Proceedings of the ASME 2019
Dynamic Systems and Control Conference
DSCC2019
October 8-11, 2019, Park City, Utah, USA

DSCC2019-9214

SAFE, AGGRESSIVE QUADROTOR FLIGHT VIA REACHABILITY-BASED TRAJECTORY DESIGN *

Shreyas Kousik
Mechanical Engineering
University of Michigan
Ann Arbor, Michigan 48109
Email: skousik@umich.edu

Patrick Holmes, Ram Vasudevan
Mechanical Engineering
University of Michigan
Ann Arbor, Michigan 48109
Email: pdholmes, ramv@umich.edu

ABSTRACT

Quadrotors can provide services such as infrastructure inspection and search-and-rescue, which require operating autonomously in cluttered environments. Autonomy is typically achieved with receding-horizon planning, where a short plan is executed while a new one is computed, because sensors receive limited information at any time. To ensure safety and prevent robot loss, plans must be verified as collision free despite uncertainty (e.g, tracking error). Existing spline-based planners dilate obstacles uniformly to compensate for uncertainty, which can be conservative. On the other hand, reachability-based planners can include trajectory-dependent uncertainty as a function of the planned trajectory. This work applies Reachability-based Trajectory Design (RTD) to plan quadrotor trajectories that are safe despite trajectory-dependent tracking error. This is achieved by using zonotopes in a novel way for online planning. Simulations show aggressive flight up to 5 m/s with zero crashes in 500 cluttered, randomized environments.

1 Introduction

Autonomous unmanned aerial robots, such as quadrotors, can replace humans for dangerous tasks such as infrastructure inspection and search-and-rescue, which require navigating cluttered environments. These robots are maneuverable, but often expensive and delicate. Therefore, verifying they can operate

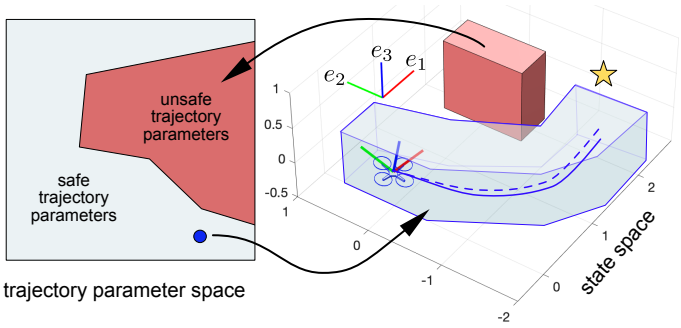

FIGURE 1: OVERVIEW OF THE PROPOSED METHOD.

safely (meaning, without collision) is important to enable them to provide such services. Such verification is difficult because state space models of aerial robots are typically nonlinear and have at least 12 states [1]. In addition, these robots typically perform receding-horizon planning, where they execute a short trajectory while planning the next one, because the robot's sensor information is limited at any time. So, the robot must plan trajectories that are verified as dynamically feasible and safe in real time. This paper plans verified trajectories for quadrotor by extending the existing Reachability-based Trajectory Design (RTD) method.

In the literature, quadrotor trajectory planners typically generate time-varying polynomial splines in position [2], [3]. Such splines have closed-form solutions for desired position, velocity, and higher derivatives, so they compute quickly [4], and one can prove that they only lie within obstacle-free space [5]. Since these splines are smooth, they can typically be tracked within

*THIS WORK IS SUPPORTED BY THE OFFICE OF NAVAL RESEARCH UNDER AWARD NUMBER N00014-18-1-2575, AND BY THE NATIONAL SCIENCE FOUNDATION AWARD #1751093

0.1 m of *tracking error* at speeds up to 8 m/s [6]; so, spline-based approaches typically treat tracking error implicitly, by dilating all obstacles by a fixed amount, which can be conservative. These methods then rely on the quadrotor's trajectory-tracking *low-level controller* to ensure the quadrotor does not crash. Since low-level controllers can compensate for aerodynamic and model disturbances [6], [7], and large orientation deviations from a reference trajectory [1], [8], these approaches have been successful at navigating unknown, cluttered environments.

However, it is unclear how these methods can be extended to incorporate *trajectory-dependent uncertainty* (such as tracking error or aerodynamic disturbance) into online planning without dilating obstacles uniformly. On the other hand, reachability-based methods address this issue by explicitly modeling trajectory-dependent uncertainty. FasTrack, for example, computes tracking error as a function of control inputs and generates a feedback controller to compensate for it, which has been demonstrated on near-hover quadrotors [9], [10]. Zonotope reachability can similarly compute tracking error, and has been shown on helicopters and cars, but requires computing a reachable set at every planning iteration, which can be too slow for real-time planning with high-dimensional system models [11], [12]. Our prior work, called Reachability-based Trajectory Design (RTD), uses a parameterized continuum of low-dimensional trajectories, and computes a Forward Reachable Set (FRS) of the trajectories plus tracking error offline with Sums-of-Squares (SOS) programming. Online, it maps obstacles to the trajectory parameter space via the FRS, then chooses a trajectory from the remaining safe set [13]. However, RTD has been only been shown for ground robots and low-dimensional models [14]–[16].

This work extends RTD using zonotope reachability to produce guaranteed-safe reference trajectories for a quadrotor despite trajectory-dependent tracking error, as shown in Figure 1. Points in the trajectory parameter space (on the left) correspond to desired trajectories (the dashed blue line on the right). The quadrotor, shown with its body-fixed coordinate frame, executes the solid blue trajectory, which has tracking error that depends on the desired trajectory. The FRS is intersected with obstacles (red box on the right) to identify unsafe trajectories (red area on the left). Then, the subset of the FRS for any safe trajectory parameter (blue tube on the right) will not intersect any obstacles. The particular desired trajectory shown attempts to reach a waypoint (gold star).

1.1 Contribution and Paper Organization

The contributions of this work are as follows. First, we extend the dynamic models used in RTD from 2D to 3D (Section 2). Second, we extend the FRS computation in RTD from SOS programming to zonotope reachability, which lets us increase the dimension of the FRS from 5 to 13 (Section 4). Third, we use the zonotope FRS to plan in real time, without recomputing the reachable set at every iteration, while adding trajectory-dependent tracking error online (Section 5). We present simulation results in Section 6 and concluding remarks in Section 7. A video is available [17]. Note that, due to space constraints, some proofs are omitted, but are available in a technical report [18].

1.2 Notation

We adopt the following notation. Variables, points, and functions are lowercase; sets and matrices are uppercase. For a point p, $\{p\}$ denotes a set containing that point as its only element. A multi-dimensional point or vector v with elements v_1 and v_2 is (v_1, v_2). Column vectors are in brackets in equations. Subscripts indicate a subspace or description; superscripts in parentheses indicate an index. An $n \times n$ identity matrix is $I_{n\times n}$. An $m \times p$ matrix of zeros is $0_{m\times p}$. An $n \times n$ matrix with diagonal elements $d_1, d_2, \cdots d_n$ is denoted $\text{diag}(d_1, d_2, \cdots, d_n)$. The positive real line is $\mathbb{R}_+ = [0, +\infty)$. The power set of A is $\mathcal{P}(A)$. Set addition is $A + B = \{a + b \mid a \in A, b \in B\}$. For a state x, its first (resp. second) time derivative is $\dot{x}$ (resp. $\ddot{x}$). Euclidean space in n dimensions is $\mathbb{R}^n$. The 3-dimensional special orthogonal group is $\mathsf{SO}(3)$, with Lie algebra $\mathfrak{so}(3)$.

Definition 1. *A* box *is a set $B(c, l, w, h) \subset \mathbb{R}^3$ defined by a center $c \in \mathbb{R}^3$ and length, width, and height $l, w, h, \in \mathbb{R}_+$. We write:*

$$B(c, l, w, h) = \{c\} + \left[-\frac{l}{2}, \frac{l}{2}\right] \times \left[-\frac{w}{2}, \frac{w}{2}\right] \times \left[-\frac{h}{2}, \frac{h}{2}\right], \qquad (1)$$

where $[\cdot, \cdot]$ is an interval on each axis of $\mathbb{R}^3$. A cube *is a box with all sides of equal length l, denoted $B(c, l)$.*

2 Dynamic Models

This section first specifies timing requirements for planning, and defines a fail-safe maneuver. Next, it introduces the high-fidelity and trajectory-producing models used to apply RTD to a quadrotor. Finally, it describes the robot's body.

RTD performs receding-horizon planning. In each planning iteration, RTD first uses a *high-fidelity model* to estimate the robot's future position while it executes the current plan. Then, RTD attempts to generate a new, safe plan starting from the future position estimate. Planning with the high-fidelity model in real time is typically prohibitively computationally intensive, so RTD instead uses a lower-dimensional *trajectory-producing model* to generate a *desired trajectory* (also called a plan).

To guarantee real-time operation, the robot must enforce a timeout on its online planning. If a new plan cannot be found within this timeout, then the robot executes its previous plan. To guarantee safe operation, RTD also requires that each desired trajectory incorporate a *fail-safe maneuver* that brings the robot to a safe state. Then, if the robot cannot plan a new trajectory while executing the previous one, it can execute the remainder

of its previous plan to come to a safe state. We formalize these requirements as follows.

Definition 2. *At each planning iteration, the robot has a* planning time *of* $t_{\text{plan}} > 0$ *in which to find a new, safe plan. If no such plan is found, the robot is required to execute a collision-free* fail-safe maneuver *that brings it to a stationary hover.*

This fail-safe maneuver is reasonable because the quadrotor can land vertically from a hover.

2.1 High-fidelity Model

We denote the *high-fidelity model* as $f_{\text{hi}} : T \times S \times U \to \mathbb{R}^{n_{\text{hi}}}$. The *planning time horizon* is $T = [t_0, t_{\text{f}}]$. Without loss of generality (WLOG), since we use receding-horizon planning, we let $t_0 = 0$ at the beginning of each planning iteration, so each planned trajectory is of duration t_{f}. The state space is $S = X \times V \times \Omega \times \mathsf{SO}(3)$ with state $s = (x, v, \omega, R)$, where $x \in X \subset \mathbb{R}^3$ is position in the inertial frame; $v \in V \subset \mathbb{R}^3$ is velocity; $\omega \in \Omega \subset \mathbb{R}^3$ is angular velocity; and $R \in \mathsf{SO}(3)$ is attitude. The inertial frame X is spanned by unit vectors denoted e_1, e_2, and e_3 with e_3 pointing "up" relative to the ground, so Re_3 is the net thrust direction of the quadrotor's body-fixed frame. We write the dynamics as per [1]:

$$\begin{aligned} \dot{x} &= v \\ \dot{v} &= \tau R e_3 - m g e_3 \\ \dot{\omega} &= J^{-1}(\mu - \omega \times J\omega) \\ \dot{R} &= R\hat{\omega}, \end{aligned} \tag{2}$$

where $\hat{\cdot} : \mathbb{R}^3 \to \mathfrak{so}(3)$ is the *hat map* that maps a 3D vector to a skew-symmetric matrix [1]. The constant g = 9.81 m/s^2 is acceleration due to gravity. The quadrotor's mass is $m \in \mathbb{R}$, and its moment of inertia matrix is $J \in \mathbb{R}^{3\times3}$. We assume J is diagonal and constant, and write $J = \text{diag}(j_1, j_2, j_3)$. The control input is $u = (\tau, \mu) \in U \subset \mathbb{R}^4$, where $\tau \in \mathbb{R}$ is net thrust and $\mu \in \mathbb{R}^3$ is the body moment; these inputs are related to rotor speeds as:

$$\begin{bmatrix} \tau \\ \mu \end{bmatrix} = \begin{bmatrix} k_\tau & k_\tau & k_\tau & k_\tau \\ 0 & k_\tau \ell & 0 & -k_\tau \ell \\ -k_\tau \ell & 0 & k_\tau \ell & 0 \\ k_\mu & -k_\mu & k_\mu & -k_\mu \end{bmatrix} \begin{bmatrix} \omega_{\text{rot},1}^2 \\ \omega_{\text{rot},2}^2 \\ \omega_{\text{rot},3}^2 \\ \omega_{\text{rot},4}^2 \end{bmatrix}, \tag{3}$$

where k_τ and k_μ are rotor parameters, ℓ is the length from quadrotor center of mass to each rotor center, and $\omega_{\text{rot},i}$ is the speed of the i^{th} rotor [1], [19].

Assumption 3. *We assume commanded inputs can be achieved instantaneously (i.e., the rotor dynamics are fast compared to* (2)*), but that rotor speed is bounded (i.e., the inputs can saturate) [1], [2], [4]. We also assume that the quadrotor has a* maximum speed $v_{\max} > 0$ *in any direction.*

We pick $v_{\max}$ = 5 m/s, since aerodynamic drag can be compensated by rotor thrust up to 6 m/s [6], [20]. Note we are not concerned with model mismatch between the high-fidelity model and a real quadrotor. However, RTD has been shown to handle model mismatch [14]. We implement (2) with the specifications of an AscTec Hummingbird [21], [22] (see Table 1).

2.2 Trajectory-Producing Model

We use a trajectory-producing model that generates desired position trajectories as polynomials in time, separately in each coordinate of X. Our approach is based on [4], but each trajectory has two piecewise polynomial segments, to include the fail-safe maneuver as in Definition 2. We first present a 1D model, then extend it to 3D. Model parameters are in Table 1.

Consider a 1D, twice-differentiable, desired position trajectory $p_{\text{des}} : T \to \mathbb{R}$, with dynamics $f_{\text{1D}} : T \times K_{\text{1D}} \to \mathbb{R}$:

$$\dot{p}_{\text{des}}(t;\kappa) = f_{\text{1D}}(t,\kappa) = \frac{c_1(t,\kappa)}{6} t^3 + \frac{c_2(t,\kappa)}{2} t^2 + \kappa_a t + \kappa_v, \tag{4}$$

where the notation $p_{\text{des}}(t;\kappa)$ indicates the trajectory parameterized by κ. We call $\kappa = (\kappa_v, \kappa_a, \kappa_{\text{pk}}) \in K_{\text{1D}} \subset \mathbb{R}^3$ a *trajectory parameter*. In particular, $\kappa_a = \ddot{p}_{\text{des}}(0)$ is the initial desired acceleration, $\kappa_v = \dot{p}_{\text{des}}(0)$ is the initial desired speed, and κ_{pk} is a desired *peak speed* to be achieved at a time $t_{\text{pk}} \in [t_{\text{plan}}, t_{\text{f}}]$. The values of c_1, c_2 are given by [4, (64)] as

$$\begin{bmatrix} c_1(t,\kappa) \\ c_2(t,\kappa) \end{bmatrix} = \frac{1}{(c_3(t))^3} \begin{bmatrix} -12 & 6c_3(t) \\ 6c_3(t) & -2(c_3(t))^2 \end{bmatrix} \begin{bmatrix} \Delta_v(t,\kappa) \\ \Delta_a(t,\kappa) \end{bmatrix}, \tag{5}$$

$$c_3(t) = \begin{cases} t_{\text{pk}} & t \in [0, t_{\text{pk}}) \\ t_{\text{f}} - t_{\text{pk}} & t \in [t_{\text{pk}}, t_{\text{f}}], \end{cases} \tag{6}$$

$$\Delta_v(t,\kappa) = \begin{cases} \kappa_{\text{pk}} - \kappa_v - \kappa_a t_{\text{pk}} & t \in [0, t_{\text{pk}}) \\ -\kappa_{\text{pk}} & t \in [t_{\text{pk}}, t_{\text{f}}], \end{cases} \tag{7}$$

$$\Delta_a(t,\kappa) = \begin{cases} -\kappa_a & t \in [0, t_{\text{pk}}) \\ 0 & t \in [t_{\text{pk}}, t_{\text{f}}]. \end{cases} \tag{8}$$

One can think of this as a time-switched model. These dynamics produce a desired position trajectory that begins at the speed κ_v with acceleration κ_a at $t = 0$. The trajectory accelerates to a speed of κ_{pk} at $t = t_{\text{pk}}$, at which point the desired acceleration is 0; the trajectory then slows down to desired speed and acceleration of 0 at $t = t_{\text{f}}$ (this is the fail-safe maneuver). Notice that c_3, Δ_v, and Δ_a are piecewise constant in t, with a jump discontinuity at t_{pk}. Therefore, c_1 and c_2 are piecewise constant in t, which makes (4) a piecewise polynomial in time. By construction, (4) and its derivative (acceleration) are continuous functions of time. Note, a desired position trajectory can be translated arbitrarily, so we assume WLOG $p_{\text{des}}(0) = 0$. Then, any desired position trajectory given by (4) is uniquely determined by κ for all $t \in T$. Note that κ_v, κ_a, and κ_{pk} lie in compact intervals $[\kappa_v^-, \kappa_v^+]$, $[\kappa_a^-, \kappa_a^+]$, and $[\kappa_{\text{pk}}^-, \kappa_{\text{pk}}^+]$, so K_{1D} is the Cartesian product of these

three intervals.

We now make a 3D *trajectory producing model* by using the dynamics (4) for each dimension, and creating a larger parameter space $K = K_{1D} \times K_{1D} \times K_{1D} \subset \mathbb{R}^9$. For a trajectory $x_{des} : T \to X$, we denote the dynamics as $f : T \times K \to \mathbb{R}^3$, so

$$x_{des}(t;k) = x_{des}(0) + \int_T f(t,k)dt, \quad f(t,k) = \begin{bmatrix} f_{1D}(t,\kappa_1) \\ f_{1D}(t,\kappa_2) \\ f_{1D}(t,\kappa_3) \end{bmatrix}, \quad (9)$$

with trajectory parameter $k = (\kappa_1, \kappa_2, \kappa_3) \in K$, where each $\kappa_i = (\kappa_{v,i}, \kappa_{a,i}, \kappa_{pk,i})$ is the peak speed, initial speed, and initial acceleration in dimension $i = 1,2,3$. As in the 1D case, WLOG we let $x_{des}(0) = 0$. For notational purposes, let $k_{pk} = (\kappa_{pk,1}, \kappa_{pk,2}, \kappa_{pk,3})$ and similarly for k_v and k_a. Then $k = (k_v, k_a, k_{pk})$ by reordering, and we denote $K = K_v \times K_a \times K_{pk}$.

By construction, (9) includes the fail-safe maneuver specified by Definition 2, and specifies what a *plan* is: at each planning iteration, the robot attempts to pick a new $k \in K$ that specifies a new desired trajectory x_{des} to begin at t_{plan}. We bound which k can be chosen at each planning iteration. First, per Assumption 3, speed is bounded: $\left\|k_{pk}\right\|_2 \le v_{max}$. Second, since k_{pk} is a desired velocity and k_v is the initial velocity, the quantity $\frac{1}{t_{pk}} \left\|k_{pk} - k_v\right\|_2$ determines an approximate desired acceleration, leading to the following definition.

Definition 4. *The* maximum desired acceleration *is* $a_{max} > 0$*. We enforce a constraint at runtime that* $\frac{1}{t_{pk}} \left\|k_{pk} - k_v\right\|_2 \le a_{max}$.

Note that acceleration due to gravity is not included in the trajectory-producing model. However, gravity is accounted for by the low-level controller we specify in Section 3.

2.3 The Robot as a Rigid Body

The high-fidelity model (2) and trajectory-producing model (9) only express the dynamics of the robot's center of mass. However, for obstacle avoidance, one must consider the robot's entire body [12], [14]. We do so as follows.

Assumption 5. *The robot is a rigid body that lies within a cube* $B_{QR} = B(x(t), w) \subset X$ *(centered at the robot's center of mass (COM) at any time with side length* w*). We assume the box does not rotate, so it is large enough to contain the robot's body at any orientation. We call this box the* body *of the robot.*

Though this is a conservative assumption, we find in Section 4 that it simplifies the computation of a reachable set for the robot's entire body, because we can first compute a reachable set of the robot's COM, then dilate the reachable set by B_{QR}. Our quadrotor has dimensions of $0.54 \times 0.54 \times 0.0855$ m^3 [21], [22], so $w = 0.54$ m. Note that $\ell = w/2$ is the distance from the COM to the center of each rotor.

TABLE 1: IMPLEMENTATION PARAMETERS.

Robot [21], [22]		**Control** [2]		**Desired Traj.** [4]	
Param.	**Value**	**Param.**	**Value**	**Param.**	**Value**
m	0.547 kg	G_x	$2.00I_{3\times3}$	t_{plan}	0.75 s
j_1, j_2	0.0033 kgm^2	G_v	$0.50I_{3\times3}$	t_{pk}	1 s
j_3	0.0058 kgm^2	G_R	$1.00I_{3\times3}$	t_f	3 s
k_τ	1.5E-7 $\frac{N}{rpm^2}$	G_ω	$0.03I_{3\times3}$	$\kappa_v^\pm$	5 m/s
k_μ	3.75E-9 $\frac{Nm}{rpm^2}$	v_{max}	5 m/s	$\kappa_a^\pm$	10 m/s^2
ℓ	0.27 m	a_{max}	3 m/s^2	$\kappa_{pk}^\pm$	5 m/s
ω_{rot}	1100–8600 rpm	d_{sense}	12 m		

3 Tracking Error

This section describes the low-level controller used to track desired trajectories, then defines tracking error as a set-valued, trajectory-dependent *tracking error function* g. The purpose of g is to include tracking error explicitly in the quadrotor's FRS (computed in Section 4) for online planning (Section 5).

Given any $k \in K$, the quadrotor uses a feedback controller $u_k : T \times S \to U$ to track the trajectory parameterized by K. For short, we say that u_k *tracks* k. This feedback controller can take any form, such as PID, LQR, or MPC; in this work, we use the PD controller specified in [2, Section IV]. Recall that the quadrotor has states $s = (x, v, \omega, R)$. Consider a twice-differentiable desired position trajectory $x_{des} : T \to \mathbb{R}$ as in (9). Using the notation in [2], we specify a desired yaw $\psi(t) = 0$. Then, $u(t)$ is uniquely determined by the current state $s(t)$, and the desired trajectory $x_{des}(t)$ and its derivatives, by leveraging differential flatness of the model (2) [2], [6]. At any time t, the state error used for feedback is

$$\begin{aligned} e_x(t) &= x(t) - x_{des}(t) \\ e_v(t) &= v(t) - \dot{x}_{des}(t) \\ e_R(t) &= \frac{1}{2}\left(R_{des}(t)^\top R(t) - R(t)^\top R_{des}(t)\right)^\vee \\ e_\omega(t) &= \omega(t) - \omega_{des}(t), \end{aligned} \quad (10)$$

where $(\cdot)^\vee : \mathfrak{so}(3) \to \mathbb{R}^3$ is the *vee map* that maps a skew-symmetric matrix to a 3D vector [1]. The desired control input $u_k(t, s(t)) = (\tau(t), \mu(t))$ is given by

$$\begin{aligned} \tau(t) &= \left\|-G_x e_x(t) - G_v e_v(t) + mge_3 + m\ddot{x}_{des}(t)\right\|_2 \\ \mu(t) &= -G_\omega e_\omega(t) - G_R e_r(t) \end{aligned} \quad (11)$$

where R_{des} is found as in [2, Section IV] and ω_{des} is found as in [2, Section III]. In simulation, τ and μ are converted to rotor speeds and saturated using (3). The feedback gains and rotor speed saturation parameters are reported in Table 1.

Using the controller in (11), the quadrotor described by (2) cannot perfectly track trajectories produced by (9). We call the

position error term e_x from (10) the *tracking error*. As shown in the literature, RTD can bound tracking error and incorporate it into a robot's FRS, which can then be used to plan safe trajectories [14], [15]. Doing so requires the following assumption.

Assumption 6. *The sets $T, S,$ and K are compact. The high-fidelity model* (2) *is Lipschitz continuous in $t, s,$ and u.*

Also notice that the desired position trajectory produced by (9) is Lipschitz continuous in t and k because it is a piecewise polynomial on a compact domain. Let $\text{proj}_X : S \to X$ project points from S to X via the identity relation. Now, we treat the tracking error as follows.

Assumption 7. *Suppose $s_0 \in S$ is an initial condition for* (2) *such that* $\text{proj}_X(s_0) = 0$*. Let $s : T \to S$ be a trajectory of* (2) *beginning from s_0. Let $t \in T$, and let $k \in K$ be arbitrary but obeying Definition 4. Let $x_{\text{des}} : T \to X$ be a trajectory of* (9) *and recall $x_{\text{des}}(0) = 0$ WLOG. We assume there exists a set-valued* tracking error function $g : T \times K \to \mathcal{P}(\mathbb{R}^3)$ *for which*

$$\text{proj}_X(s(t; s_0, k)) \in \{x_{\text{des}}(t; k)\} + g(t, k). \tag{12}$$

We assume every $g(t, k)$ is compact.

Note that $g(t, k)$ being compact is reasonable since K is compact and the dynamics (2) are continuous, so the quadrotor cannot diverge infinitely far from any desired trajectory. By Assumption 7, we can dilate any desired trajectory with g to check if the high-fidelity model can collide with obstacles in X. Computing g is difficult for nonlinear systems such as (2) with more than 6 dimensions [9], [14]. We approximate g via sampling [18].

4 Reachability Analysis

To produce safe trajectory plans, RTD first performs an offline FRS computation, then uses the FRS for online planning. This section explains the offline FRS computation. Given the time horizon T, we define the *exact FRS* $F \subset X \times K$ as all points in X that are reachable by the trajectory-producing model (9) plus tracking error:

$$\begin{aligned} F = \Big\{(x, k) \in X \times K \mid \exists\, t \in T \text{ s.t. } x \in \{\tilde{x}(t)\} + g(t, k) \\ \text{and } \dot{\tilde{x}}(t, k) = f(t, k)\Big\}. \end{aligned} \tag{13}$$

Prior work on RTD used SOS programming to compute the FRS [13]–[16]. In this work, we instead use zonotope reachability via the CORA toolbox [23] for FRS computation, because we found it performs well for the 13D trajectory-producing model (9). This section describes how to use CORA to find a *computed FRS* $\tilde{F}$ that conservatively approximates the FRS of the trajectory-producing model (9) [24]:

$$\tilde{F} \supseteq \Big\{(x, k) \in X \times K \mid \exists\, t \in T \text{ s.t. } x(t) = \tilde{x}(t) \tag{14}$$

$$\text{and } \dot{\tilde{x}}(t, k) = f(t, k)\Big\}. \tag{15}$$

Then, per Assumption 7, if the tracking error function g is added to the computed FRS $\tilde{F}$, one can conservatively overapproximate the exact FRS F in (13). Conservatism of the computed FRS is necessary to ensure safety; if no trajectories of the computed FRS plus tracking error hit an obstacle, then no trajectories of the exact FRS can hit that obstacle. Further, since the desired trajectories are only for the center of mass of the robot, we add the size of the robot's body B_{QR} to $\tilde{F}$ as well. In this section, we compute $\tilde{F}$. We add g and B_{QR} in Section 5.

CORA represents sets using zonotopes. A zonotope Z is a polytope in $\mathbb{R}^n$ that is closed under linear maps and Minkowski sums [23], and is parameterized by its center $c \in \mathbb{R}^n$ and generators $g^{(1)}, \ldots g^{(p)} \in \mathbb{R}^n$. A zonotope describes the set of points that can be written as the center c plus a linear combination of the generators, where the coefficient $\beta^{(i)}$ on each generator must be between -1 and 1:

$$Z = \left\{ y \in \mathbb{R}^n \mid y = c + \sum_{i=1}^{p} \beta^{(i)} g^{(i)},\ -1 \le \beta^{(i)} \le 1 \right\} \tag{16}$$

For convenience, we concatenate the generators into an $n \times p$ *generator matrix* G, and the coefficients into a *coefficient vector* β:

$$G = \left[g^{(1)}, g^{(2)}, \ldots, g^{(p)} \right] \text{ and } \beta = [\beta^{(1)}, \beta^{(2)}, \cdots, \beta^{(p)}]^\top. \tag{17}$$

We can then rewrite (16) as

$$Z = \{ y \in \mathbb{R}^n \mid y = c + G\beta,\ -1 \le \beta \le 1 \}, \tag{18}$$

where $\ge$ and $\le$ are applied elementwise. From here on, for brevity we will assume that $\beta \in [-1, 1]$, and will write the constraints explicitly when this is not true.

Boxes can be exactly represented as zonotopes:

$$B(c, l, w, h) = \left\{ y \in \mathbb{R}^3 \mid\ y = c + \text{diag}\left(\frac{l}{2}, \frac{w}{2}, \frac{h}{2}\right)\beta \right\} \tag{19}$$

where c, l, w, and h refer to the center, length, width, and height (Definition 1). From here on, boxes and their zonotope representations are used interchangeably. We define addition of a zonotope $Z \subset \mathbb{R}^3 \times \mathbb{R}^n$ with a box $B \subset \mathbb{R}^3$ as follows:

$$Z + B = Z + \left\{ y \in \mathbb{R}^3 \times \mathbb{R}^n \mid y = \begin{bmatrix} y_B \\ 0_{1 \times n} \end{bmatrix}, y_B \in B \right\}. \tag{20}$$

This is useful, e.g., when a zonotope is defined over the position and parameter spaces, but we want to add a box defined only in position space (note, this assumes WLOG that the first three rows of the zonotope Z's center c and generator matrix G correspond

to the quadrotor's position coordinates).

4.1 Implementation

Since the 3D trajectory-producing model uses the same 1D dynamics (4) separately in each dimension, we begin by computing the FRS for the 1D trajectory-producing model. Then, we combine three 1D FRSes to create a single 3D FRS.

Recall that the 1D trajectory-producing model's position $p_{\text{des}}(t,\kappa)$ depends only on time and the trajectory parameter $\kappa = (\kappa_v, \kappa_a, \kappa_{\text{pk}}) \in K_{\text{1D}} \subset \mathbb{R}^3$. Therefore, we want to compute the set of all positions that can be reached given a time interval T and parameter set K_{1D}.

CORA represents the FRS as a zonotope at each of a finite collection of compact *time steps* that are intervals in T. With a minor abuse of notation, we let t act as an index, so that $Z_{\text{1D}}^{(t)}$ denotes the zonotope describing the 1D FRS over the time step containing t. Note that each $Z_{\text{1D}}^{(t)}$ is a subset of the 1D position space X_i (where $i = 1,2,3$) and parameters K_{1D}: $Z_{\text{1D}}^{(t)} \subseteq X_i \times K_{\text{1D}}$. CORA works by first linearizing the system dynamics at the beginning of each time step about the center of $Z_{\text{1D}}^{(t)}$, and obtaining the zonotope for the next time step by multiplying $Z_{\text{1D}}^{(t)}$ by an over-approximation of the matrix exponential over that time step. CORA also accounts for linearization error; since the trajectory producing quadrotor model (4) does not depend on state, this error remains small in practice. The 1D computed FRS $\tilde{F}_{\text{1D}} \subseteq X_i \times K_{\text{1D}}$, $i = 1,2,3$, is the union of the zonotopes defined at each time step: $\tilde{F}_{\text{1D}} = \bigcup_{t\in T} Z_{\text{1D}}^{(t)}$.

CORA requires specifying an initial set and dynamics to compute the FRS. We treat $\kappa \in K_{\text{1D}}$ as states, and define the initial set $Z_{\text{1D}}^{(0)}$ as:

$$Z_{\text{1D}}^{(0)} = \left\{ y \in \mathbb{R}^4 \mid y = 0_{4\times 1} + \text{diag}(0, \kappa_v^+, \kappa_a^+, \kappa_{\text{pk}}^+)\beta \right\}, \tag{21}$$

where the first dimension is position, so initial position $p_{\text{des}}(0)$ is at the origin WLOG. The dynamics given to CORA are $\dot{p}_{\text{des}}$ as in (4), and $\dot{\kappa}_{\text{pk}}, \dot{\kappa}_v, \dot{\kappa}_a = 0$ (since parameters are constant over a trajectory).

As described in (9), 3D trajectories of the trajectory-producing model can be constructed by concatenating 1D trajectories. A similar process is followed to construct the 3D FRS $\tilde{F}$ by concatenating 1D FRSes $\tilde{F}_{\text{1D}}$. Specifically, given $Z_{\text{1D}}^{(t)}$ with center $c_{\text{1D}}^{(t)}$ and generator matrix $G_{\text{1D}}^{(t)}$, we construct $Z^{(t)}$ as:

$$Z^{(t)} = \left\{ y \in \mathbb{R}^{12} \;\middle|\; y = \begin{bmatrix} c_{\text{1D}}^{(t)} \\ c_{\text{1D}}^{(t)} \\ c_{\text{1D}}^{(t)} \end{bmatrix} + \begin{bmatrix} G_{\text{1D}}^{(t)} & 0_{4\times p} & 0_{4\times p} \\ 0_{4\times p} & G_{\text{1D}}^{(t)} & 0_{4\times p} \\ 0_{4\times p} & 0_{4\times p} & G_{\text{1D}}^{(t)} \end{bmatrix} \beta \right\}, \tag{22}$$

where p is the number of generators in the matrix $G_{\text{1D}}^{(t)}$. The 3D FRS $\tilde{F} \subseteq X \times K_v \times K_a \times K_{\text{pk}}$ of the quadrotor's COM position and control parameters is then the union of $Z^{(t)}$ through time: $\tilde{F} = \bigcup_{t\in T} Z^{(t)}$. Notice that $\tilde{F}$ represents a continuum of initial conditions of (9) since it is defined over K_v and K_a. Next, we discuss how to use $\tilde{F}$ and the tracking error function g for online planning.

5 Online Planning

RTD plans trajectories online by intersecting the FRS $\tilde{F}$ with obstacles to identify safe desired trajectories, then optimizes over this set to fulfill an arbitrary cost function (e.g., minimize distance to a waypoint, desired acceleration, power usage) [13]–[16]. Past implementations of RTD used polynomial superlevel sets to represent the FRS, and were required to incorporate tracking error in the offline computation, so the intersection of the FRS with obstacles implicitly accounted for tracking error. In contrast, this section details how to add tracking error to the FRS online. We also discuss obstacle representation, and the optimization program solved at each receding horizon iteration.

5.1 The Online Planning Algorithm

RTD plans trajectories in a receding horizon way by running Algorithm 1 at each planning iteration. At the beginning of each planning iteration, at time $t = 0$ WLOG, the parameters k_v and k_a are set as the estimated velocity and acceleration of the high-fidelity model (2):

$$k_v = v(0), \quad k_a = \tau(0)R(0)e_3 - mge_3. \tag{23}$$

These are found by forwarding-integrating the model given the previous plan for a duration of t_{plan}.

For this discussion, suppose we have g as in Assumption 7. The tracking error associated with the initial condition is added to $\tilde{F}$ to make it a conservative approximation of the exact FRS. RTD attempts to find a safe trajectory by optimizing over the set of safe parameters. These safe trajectory parameters are found by taking the complement of the intersection of the FRS $\tilde{F}$ with the sensed obstacles. A limited amount of time t_{plan} is specified within which RTD attempts to choose the trajectory to be followed in the next iteration. If no trajectory is found in time, the quadrotor continues executing its previous trajectory, which brings it to a stationary hover as per (4). A single iteration of the online planning algorithm is summarized in Algorithm 1. Each step of the algorithm is explained in this section.

In this work, we represent obstacles as boxes in 3D:

Definition 8. *An obstacle $O \subset X$ is a box as in Definition 1, with center $c \in X$, and length, width, and height $l, w, h \in \mathbb{R}_+$. Obstacles are static with respect to time.*

This is not a restrictive definition, since obstacles are typically represented as occupancy grids composed of boxes [5]. When moving through the world, we assume that a quadrotor has a limited range over which it can sense obstacles. We refer to this as the quadrotor's *sensor horizon* d_{sense}.

Algorithm 1 A Single Planning Iteration (Online)

Require: g as in Assum. 7, $\tilde{F}$ as in (15), d_{sense} as in Assumption 9, s_0 as in (2), and cost function $J : K \to \mathbb{R}$, previous plan $x_{\text{prev}} : T \to S$, k_v, k_a as in (23), $A \leftarrow \emptyset$, $b \leftarrow \emptyset$
$\mathscr{O} \leftarrow$ `SenseObstacles`(x_0, d_{sense})
$\tilde{F}_\epsilon \leftarrow$ `AddTrackingError`$\left(\tilde{F}, g, k_v\right)$ // error-augmented FRS
For: $Z_\epsilon^{(t)} \in \tilde{F}_\epsilon$ // for each zonotope in error-aug. FRS slice
  For: $O^{(j)} \in \mathscr{O}$
    $K_{\text{pk,u}}^{(t,j)} \leftarrow$ `IntersectObsWithFRS`$\left(Z_{\epsilon,\sigma}^{(t)}, O^{(j)}\right)$
    $\left(A^{(t,j)}, b^{(t,j)}\right) \leftarrow$ `GenerateConstraints`$\left(K_{\text{pk,u}}^{(t,j)}\right)$
    Concatenate $(A, b) \leftarrow [A; A^{(t,j)}]$, $[b; b^{(t,j)}]$
  End
End
$x_{\text{des}} \leftarrow$ `OptimizeTrajectory`$\left(J, A, b, k_v, k_a, t_{\text{plan}}, x_{\text{prev}}\right)$
Return x_{des}

Assumption 9. *At any time t, an obstacle is considered to be sensed if any point x_{obs} of the obstacle O is within the sensor horizon from the quadrotor's COM position x:*

$$\|x(t) - x_{\text{obs}}\|_2 \le d_{\text{sense}} \quad \forall x_{\text{obs}} \in O. \tag{24}$$

Note, to ensure safety, the sensor horizon d_{sense} must be larger than the distance traveled by the longest desired trajectory plus $v_{\max}$ times t_{plan} [13, Theorem 35].

Let $O^{(j)} \subset X$ denote the j^{th} sensed obstacle, whose position is given relative the quadrotor's current position, and $n_{\mathscr{O}}$ be the number of obstacles within the quadrotor's sensor horizon. Finally, let $X_{\text{obs}} \subset X$ represent the union of all sensed obstacles: $X_{\text{obs}} = \bigcup_{j \in n_{\mathscr{O}}} O^{(j)}$.

After sensing obstacles, tracking error must be included in the FRS to identify safe trajectories of the high fidelity model. Recall g is as in Assumption 7. For any t, k, we first overapproximate the tracking error $g(t,k)$ by a box. Let $\text{box}(\cdot) : \mathcal{P}(\mathbb{R}^3) \to \mathcal{P}(\mathbb{R}^3)$ overapproximate a bounded set of 3D positions with a box. We add a tracking error box to each zonotope $Z^{(t)}$ comprising the FRS $\tilde{F}$ to obtain the *error-augmented FRS* $\tilde{F}_\epsilon$. We also add the box B_{QR} representing the quadrotor's body:

$$Z_\epsilon^{(t)} = Z^{(t)} + \text{box}(g(t,k)) + B_{\text{QR}} \tag{25}$$

$$\tilde{F}_\epsilon = \bigcup_{t \in T} Z_\epsilon^{(t)}. \tag{26}$$

Now, we identify unsafe trajectories. Recall that k_v and k_a are specified by the quadrotor's state at the beginning of each planning iteration, so online planning is performed over the peak speeds K_{pk}. We intersect obstacles $O^{(j)}$ with the error-augmented FRS $\tilde{F}_\epsilon$ to identify the unsafe set $K_{\text{pk,u}} \subset K_{\text{pk}}$ that could cause a collision with an obstacle. A peak speed k_{pk} is *unsafe* if the position dimensions of $\tilde{F}_\epsilon$ associated with k_{pk} intersect an obstacle. Here, we detail how obstacles are intersected with $\tilde{F}_\epsilon$

Notice that $\tilde{F}_\epsilon \subset X \times K_v \times K_a \times K_{\text{pk}}$ is defined over a continuum of positions and parameters. Recall that X_{obs} represents all sensed obstacle positions, and k_v and k_a are set as the initial velocity and acceleration of the quadrotor in the current planning step. We obtain the *unsafe subset* $\tilde{F}_{\text{u}}$ by intersecting $\tilde{F}_\epsilon$ with the obstacles and initial condition:

$$\tilde{F}_{\text{u}} = \tilde{F}_\epsilon \bigcap X_{\text{obs}} \times \{k_v\} \times \{k_a\} \times K_{\text{pk}} \tag{27}$$

The set of unsafe trajectory parameters $K_{\text{pk,u}}$ is the projection of $\tilde{F}_{\text{u}}$ onto the K_{pk} subspace: $K_{\text{pk,u}} = \text{proj}_{K_{\text{pk}}}(\tilde{F}_{\text{u}})$, where $\text{proj}_{K_{\text{pk}}} : \mathcal{P}(X \times K) \to \mathcal{P}(K_{\text{pk}})$ projects sets via the identity relation.

The final step in online planning is trajectory optimization. Let $J : K \to \mathbb{R}$ be an arbitrary cost function. Recall that, by Assumption 3, the quadrotor has a max speed; and, by Definition 4, a maximum acceleration. These become the following constraints on k_{pk}:

$$\frac{1}{t_{\text{pk}}} \left\| k_{\text{pk}}^{(n)} - k_v^{(n)} \right\|_2 \le a_{\max} \quad \text{and} \quad \left\| k_{\text{pk}}^{(n)} \right\|_2 \le v_{\max}. \tag{28}$$

Then, online, we find

$$k_{\text{pk}}^* = \text{argmin}_{k_{\text{pk}}} \left\{ J(k) \mid k_{\text{pk}} \notin K_{\text{pk,u}} \text{ and feas. to (28)} \right\}, \tag{29}$$

where $k = (k_v, k_a, k_{\text{pk}})$. Suppose that adding tracking error to the FRS, intersecting the FRS with obstacles, and solving (29) all complete executing within t_{plan}. Then, we return a desired trajectory x_{des} as in (9) parameterized by $(k_v, k_a, k_{\text{pk}}^*)$. Otherwise, we continue executing the previously-found trajectory (which includes a fail-safe maneuver).

Now, we formalize that Algorithm 1 is safe.

Theorem 10. *Suppose the quadrotor is described by* (2) *as in Assumption 3. Suppose $g : T \times K \to \mathcal{P}(\mathbb{R}^3)$ is as in Assumption 7. Suppose the FRS $\tilde{F}$ is found as in* (15)*. Suppose WLOG $t = 0$ and that the quadrotor is initially safe in a stationary hover. Then, if the quadrotor plans in a receding-horizon way using Algorithm 1, it is safe for all time.*

Proof. This theorem follows from the conservative definitions of $\tilde{F}$ and g, and from the fact that any planned trajectory contains a fail-safe maneuver. In other words, by construction, the quadrotor always either executes a safe trajectory, or maintains a stationary hover. ∎

Theorem 10 is stated briefly to summarize how RTD either constructs safe plans or commands the robot to execute a fail-safe maneuver, by relying on conservatism. For a more detailed treatment, see [14, Remark 70].

5.2 Implementation

We implement Algorithm 1 as follows. Note that the proofs of the lemmas and theorems in this section are available in the

extended technical report [18].

Obstacles (as in Definition 8 and Assumption 9) are given by `SenseObstacles` (Line 2). For our choice of $v_{\max}$, t_{pk}, and t_{f} (see Table 1), we find that $d_{\text{sense}} = 12$ m is sufficient [13, Theorem 35]. To add tracking error to the FRS recall that we approximate g with sampling Section 3. We implement `AddTrackingError` (Line 3) using (25) as written, with g approximated by sampling.

The intersection in (27) to obtain $\tilde{F}_{\text{u}}$ is implemented as `IntersectObsWithFRS` (Line 6) by intersecting each zonotope $Z_\epsilon^{(t)}$ comprising $\tilde{F}_\epsilon$ with each obstacle $O^{(j)}$. We find that for the quadrotor model and obstacle representations we have chosen, the intersection in (27) can be computed exactly. This requires the use of the following lemma regarding the structure of $Z_\epsilon^{(t)}$.

Lemma 11. *The zonotope $Z_\epsilon^{(t)}$ can be written as*

$$Z_\epsilon^{(t)} = \left\{ y \in \mathbb{R}^{12} \;\middle|\; y = \begin{bmatrix} c_{\epsilon,1}^{(t)} \\ c_{\epsilon,2}^{(t)} \\ c_{\epsilon,3}^{(t)} \end{bmatrix} + \begin{bmatrix} G_{\epsilon,1}^{(t)} & 0_{4\times4} & 0_{4\times4} \\ 0_{4\times4} & G_{\epsilon,2}^{(t)} & 0_{4\times4} \\ 0_{4\times4} & 0_{4\times4} & G_{\epsilon,3}^{(t)} \end{bmatrix} \beta \right\} \tag{30}$$

where $c_{\epsilon,i}^{(t)}$ and $G_{\epsilon,i}^{(t)}$ take the form

$$c_{\epsilon,i}^{(t)} = \begin{bmatrix} c_x \\ c_v \\ c_a \\ c_{\text{pk}} \end{bmatrix}, \qquad G_{\epsilon,i}^{(t)} = \begin{bmatrix} \gamma_{x,v} & \gamma_{x,a} & \gamma_{x,\text{pk}} & \epsilon_i^{(t)} \\ \gamma_v & 0 & 0 & 0 \\ 0 & \gamma_a & 0 & 0 \\ 0 & 0 & \gamma_{\text{pk}} & 0 \end{bmatrix} \tag{31}$$

where each $c, \gamma \in \mathbb{R}$, $\gamma_v, \gamma_a, \gamma_{\text{pk}}$ are nonzero, $\epsilon_i^{(t)} \in \mathbb{R}_+$ and $i = 1, 2, 3$.

Proof. See the technical report [18]. ∎

Next, we describe how to find the unsafe set of control parameters $K_{\text{pk,u}}^{(t,j)}$ given a zonotope $Z_\epsilon^{(t)}$, obstacle $O^{(j)}$, and initial velocity and acceleration k_v and k_a. Let $\text{proj}_{X_i} : \mathcal{P}(X) \to \mathcal{P}(X_i)$ project sets from X into the i^{th} position dimension via the identity relation.

Theorem 12. *Given $O^{(j)}$, let $[x_{\text{obs}}^-, x_{\text{obs}}^+] = \text{proj}_{X_i}(O^{(j)})$ be the interval that is the projection of the obstacle onto the i^{th} position dimension. Given time t and obstacle $O^{(j)}$, the set of unsafe control parameters in the i^{th} dimension $K_{\text{pk,u,i}}^{(t,j)}$ that could cause a collision with that obstacle at that time is given by*

$$K_{\text{pk,u,i}}^{(t,j)} = \begin{cases} [\gamma_{\text{pk}}\beta_{\text{pk}}^{\min}, \gamma_{\text{pk}}\beta_{\text{pk}}^{\max}], & \text{if } \beta_{\text{pk}}^- \le 1 \text{ and } \beta_{\text{pk}}^+ \ge -1 \\ \emptyset, & \text{otherwise,} \end{cases} \tag{32}$$

where

$$\beta_{\text{pk}}^- = \frac{1}{\gamma_{x,\text{pk}}} \left(x_{\text{obs}}^- - \epsilon_i^{(t)} - c_x - \gamma_{x,v} \frac{\kappa_{v,i} - c_v}{\gamma_v} - \gamma_{x,a} \frac{\kappa_{a,i} - c_a}{\gamma_a} \right), \tag{33}$$

$$\beta_{\text{pk}}^+ = \frac{1}{\gamma_{x,\text{pk}}} \left(x_{\text{obs}}^+ + \epsilon_i^{(t)} - c_x - \gamma_{x,v} \frac{\kappa_{v,i} - c_v}{\gamma_v} - \gamma_{x,a} \frac{\kappa_{a,i} - c_a}{\gamma_a} \right), \tag{34}$$

and

$$\beta_{\text{pk}}^{\min} = \min(\beta_{\text{pk}}^-, -1), \quad \beta_{\text{pk}}^{\max} = \max(\beta_{\text{pk}}^+, 1). \tag{35}$$

The set of unsafe control parameters is then the box

$$K_{\text{pk,u}}^{(t,j)} = K_{\text{pk,u},1}^{(t,j)} \times K_{\text{pk,u},2}^{(t,j)} \times K_{\text{pk,u},3}^{(t,j)} \tag{36}$$

Proof. See the technical report [18]. ∎

The set of unsafe trajectory parameters $K_{\text{pk,u}}$ is then given by the union of each time and each obstacle's unsafe parameters:

$$K_{\text{pk,u}} = \bigcup_{t \in T, j \in n_{\mathcal{O}}} K_{\text{pk,u}}^{(t,j)}. \tag{37}$$

The function `GenerateConstraints` (Line 7) represents each unsafe set $K_{\text{pk,u}}^{(t,j)}$ as constraints for trajectory optimization. Recall that each $K_{\text{pk,u}}^{(t,j)} \subseteq K_{\text{pk}}$ as in (36) is a 3-dimensional interval, which can therefore be represented as a box. Let $c = (c_1, c_2, c_3) \in \mathbb{R}^3$ be the center of $K_{\text{pk,u}}^{(t,j)}$, and $l, w, h \in R$ be the length, width and height of $K_{\text{pk,u}}^{(t,j)}$. We now discuss how to generate constraints to check whether $k_{\text{pk}} \in K_{\text{pk}}$ is contained in $K_{\text{pk,u}}^{(t,j)}$.

Theorem 13. *Given $k_{\text{pk}} \in K_{\text{pk}}$, we can check if it is in $K_{\text{pk,u}}^{(t,j)}$ with:*

$$\min(A^{(t,j)} k_{\text{pk}} + b^{(t,j)}) < 0 \implies k_{\text{pk}} \notin K_{\text{pk,u}}^{(t,j)} \tag{38}$$

where the min *is taken over the elements of its argument, and $A^{(t,j)}$ and $b^{(t,j)}$ are:*

$$A^{(t,j)} = \begin{bmatrix} 1 & 0 & 0 \\ -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & -1 \end{bmatrix}, \quad b^{(t,j)} = \begin{bmatrix} -c_1 + \frac{l}{2} \\ c_1 + \frac{l}{2} \\ -c_2 + \frac{w}{2} \\ c_2 + \frac{w}{2} \\ -c_3 + \frac{h}{2} \\ c_3 + \frac{h}{2} \end{bmatrix} \tag{39}$$

Proof. See the technical report [18]. ∎

Each $A^{(t,j)}$ and $b^{(t,j)}$ are concatenated (`Concatenate`, Line 8) into a single A and b, so that the constraints for all obstacles can be efficiently checked with matrix operations.

The final step in online planning is trajectory optimization (Line 11), wherein we solve (29). This requires optimizing over K_{pk}, which is 3D, so `OptimizeTrajectory` uses brute force

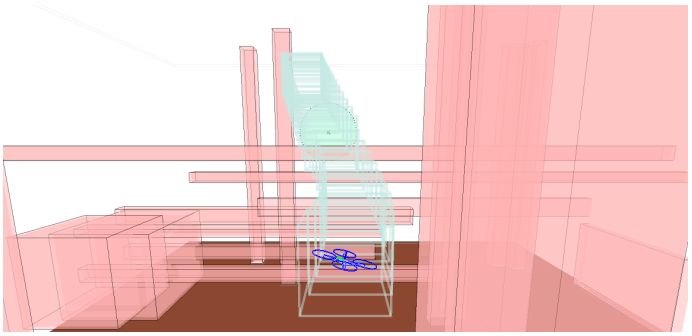

FIGURE 2: EXAMPLE TRAJECTORY PLAN.

sampling. It generates a ball of approximately 10,000 samples in K_{pk}, and evaluates the constraints (38) and (28) on the samples (this takes 50–150 ms). It eliminates all infeasible samples, then evaluates an arbitrary cost function $J : K_{\text{pk}} \to \mathbb{R}$ on the remaining samples. The sampled point with the lowest cost is denoted k^*_{pk}, which defines a new safe trajectory x_{des} given by (9) with $k = (k_v, k_a, k^*_{\text{pk}})$. If no feasible k^*_{pk} can be found within t_{plan}, the quadrotor continues executing the previous trajectory, which ends in a stationary hover.

6 Results

All simulations, along with the tracking error and zonotope reachability computations, are performed in MATLAB. We simulate an AscTec Hummingbird quadrotor [21] at up to 5 m/s. The system parameters are given in Table 1. A video is available [17].

We simulated 500 cluttered worlds with 120 random obstacles each. Each world is $80 \times 20 \times 10$ m, with a random start location at one end and random goal at the other. Note that the simulation environment performs collision checking of the body of the quadrotor with obstacles separately from how we generate and enforce constraints in Algorithm 1. At each planning iteration, the quadrotor senses obstacles within a 12 m sensor horizon as in Assumption 9, along with the world boundaries as obstacles. The quadrotor is given t_{plan} s to run Algorithm 1 at each iteration, meaning we enforce real time planning.

We ran two different implementations of Algorithm 1: one with a constant tracking error of 0.1 m, and one with trajectory-dependent tracking error computed as in Section 3. The distance 0.1 m is the maximum tracking error found in any direction from sampling tracking error to approximate g. The cost function used at each planning iteration is to minimize the distance between the quadrotor and a waypoint at the time t_{pk}; the waypoint is placed 5 m ahead of the quadrotor along a straight line between the robot and the global goal. Note that this choice of waypoint is deliberate to force the quadrotor into situations where it has to execute a fail-safe maneuver.

An example planning iteration is shown in Figures 2 and 3 with a trajectory (dark blue) planned and executed with RTD. Obstacles are in light red and the ground is brown. The global desired goal is a green sphere. The tube of light blue boxes, which does not intersect any obstacles, is the subset of the zonotope FRS for the current plan plus tracking error, so the quadrotor (in dark blue) is guaranteed to fly within the tube.

The quadrotor never crashed. With constant tracking error of 0.1 m, it reached the goal in 84.8% of trials. With trajectory-dependent tracking error, it reached the goal in 91.2 % of trials. Note, we did not expect 100% of goals reached, since the trials used randomly-generated obstacles and thus may have no feasible path from start to goal. This result confirms that including trajectory-dependent tracking error reduces conservatism.

7 Conclusion

We propose Reachability-based Trajectory Design (RTD) as a method for enabling autonomous quadrotors to plan aggressive, safe trajectories in unforeseen, cluttered environments. This work extends RTD to a 13D system with zonotope reachability and introduces a novel method to use zonotopes for safe planning online. The proposed method was tested in 500 simulations in random cluttered environments at speeds up to 5 m/s, with zero crashes. In future work, we will implement RTD on hardware, and explore more types of trajectory-dependent uncertainty.

References

[1] T. Lee, M. Leok, and N. McClamroch, "Control of Complex Maneuvers for a Quadrotor UAV using Geometric Methods on SE(3)," *arXiv e-prints*, arXiv:1003.2005, arXiv:1003.2005, Mar. 2010, View online.

[2] D. Mellinger and V. Kumar, "Minimum snap trajectory generation and control for quadrotors," in *2011 IEEE International Conference on Robotics and Automation*, View online., May 2011, pp. 2520–2525.

[3] C. Andrew Richter, A. P. Bry, and N. Roy, "Polynomial trajectory planning for aggressive quadrotor flight in dense indoor environments," in. Apr. 2016, pp. 649–666, View online.

[4] M. W. Mueller, M. Hehn, and R. D'Andrea, "A computationally efficient motion primitive for quadrocopter trajectory generation," *IEEE Transactions on Robotics*, vol. 31, no. 6, pp. 1294–1310, Dec. 2015,

[5] J. Chen, T. Liu, and S. Shen, "Online generation of collision-free trajectories for quadrotor flight in unknown cluttered environments," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, View online., May 2016, pp. 1476–1483.

[6] E. Tal and S. Karaman, "Accurate tracking of aggressive quadrotor trajectories using incremental nonlinear dynamic inversion and differential flatness," *CoRR*, vol. abs/1809.04048, 2018, View online.

[7] P. Bouffard, A. Aswani, and C. J. Tomlin, "Learning-based model predictive control on a quadrotor: Onboard implementation and experimental results," *2012 IEEE International Conference on Robotics and Automation*, pp. 279–284, 2012.

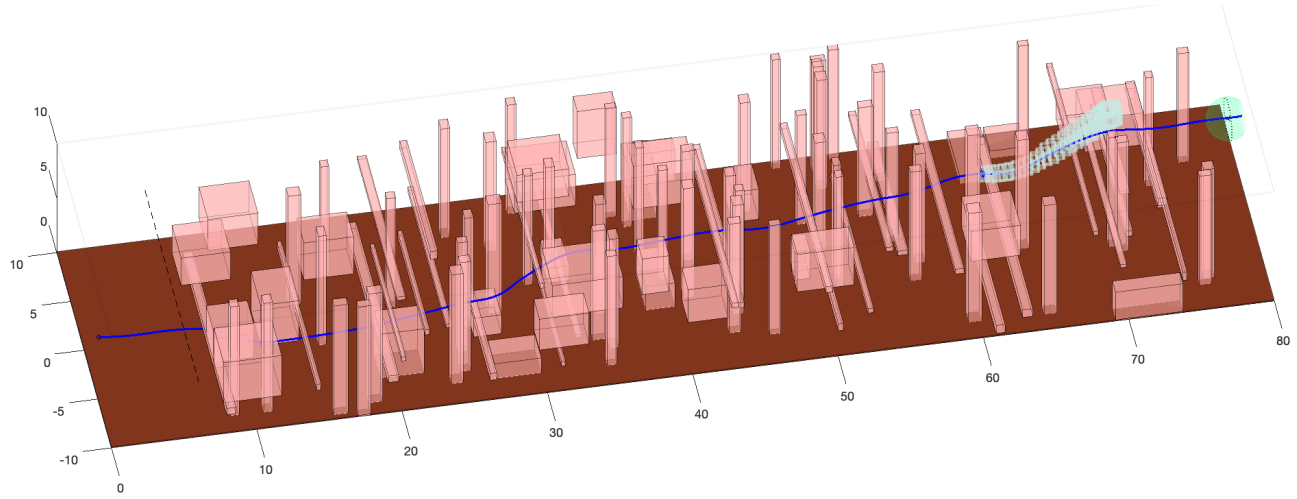

FIGURE 3: EXAMPLE SIMULATED WORLD.

[8] M. Faessler, F. Fontana, C. Forster, and D. Scaramuzza, "Automatic re-initialization and failure recovery for aggressive flight with a monocular vision-based quadrotor," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, View online., May 2015, pp. 1722–1729.

[9] S. L. Herbert, M. Chen, S. Han, S. Bansal, J. F. Fisac, and C. J. Tomlin, "Fastrack: A modular framework for fast and guaranteed safe motion planning," *IEEE Conference on Decision and Control (submitted)*, 2017, View online.

[10] A. Bajcsy, S. L. Herbert, D. Fridovich-Keil, J. F. Fisac, S. Deglurkar, A. D. Dragan, and C. J. Tomlin, "A Scalable Framework For Real-Time Multi-Robot, Multi-Human Collision Avoidance," *arXiv e-prints*, arXiv:1811.05929, arXiv:1811.05929, Nov. 2018, View online.

[11] D. Althoff, M. Althoff, and S. Scherer, "Online safety verification of trajectories for unmanned flight with offline computed robust invariant sets," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Sep. 2015, pp. 3470–3477.

[12] M. Althoff and J. M. Dolan, "Online verification of automated road vehicles using reachability analysis," *IEEE Transactions on Robotics*, vol. 30, no. 4, pp. 903–918, 2014, View online.

[13] S. Kousik, S. Vaskov, M. Johnson-Roberson, and R. Vasudevan, "Safe trajectory synthesis for autonomous driving in unforeseen environments," in *ASME 2017 Dynamic Systems and Control Conference*, View online, American Society of Mechanical Engineers, 2017, V001T44A005–V001T44A005.

[14] S. Kousik, S. Vaskov, F. Bu, M. Johnson-Roberson, and R. Vasudevan, "Bridging the gap between safety and real-time performance in receding-horizon trajectory design for mobile robots," *ArXiv e-prints arXiv:1809.06746*, Sep. 2018, View online.

[15] S. Vaskov, U. Sharma, S. Kousik, M. Johnson-Roberson, and R. Vasudevan, *Guaranteed safe reachability-based trajectory design for a high-fidelity model of an autonomous passenger vehicle*, 2019.

[16] S. Vaskov, S. Kousik, H. Larson, F. Bu, J. Ward, S. Worrall, M. Johnson-Roberson, and R. Vasudevan, *Towards provably not-at-fault control of autonomous robots in arbitrary dynamic environments*, 2019.

[17] S. Kousik, P. Holmes, and R. Vasudevan, *Quadrotor rtd demo*, `https://www.youtube.com/watch?v=toFpIC7Zh18&feature=youtu.be`, Apr. 2019.

[18] ——, "Technical Report: Safe, Aggressive Quadrotor Flight via Reachability-based Trajectory Design," *arXiv e-prints*, arXiv:1904.05728, arXiv:1904.05728, Apr. 2019, View online.

[19] C. Powers, D. Mellinger, A. Kushleyev, B. Kothmann, and V. Kumar, "Influence of aerodynamics and proximity effects in quadrotor flight," in *Experimental Robotics: The 13th International Symposium on Experimental Robotics*, J. P. Desai, G. Dudek, O. Khatib, and V. Kumar, Eds. Heidelberg: Springer International Publishing, 2013, pp. 289–302.

[20] G. M. Hoffmann, H. Huang, S. L. Waslander, and C. J. Tomlin, "Precision flight control for a multi-vehicle quadrotor helicopter testbed," *Control Engineering Practice*, vol. 19, no. 9, pp. 1023 –1036, 2011, Special Section: DCDS '09 The 2nd IFAC Workshop on Dependable Control of Discrete Systems. View online.

[21] A. Technologies, *Asctec hummingbird*, View online., Mar. 2019.

[22] W. Dong, G.-Y. Gu, X. Zhu, and H. Ding, "Development of a quadrotor test bed: Modelling, parameter identification, controller design and trajectory generation," *International Journal of Advanced Robotic Systems*, vol. 12, no. 2, p. 7, 2015, View online.

[23] M. Althoff, "An introduction to CORA 2015," in *Proc. of the Workshop on Applied Verification for Continuous and Hybrid Systems*, 2015, pp. 120–151.

[24] M. Althoff, O. Stursberg, and M. Buss, "Reachability analysis of nonlinear systems with uncertain parameters using conservative linearization," in *2008 47th IEEE Conference on Decision and Control*, Dec. 2008, pp. 4042–4048.