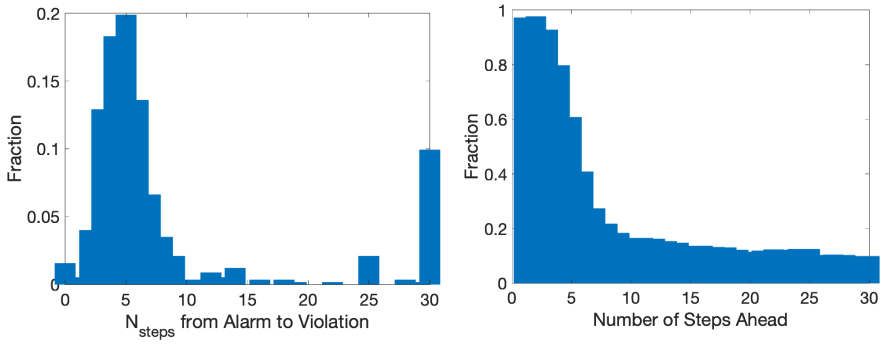

Fig. 4. (Left): Histogram showing the number of steps between a known violation event and the *earliest* alarm corresponding to the event using monitors with time horizon ranging from $1-30$. 0 steps is used to indicate that all monitors deployed did not alarm for a given violation. **(Right):** The fraction of violation points successfully predicted by a monitor looking ahead i steps for $i \in [1, 30]$.

interest can be guaranteed to be viable. We use a threshold of 0.95 for reporting violations: i.e., if $\theta(t) \leq 0.95$, we report potential violations.

Computation Times: First, we will analyze the overall computation times taken for various phases of our approach. The model construction solving a linear regression problem required 0.1 seconds using Matlab (tm) to solve the least squares problem. The use of sparse matrix computations yielded significant savings in the overall computation time. The average offline computation time required for each geofence property was 0.15 seconds. This includes the offline computation time for all the 30 monitors that were instantiated corresponding to each geofence. Likewise, the average online computation time at each time instant was 0.09 seconds for all 30 monitors. Recalling that the monitors looked ahead between $1-30$ steps with 0.4 seconds/step, these times are much smaller than the overall time horizon.

Analysis of Missed Violation Points: Figure 4 **(left)** plots the number of steps between a violation point and the *earliest* alarm corresponding to that violation. We use 0 steps to indicate that a violation point was missed by all monitors. We note that 99% of the violation points are detected at least 1 step (0.4 seconds ahead). In fact, nearly 98.5% of the violation points are detected 0.8 seconds ahead, while 65% of the violation points are detected more than 2 seconds ahead of time. At the other end, about 15% of the violation points are detected 12 seconds ahead of time. Interestingly, we note a strong correlation between violations that are predicted $15-29$ steps in advance and those predicted 30 steps in advance. In other words, most violations that are predicted 15 steps in advance are also predicted 30 steps in advance.

Figure 4 **(right)** focuses on individual monitors monitoring $1-30$ steps ahead in the future and the fraction of violation points successfully predicted by each monitor. As expected, the monitors looking ahead less than 5 steps ($\leq 2s$) are successful more than 90% of the time in predicting violation points, whereas monitors 30 steps ahead predict less than 20% of the violation points.

Overall, the analysis shows that using a bank of monitors in parallel wherein each monitor has a different lookahead time horizon can reduce the cumulative missed alarm rate to less than 1%. However, this also means that impending violations may be caught as early as 12 seconds in some cases, and as late as 0.4 seconds in advance in some cases with most alarms occuring between $2-4$ seconds ahead of a violation.

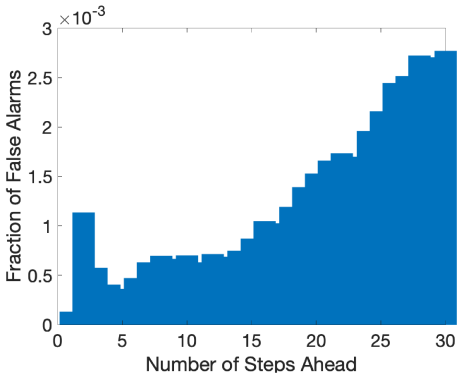

Fig. 5. Fraction of false alarms for monitor looking ahead $N = i$ steps into the future for $i \in [1, 30]$. Notice that the y-axis numbers are scaled by 10^{-3}.

Analysis for False Positives: Another key issue is that of false positives. To analyze for false positives, we focus on each alarm raised by the monitor that looks ahead $N = i$ steps into the future at time t and ask whether the UAV violates the geofence at time $t + i\delta$. Figure 5 shows what fraction of the alarms do not result in corresponding violations i steps into the future. We note that the false positive rate is quite tiny: I.e., most alarms do result in violations.

6 Conclusions

To conclude, we present the notion of θ-viability and derive sufficient conditions for monitoring whether or not a linear system driven by stochastic disturbances is θ viable at its current state. We apply this to geofence monitoring of UAVs by first building data-driven autoregressive models and then using these models to build predictive monitors for geofences. Our experimental evaluation considers data from an actual UAV flight and shows that the viability monitor can provide useful advance warnings $5-10$ seconds before a violation. Our future work will investigate strategies for model validation and updating, which is not studied in this paper. We also plan to consider multi-modal approaches wherein different modes such as loitering, turning and waypoint following are modeled differently. The integration of richer onboard events in our model and monitor along the lines of systems such as R2U2 remains an exciting avenue for future work [19].

Also, our work is currently restricted to safety properties. Efficiently monitoring viability of richer temporal properties remains an important challenge.

Acknowledgments: We are grateful to Drs. Jyotirmoy Deshmukh and Derek Kingston for valuable discussions. This work was funded in part by the US National Science Foundation (NSF) under award number 1815983, the US Airforce Research Laboratory and the NSF-IUCRC Center for Unmanned Aerial Systems (C-UAS). All ideas and opinions expressed here are those of the authors and do not necessarily represent those of NSF, AFRL or C-UAS.

References

1. Alshiekh, M., Bloem, R., Ehlers, R., Könighofer, B., Niekum, S., Topcu, U.: Safe reinforcement learning via shielding. In: Thirty-Second AAAI Conference on Artificial Intelligence (2018)
2. Althoff, M.: An introduction to CORA 2015. In: Proc. of the Workshop on Applied Verification for Continuous and Hybrid Systems. pp. 120–151 (2015)
3. BBC News: Heathrow airport: Drone sighting halts departures, bBC News 8 January 2019: Cf. https://www.bbc.com/news/uk-46803713
4. Boyd, S., Vandenberghe, S.: Convex Optimization. Cambridge University Press (2004)
5. Brockwell, P.J., Davis, R.A.: Time Series: Theory and Methods (2nd ed.). Springer, New York (2009)
6. Chati, Y.S., Balakrishnan, H.: A gaussian process regression approach to model aircraft engine fuel flow rate. In: Proceedings of the 8th International Conference on Cyber-Physical Systems. pp. 131–140. ICCPS '17 (2017)
7. Chen, X., Ábrahám, E., Sankaranarayanan, S.: Flow*: An analyzer for non-linear hybrid systems. In: Proc. of CAV'13. LNCS, vol. 8044, pp. 258–263. Springer (2013)
8. Chen, X., Sankaranarayanan, S.: Decomposed reachability analysis for nonlinear systems. In: 2016 IEEE Real-Time Systems Symposium (RTSS). pp. 13–24. IEEE Press (Nov 2016)
9. Chen, X., Ábrahám, E., Sankaranarayanan, S.: Taylor model flowpipe construction for non-linear hybrid systems. In: Proc. RTSS'12. pp. 183–192. IEEE (2012)
10. Chen, X., Sankaranarayanan, S.: Model-predictive real-time monitoring of linear systems. In: IEEE Real-Time Systems Symposium (RTSS). pp. 297–306. IEEE Press (2017)
11. Chvátal, V.: Linear Programming. Freeman (1983)
12. Duggirala, P.S., Potok, M., Mitra, S., Viswanathan, M.: C2E2: a tool for verifying annotated hybrid systems. In: Proceedings of the 18th International Conference on Hybrid Systems: Computation and Control, HSCC'15, Seattle, WA, USA, April 14-16, 2015. pp. 307–308 (2015)
13. Hoerl, A.E., Kennard, R.W.: Ridge regression: Biased estimation for nonorthogonal problems. Technometrics **12**(1), 55–67 (1970)
14. Johnson, T.T., Bak, S., Caccamo, M., Sha, L.: Real-time reachability for verified simplex design. ACM Trans. Embedd. Comput. Syst. **15**(2), 29 (May 2016)
15. Könighofer, B., Alshiekh, M., Bloem, R., Humphrey, L., Könighofer, R., Topcu, U., Wang, C.: Shield synthesis. Formal Methods in System Design **51**(2), 332–361 (Nov 2017)

16. Lygeros, J., Prandini, M.: Aircraft and weather models for probabilistic collision avoidance in air traffic control. In: Proceedings of the 41st IEEE Conference on Decision and Control, 2002. vol. 3, pp. 2427–2432 vol.3 (Dec 2002)
17. Mattingley, J., Wang, Y., Boyd, S.: Receding horizon control: Automatic generation of high-speed solvers. IEEE Control Systems Magazine **31**(3), 5265 (June 2011)
18. McLeod, A.I., Li, W.K.: Diagnostic checking arma time series models using squared-residual autocorrelations. J. Time Series Analysis **4**(4), 1467–9892 (1983)
19. Moosbrugger, P., Rozier, K.Y., Schumann, J.: R2u2: monitoring and diagnosis of security threats for unmanned aerial systems. Formal Methods in System Design (1), 31–61 (2017)
20. Phan, D., Paoletti, N., Zhang, T., Grosu, R., Smolka, S.A., Stoller, S.D.: Neural state classification for hybrid systems. In: Lahiri, S.K., Wang, C. (eds.) Automated Technology for Verification and Analysis (ATVA). pp. 422–440. Springer International Publishing (2018)
21. Pixhawk: Independent open-hardware autopilot (2018), cf. pixhawk.org accessed October 2018
22. Prandini, M., Lygeros, J., Nilim, A., Sastry, S.: Randomized algorithms for probabilistic aircraft conflict detection (02 1999)
23. Sha, L.: Using simplicity to control complexity. IEEE Software **18**(4), 20–28 (2001)
24. Stevens, M.N., Atkins, E.M.: Multi-mode guidance for an independent multicopter geofencing system. In: 16th AIAA Aviation Technology, Integration, and Operations Conference. p. 3150. AIAA (2016)
25. Stevens, M.N., Rastgoftar, H., Atkins, E.M.: Specification and evaluation of geofence boundary violation detection algorithms. In: International Conference on Unmanned Aircraft Systems (ICUAS). pp. 1588–1596. IEEE (2017)
26. Vinod, A.: Scalable Stochastic Reachability: Theory, Computation, and Control. Ph.D. thesis, University of New Mexico (2018)
27. Vinod, A.P., Gleason, J.D., Oishi, M.: SReachTools: A MATLAB Stochastic Reachability Toolbox. In: Proceedings of the International Conference on Hybrid Systems: Computation and Control. pp. 33 – 38. Montreal, Canada (April 16–18 2019), https://sreachtools.github.io
28. Watza, S.Z.: Assessment of an online rf propagation hybrid architecture for communication-aware small unmanned aircraft systems (2018)