Title

Co-design of Control and Planning for Multi-rotor UAVs with Signal Temporal Logic Specifications

ISBN

9781665441971

Authors

Pant, Yash Vardhan
Yin, He
Arcak, Murat
et al.

Publication Date

2021-05-28

DOI

10.23919/acc50511.2021.9483206

Peer reviewed

Co-design of Control and Planning for Multi-rotor UAVs with Signal Temporal Logic Specifications

Yash Vardhan Pant*, He Yin*, Murat Arcak, Sanjit A. Seshia

***Abstract*— Urban Air Mobility (UAM), or the scenario where multiple manned and Unmanned Aerial Vehicles (UAVs) carry out various tasks over urban airspaces, is a transportation concept of the future that is gaining prominence. UAM missions with complex spatial, temporal and reactive requirements can be succinctly represented using Signal Temporal Logic (STL), a behavioral specification language. However, planning and control of systems with STL specifications is computationally intensive, usually resulting in planning approaches that do not guarantee dynamical feasibility, or control approaches that cannot handle complex STL specifications. Here, we present an approach to co-design the planner and control such that a given STL specification (possibly over multiple UAVs) is satisfied with trajectories that are dynamically feasible and our controller can track them with a bounded tracking-error that the planner accounts for. The tracking controller is formulated for the non-linear dynamics of the individual UAVs, and the tracking error bound is computed for this controller when the trajectories satisfy some kinematic constraints. We also augment an existing multi-UAV STL-based trajectory generator in order to generate trajectories that satisfy such constraints. We show that this co-design allows for trajectories that satisfy a given STL specification, and are also dynamically feasible in the sense that they can be tracked with bounded error. The applicability of this approach is demonstrated through simulations of multi-UAV missions.**

I. Introduction

For Urban Air Mobility to become a reality in the near future, the underlying planning and control approaches for the Unmanned Aerial Vehicles (UAVs) carrying out various tasks should be robust and have strong guarantees on safety. For this, the UAVs must respect *spatial* requirements, e.g. no-fly zones around geo-fenced areas and maintaining safe distances from each other, *temporal* requirements like visiting particular regions only during pre-defined time intervals, and *reactive* requirements, e.g. executing a contingency plan in case of a system failure. It is hard to capture these requirements with the objectives used in traditional control and planning approaches, however recent work on planning and control with Signal Temporal Logic (STL) [1]–[3] has shown promise in representing these missions with STL, and also on developing algorithms for UAVs and fleets of UAVs to satisfy these requirements. However in order to be computationally tractable, these approaches either work with simplifying abstractions of the system dynamics [1] or the workspace [4], or cannot use the full expressiveness of STL [3], e.g. use of the logical *or* operator to represent choices between multiple tasks in a mission.

[2]Department of Electrical Engineering and Computer Sciences, University of California, Berkeley, USA yashpant@berkeley.edu, he_yin@berkeley.edu, arcak@berkeley.edu, sseshia@berkeley.edu.

*The authors contributed equally.

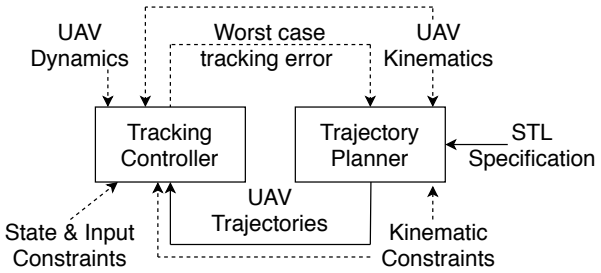

Fig. 1: Overview of the co-design framework. The dashed lines represents information used in the offline design phase, while solid line represent information used during online execution.

In this paper, we present an approach that overcomes these limitations by co-designing the planning and control for multi-rotor UAVs with missions represented as STL specifications (see figure 1). To do so, we augment a state-of-the-art trajectory planning approach [1] to generate trajectories such that a controller that considers the non-linear UAV dynamics can track them. We also use sum-of-squares programming [5] to synthesize a controller that can track these trajectories with bounded error. We explicitly compute this bound and use it in the trajectory planner to ensure that the planned trajectories *robustly* satisfy the STL specification, and that the system, when tracking with our controller, does not violate the specification.

A. Contributions of this work

The main contributions of our co-design approach are that it ensures:

1) The trajectories (generated via a centralized optimization) satisfy the STL specification in continuous time.
2) They also respect pre-defined kinematic constraints in the form of bounds on velocity, acceleration, and jerk.
3) The non-linear dynamics and state/input constraints of the UAVs are taken into account by the tracking controller, and worst-case tracking error bounds are computed.
4) The trajectory planner satisfies the STL specification with a robustness margin, large enough to accommodate the tracking error bound.

Additionally, our approach can harness the full grammar of Signal Temporal Logic (see section II-B). Through two simulation case studies, we show how this framework can be applied to realistic scenarios and how the guarantees above translate to actual control performance and robust satisfaction of the STL specification.

B. Related work

Planning and Control with STL specifications. The problem of controlling systems with STL specifications has been studied extensively. [6], [7] build upon [8] to develop a Mixed Integer Programming-based Model Predictive Control approach (MPC). These methods are developed with Linear Time Invariant (LTI) systems in mind, and are computationally complex to be applied for the control of multiple UAVs with complex STL specifications. [2] presents a non-convex optimization-based MPC approach that is applicable to some non-linear systems, but is also computationally heavy to be used for online control. A barrier function-based approach is presented in [3], but is applicable only to a fragment of STL. Given the computational limitations in developing controllers for STL specifications, the work in [1] instead aims to generate trajectories for multi-rotor UAVs such that they satisfy a given STL specification. It however offers only kinematic feasibility of the planned trajectories, and not dynamic feasibility. Our co-design framework builds on this approach, and we show we can also ensure dynamic feasibility through additional constraints, and through our design of a tracking controller. There are also planning approaches that, unlike our approach, discretize the workspace and abstract away the system dynamics (e.g. [4]).

Co-designing the planner and controller. A hierarchical control framework has been explored by the path planning community [9]–[11]. In this framework, a low-fidelity model is used for planning online, and a high-fidelity model is used for designing a tracking controller offline, and the controller is then implemented online. A robust forward reachable set used for planning is computed in [9] by considering the difference between the planning and tracking models. In [10], a Hamilton-Jacobi equation based method is presented to synthesize a tracking error bound and an optimal tracking controller. An extension in [11] shows that for a class of robotics systems, the problem can also be solved using sum-of-squares (SOS) programming. Related works other than those from the path planing literature include the continuous-state abstraction [12], [13], which provides a tracking controller and ensure the boundedness of the error between the original high dimensional linear system and its low dimensional abstraction. This idea is extended to a general class of nonlinear systems in [14]–[16], and is applied to UAVs in this paper.

C. Outline of the paper

In Section II we formalize the problem that we aim to solve and provide a brief introduction to Signal Temporal Logic and its associated robustness metric. Section III gives an overview of the co-design approach, introduces the system dynamics used for planning and control, and the associated hierarchical control structure. Section IV presents the trajectory planner, which builds upon the work in [1], and discusses how the controller performance and kinematic constraints are taken into account. Section V presents the synthesis for the tracking controller that uses the full nonlinear UAV dynamics, and its associated tracking error. Two case studies for multi-UAV settings with STL specifications are presented in Section VI. Finally, we conclude with a discussion of the limitations of this approach and future work in Section VII.

D. Notations

For $\xi \in \mathbb{R}^n$, $\mathbb{R}[\xi]$ represents the set of polynomials in ξ with real coefficients, and $\mathbb{R}^m[\xi]$ and $\mathbb{R}^{m\times p}[\xi]$ to denote all vector and matrix valued polynomial functions. The subset $\Sigma[\xi] := \left\{\pi = \sum_{i=1}^{M} \pi_i^2 : M \geq 1, \pi_i \in \mathbb{R}[\xi]\right\}$ of $\mathbb{R}[\xi]$ is the set of sum-of-squares (SOS) polynomials. The state of UAV i is denoted by $x_i \in \mathbb{R}^n$. A trajectory of states over a time interval $[0, T]$ is denoted by $\mathbf{x}_i : [0, T] \to \mathbb{R}^n$. The value of the state at time t is denoted by $\mathbf{x}_i(t)$.

II. Problem Statement and preliminaries

A. Problem Statement

Formally, in order to achieve the objectives laid out earlier in the paper, we aim to solve the following problem:

Problem 1: Given UAV dynamics $\dot{\mathbf{x}}_i(t) = f(\mathbf{x}_i(t)) + g(\mathbf{x}_i(t))\mathbf{u}_i(t)$ and output $\mathbf{y}_i(t) = C\mathbf{x}_i(t)$ for UAV $i \in \{1, \ldots, D\}$, control the UAVs such that resulting output trajectories ($\mathbf{y}_i$) are such that:

1) $(\mathbf{y}_1, \ldots, \mathbf{y}_D) \models \varphi(\mathbf{y}_1, \ldots, \mathbf{y}_D)$, i.e. satisfy the STL specification φ and,
2) respect constraints $\mathbf{x}_i(t) \in X \subset \mathbb{R}^n$, $\mathbf{u}_i(t) \in U \subset \mathbb{R}^m$, $\forall i = 1, \ldots, D$, $\forall t \in [0, \text{hrz}(\varphi)]$.

Here, $\mathbf{x}_i(t)$ and $\mathbf{u}_i(t)$ are the state and input for UAV i at time t, X and U are (hyper-rectangular) state and input constraints, and $\text{hrz}(\varphi) < \infty$ is the time horizon of the specification φ, i.e. the minimum time duration needed to evaluate φ. For simplicity we assume that all UAVs have identical dynamics, however this not necessary for our approach to work.

In order to avoid the pitfalls of existing approaches, we solve this problem by solving the two following subproblems instead, which constitute a hierarchical planner-controller framework:

Problem 2 (Trajectory planning): Develop a planner that uses the UAV kinematics of the form $\dot{\hat{\mathbf{x}}}_i(t) = A\hat{\mathbf{x}}_i(t) + B\hat{\mathbf{u}}_i(t)$ (details in Section III-A) to generate trajectories $\hat{\mathbf{x}}_\mathbf{i}$ such that:

1) $(\hat{\mathbf{x}}_1, \ldots, \hat{\mathbf{x}}_D) \models \varphi(\hat{\mathbf{x}}_1, \ldots, \hat{\mathbf{x}}_D)$ and,
2) $\hat{\mathbf{x}}_i(t) \in \hat{X}$, $\hat{\mathbf{u}}_i(t) \in \hat{U}$ $\forall i = 1, \ldots, D$, $\forall t \in [0, \text{hrz}(\varphi)]$,
3) any trajectories $\mathbf{x}_i$ where $||C(\mathbf{x}_i(t) - G\hat{\mathbf{x}}_i(t))||_\infty \leq \delta$, $\forall t \in [0, \text{hrz}(\varphi)]$ also satisfy φ.

Here, G is a matrix of appropriate size that relates the states in the kinematic model to those in the full dynamics, and $\delta \in \mathbb{R}_{>0}$ is the worst-case tracking error bound. See Section V for details. The controller design is done individually for each UAV, so we drop the subscript i for ease of notation:

Problem 3 (Tracking Controller): For the full UAV dynamics (problem 1), develop a controller $\mathbf{u}(t) = k(\mathbf{e}(t))$ s.t. the error state $\mathbf{e}(t) = \mathbf{x}(t) - G\hat{\mathbf{x}}(t)$ for tracking a trajectory $\hat{\mathbf{x}}$ is s.t. $||\mathbf{e}(t)||_\infty \leq \delta$, and $\mathbf{x}(t) \in X$, $\mathbf{u}(t) \in U$ $\forall t \in [0, \text{hrz}(\varphi)]$.

B. Review of Signal Temporal Logic (STL)

Signal Temporal Logic (STL) [17] is a behavioral specification language that can be used to encode requirements on signals. The grammar of STL [6] allows for capturing a rich set of behavioral requirements using temporal operators, such as the *Until* ($\mathcal{U}$) operator and derived *Always* ($\Box$) and *Eventually* ($\Diamond$), as well as logical operators *And* ($\wedge$), *Or* ($\vee$), and *negation* ($\neg$). With these operators, an STL specification φ is defined over a signal, e.g. over the trajectories of the UAVs, and evaluates to either *True* or *False*.

Example 1: (A two-UAV timed reach-avoid problem) Two quad-rotor UAVs are tasked with a mission with spatial and temporal requirements in the workspace shown in Fig. 2:

1) The two UAVs have to reach a Goal set (shown in green), or a region of interest, within a time of 8 seconds after starting. UAV j (where $j \in \{1, 2\}$), with position (in 3D) denoted by p_j, has to satisfy: $\varphi_{\text{reach},j} = \Diamond_{[0,8]}(p_j \in \text{Goal})$. The *Eventually* operator over the time interval $[0, 8]$ requires UAV j to be inside the set Goal at some point within 8 seconds.
2) The two UAVs also have to avoid an Unsafe set (in red), or no-fly zone. For each UAV j, this is encoded with *Always* and *Negation* operators: $\varphi_{\text{avoid},j} = \Box_{[0,8]}\neg(p_j \in \text{Unsafe})$
3) Finally, the two UAVs should also be separated by at least 0.2 meters along every axis of motion: $\varphi_{\text{separation}} = \Box_{[0,8]}||p_1 - p_2||_\infty \geq 0.2$

These requirements are combined into a two UAV timed reach-avoid specification:

$$\varphi_{\text{reach-avoid}} = \wedge_{j=1}^{2}(\varphi_{\text{reach},j} \wedge \varphi_{\text{avoid},j}) \wedge \varphi_{\text{separation}} \quad (1)$$

In order to satisfy φ, a planning method generates trajectories $\hat{\mathbf{x}}_1, \hat{\mathbf{x}}_2$ (where position p is obtained from the full kinematic state $\hat{x}$ as $p = C\hat{x}$) of a duration that is at least $\text{hrz}(\varphi) = 8\text{s}$, where $\text{hrz}(\varphi)$ is the time *horizon* of φ. If the trajectories satisfy the specification, i.e. $(\mathbf{p}_1, \mathbf{p}_2) \models \varphi$, then the specification φ evaluates to *True*, otherwise it is *False*. In general, an upper bound for the time horizon can be computed as shown in [6]. Here, we only consider specifications for which the horizon is bounded. More details on STL are in [17] or [6].

C. Robustness of STL specifications

For every STL specification φ, we can construct a *robustness function* [18] by following the grammar of STL. It outputs a *robustness* value $\rho_\varphi(\mathbf{x})$ for this formula, with respect to the signal $\mathbf{x}$ that it is defined over, and has the important following property:

Theorem 2.1: [18] For any $\mathbf{x}$ and STL formula φ, if $\rho_\varphi(\mathbf{x}) < 0$ then $\mathbf{x}$ violates φ, and if $\rho_\varphi(\mathbf{x}) > 0$ then $\mathbf{x}$ satisfies φ. The case $\rho_\varphi(\mathbf{x}) = 0$ is inconclusive.

Thus, the degree of satisfaction or violation of a specification is indicated by the robustness value. For simplicity, the distances are defined in the inf-norm sense. This, combined with Theorem 2.1 gives us the following result:

Corollary 1.1: Given a trajectory $\hat{\mathbf{x}}$ such that $\hat{\mathbf{x}} \models \varphi$ with robustness value $\rho_\varphi(\hat{\mathbf{x}}) > 0$, then any trajectory $\mathbf{x}$ that is within $\rho_\varphi(\hat{\mathbf{x}})$ of $\hat{\mathbf{x}}$ at each time, i.e. $||\hat{\mathbf{x}}(t) - \mathbf{x}(t)||_\infty < \rho_\varphi(\hat{\mathbf{x}})\, \forall t \in [0, \text{hrz}(\varphi)]$, is such that $\mathbf{x} \models \varphi$ (satisfies φ).

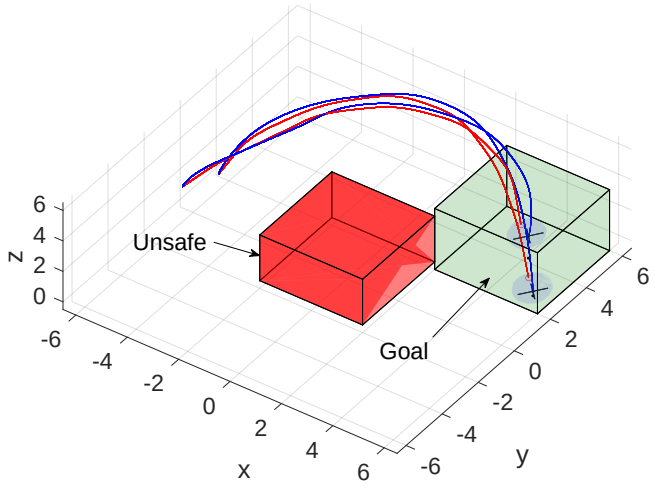

Fig. 2: The workspace, planned trajectories (red) and the tracked trajectories (blue) for the two UAVs carrying out the timed reach-avoid mission of example 1. Simulation videos are at `http://bit.ly/TimedRA`. Color in digital version.

Significance: We use this property in the co-design, where the planner generates a trajectory $\hat{\mathbf{x}}$ and the tracking controller tracks it with a trajectory $\mathbf{x}$ such that $||C(\mathbf{x}(t) - G\hat{\mathbf{x}}(t))||_\infty \leq \rho_\varphi(\hat{\mathbf{x}})\, \forall t \in [0, \text{hrz}(\varphi)]$. This requires the the inf-norm of the worst-case tracking error is always less than $\rho_\varphi(\hat{\mathbf{x}})$. Note, φ is usually over a subset of states given by Cx, e.g. the specification 1 is over the position components of the UAV state. In general, our approach allows for specifications over the translational positions, velocities and accelerations of the UAV (Section IV).

III. Overview of the Co-design Approach

In order to solve problems 2 and 3, we adopt a hierarchical planning and control stack for the UAVs of the kind that is commonly used, see figure 3. Our framework co-designs (see figure 1) the two main components here, namely:

- **Trajectory planner** that takes as input the STL specification defining the (possibly multi-UAV) mission. Using the UAV kinematics (Section III-A, the trajectory generator solves an optimization (in a centralized manner) that generates trajectories for all UAVs involved. The trajectories satisfy the STL specification in continuous time, ensure that kinematic constraints are respected, and take into account the worst-case tracking error. As long as the tracking controller does not deviate from a given bound, obtained from this optimization's objective, the tracking of these trajectories will be such that the STL specification is satisfied. Section IV presents the details.
- **Tracking controller** that follows the trajectories generated by the planner by taking into account both the non-linear dynamics of multi-rotor UAVs (Section III-B) and the kinematics that the planner uses to generate

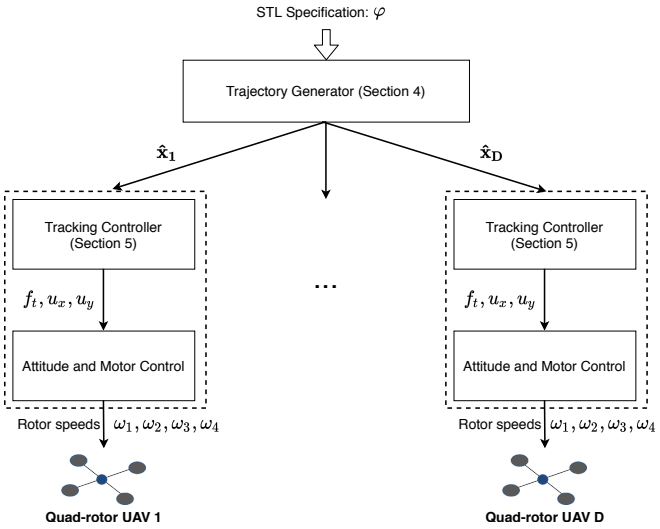

Fig. 3: The planning and control hierarchy for multi UAV missions.

trajectories. This controller is synthesized via a sum-of-squares optimization through which we obtain (state-dependent) tracking error bounds and worst-case tracking error bounds, which are then used in the trajectory planning optimization. Details on the controller design are in Section V.

Commands from the tracking controller are realized by lower level attitude controllers (Section III-D). The details of attitude controllers are beyond the scope of this paper, but standard approaches can be found in [19]. The resulting planning and control hierarchy is shown in figure 3. Note that the trajectory planning is done in a centralized manner, but the synthesized tracking controllers run independently on each UAV to track these generated trajectories. The rest of this section introduces the UAV kinematics and dynamics used by the trajectory planner and tracking controller, respectively.

A. Multi-rotor UAV kinematics for trajectory planning

To generate UAV trajectories that satisfy a STL specification φ, we use the method in [1] that generates trajectories by connecting segments of jerk-minimizing splines [20]. These are based on the following kinematic model.

$$\dot{\hat{\mathbf{x}}}(t) = \begin{bmatrix} \dot{\hat{\mathbf{p}}}_x(t) \\ \dot{\hat{\mathbf{p}}}_y(t) \\ \dot{\hat{\mathbf{p}}}_z(t) \\ \dot{\hat{\mathbf{v}}}_x(t) \\ \dot{\hat{\mathbf{v}}}_y(t) \\ \dot{\hat{\mathbf{v}}}_z(t) \\ \dot{\hat{\mathbf{a}}}_x(t) \\ \dot{\hat{\mathbf{a}}}_y(t) \\ \dot{\hat{\mathbf{a}}}_z(t) \end{bmatrix} = \begin{bmatrix} \hat{\mathbf{v}}_x(t) \\ \hat{\mathbf{v}}_y(t) \\ \hat{\mathbf{v}}_z(t) \\ \hat{\mathbf{a}}_x(t) \\ \hat{\mathbf{a}}_y(t) \\ \hat{\mathbf{a}}_z(t) \\ 0 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ \hat{\mathbf{j}}_x(t) \\ \hat{\mathbf{j}}_y(t) \\ \hat{\mathbf{j}}_z(t) \end{bmatrix} \tag{2}$$

Here $\hat{p}_x$, $\hat{v}_x$, $\hat{a}_x$ represents the position, velocity, and acceleration of the UAV in the global x coordinate (and similarly for y and z coordinates), and jerks $\hat{j}_x$, $\hat{j}_y$, and $\hat{j}_z$ are the inputs $\hat{u}$ to this system.

B. Quad-rotor UAV dynamics

For the UAVs, we consider the following non-linear dynamics, adapted from [19, Chapter 2],

$$\dot{\mathbf{x}}(t) = \begin{bmatrix} \dot{\mathbf{p}}_x(t) \\ \dot{\mathbf{p}}_y(t) \\ \dot{\mathbf{p}}_z(t) \\ \dot{\mathbf{v}}_x(t) \\ \dot{\mathbf{v}}_y(t) \\ \dot{\mathbf{v}}_z(t) \\ \dot{\boldsymbol{\phi}}(t) \\ \dot{\boldsymbol{\theta}}(t) \end{bmatrix} = \begin{bmatrix} \mathbf{v}_x(t) \\ \mathbf{v}_y(t) \\ \mathbf{v}_z(t) \\ -\frac{\mathbf{f_t}(t)}{m} c_{\boldsymbol{\phi}(t)} s_{\boldsymbol{\theta}(t)} \\ \frac{\mathbf{f_t}(t)}{m} s_{\boldsymbol{\phi}(t)} \\ g - \frac{\mathbf{f_t}(t)}{m} c_{\boldsymbol{\phi}(t)} c_{\boldsymbol{\theta}(t)} \\ \frac{\mathbf{u}_x(t)}{I_x} \\ \frac{\mathbf{u}_y(t)}{I_y}, \end{bmatrix} \tag{3}$$

where $c_\phi := \cos(\phi)$, $s_\phi := \sin(\phi)$ (and similarly for θ).

The 8-state system has 6 translational states p_x, p_y and p_z (the positions of the quad-rotor in a global co-ordinate frame) and v_x, v_y and v_z (velocities), as well as two states for orientation ϕ and θ which are the roll and pitch angles respectively. To obtain this model, we assume that yaw ψ and its derivatives are zero throughout. The control inputs to this model are f_t (thrust), and u_x and u_y which are related to the torques generated by the rotors. We define a variable u to gather all the control inputs: $u = [f_t, u_x, u_y]^\top$. Without loss of generality, we assume the input constraint set is of the form: $U = \{u \in \mathbb{R}^3 : \underline{u} \leq u \leq \overline{u}\}$, where $\underline{u}, \overline{u} \in \mathbb{R}^3$, and $\leq$ is applied elementwise. Here, m is the UAV's mass, and I_x and I_y are the moments of inertia with respect to x and y axes in the body-fixed frame. Values for them are: $m = 0.5$, and $I_x = I_y = 0.2$. [1]

C. Error dynamics for the tracking controller

To help keep track of the "distance" between the planning and tracking trajectories, we define the error state as:

$$e = x - G\hat{x} = \begin{bmatrix} p_x - \hat{p}_x \\ p_y - \hat{p}_y \\ p_z - \hat{p}_z \\ v_x - \hat{v}_x \\ v_y - \hat{v}_y \\ v_z - \hat{v}_z \\ \phi \\ \theta \end{bmatrix}, \tag{4}$$

which include the positional and velocity differences, and the roll and pitch angles of the UAV. Here $G = \text{diag}(I_6, 0_{2\times 3})$. Later in Section V, we will design a tracking controller to penalize the error state e, and ensure its boundedness.

The corresponding error dynamics are:

$$\dot{\mathbf{e}}(t) = f_e(\mathbf{e}(t), \hat{\mathbf{x}}(t)) + g_e(\mathbf{e}(t))\mathbf{u}(t), \tag{5}$$

where

$$f_e(e, \hat{x}) = \begin{bmatrix} e_4 \\ e_5 \\ e_6 \\ -\hat{a}_x \\ -\hat{a}_y \\ g - \hat{a}_z \\ 0 \\ 0 \end{bmatrix}, \; g_e(e) = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ \frac{-c_{e_7} s_{e_8}}{m} & 0 & 0 \\ \frac{s_{e_7}}{m} & 0 & 0 \\ \frac{-c_{e_7} c_{e_8}}{m} & 0 & 0 \\ 0 & \frac{1}{I_x} & 0 \\ 0 & 0 & \frac{1}{I_y} \end{bmatrix}.$$

[1] For simplicity, we assume all UAVs have identical dynamics, however our approach would apply otherwise as well.

In the error dynamics, $\hat{a}_x, \hat{a}_y$ and $\hat{a}_z$ are treated as bounded disturbances, whose bounds are enforced by the STL based planning algorithm described in Section IV.

D. Lower level attitude and motor controllers

The attitude controller takes as measurements the roll and pitch rates $\dot{\phi}$, $\dot{\theta}$, and u_x, u_y as references. It is tasked with generating torques T_x, T_y and T_z to realize u_x and u_y.

$$\begin{aligned} \dot{\mathbf{u}}_x(t) &= \mathbf{T}_x(t) \\ \dot{\mathbf{u}}_y(t) &= \mathbf{T}_y(t) \\ \mathbf{T}_z(t) &= (I_y - I_x)\dot{\boldsymbol{\phi}}(t)\dot{\boldsymbol{\theta}}(t) \end{aligned} \tag{6}$$

Finally, from the computed values of f_t, T_x, T_x and T_z, the desired rotor (angular) speeds ω_i, $i \in \{1, \ldots, 4\}$ can be computed by solving the equality constrained system [19]:

$$\begin{aligned} f_t &= b\sum_{i=1}^{4} \omega_i^2 \\ T_x &= bl(\omega_3^2 - \omega_1^2) \\ T_y &= bl(\omega_4^2 - \omega_2^2) \\ T_z &= d(\omega_4^2 + \omega_2^2 - \omega_3^2 - \omega_1^2) \end{aligned} \tag{7}$$

Here, b, l, d are UAV specific constants [19]. The solution to this system acts as the desired angular speeds for the lower-level motor controllers to track. The formulation of these lower-level controllers is beyond the scope of this work, and the interested reader can refer to [19] for details.

IV. Planning for UAVs with STL specifications

For trajectory planning with a given STL specification φ, defined possibly over trajectories of multiple UAVs, we use the method of [1] with some modifications. This approach maximizes the robustness ρ_φ (see Theorem 2.1) of the given STL specification by selecting position waypoints connected by jerk minimizing splines [20]. The segment between one waypoint, given by position $\hat{p}^0 = [\hat{p}_x^0, \hat{p}_y^0, \hat{p}_z^0]^\top$ and velocity $\hat{v}^0 = [\hat{v}_x^0, \hat{v}_y^0, \hat{v}_z^0]^\top$, and another with desired position $\hat{p}^1 = [\hat{p}_x^1, \hat{p}_y^1, \hat{p}_z^1]^\top$, is a trajectory (see figure 4) of fixed time duration T with the states of the kinematic model (2) ($\forall l \in \{x, y, z\}$, $\forall t \in [0, T]$) given by:

$$\begin{bmatrix} \hat{\mathbf{p}}_l(t) \\ \hat{\mathbf{v}}_l(t) \\ \hat{\mathbf{a}}_l(t) \end{bmatrix} = \begin{bmatrix} \frac{\alpha}{120}t^5 + \frac{\beta}{24}t^4 + \frac{\gamma}{6}t^3 + \hat{v}_l^0 t + \hat{p}_l^0 \\ \frac{\alpha}{24}t^4 + \frac{\beta}{6}t^3 + \frac{\gamma}{2}t^2 + \hat{v}_l^0 \\ \frac{\alpha}{6}t^3 + \frac{\beta}{2}t^2 + \gamma t \end{bmatrix} \tag{8}$$

Here, α, β and γ are linear functions of $\hat{p}^0$, $\hat{v}^0$ and $\hat{p}^1$ (and parameter T) [20]. We assume that the start and end accelerations are zero, i.e. $\hat{a}^0 = \hat{a}^1 = 0_{3\times 1}$ and end velocity $\hat{v}^1$ is not fixed. For brevity, we omit further details here. The interested reader can refer to [1], [20].

Trajectory planning optimization: We use the Fly-by-Logic method [1] to generate a sequence of $N+1$ position waypoints in 3-D space for each UAV (i) $\hat{p}_i^{0:N} = [\hat{p}_i^0, \ldots, \hat{p}_i^N]$, where total flight time $NT \geq \text{hrz}(\varphi)$, such that the resulting trajectories (see figure 4) of (8) $\hat{\mathbf{p}}_i : [0, NT] \to \mathbb{R}^9$ maximize a continuously differentiable approximation ($\tilde{\rho}_\varphi$) [2] of the robustness ρ_φ.

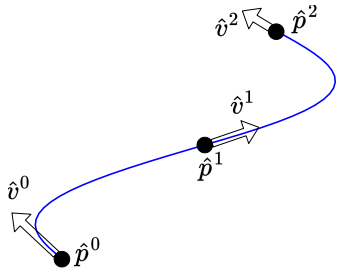

Fig. 4: Jerk minimizing splines connecting position waypoints $\hat{p}^1$, $\hat{p}^2$ and $\hat{p}^3$. The arrows show the velocities at these waypoints $\hat{v}^i$, $\forall i \in \{0, 1, 2\}$.

$$\max_{\hat{p}_1^{0:N}, \ldots, \hat{p}_D^{0:N}} \quad \tilde{\rho}_\varphi([\hat{\mathbf{p}}_1, \ldots, \hat{\mathbf{p}}_D]) \tag{9a}$$

$$\text{s.t. } \forall i = 1, \ldots, D,\ \forall j = 0, \ldots, N-1 \tag{9b}$$

$$\text{LB}_{\text{vel}}(\hat{v}_i^j) \leq \hat{p}_i^{j+1} - \hat{p}_i^j \leq \text{UB}_{\text{vel}}(\hat{v}_i^j) \tag{9c}$$

$$\text{LB}_{\text{acc}}(\hat{v}_i^j) \leq \hat{p}_i^{j+1} - \hat{p}_i^j \leq \text{UB}_{\text{acc}}(\hat{v}_i^j) \tag{9d}$$

$$\text{LB}_{\text{jerk}}(\hat{v}_i^j) \leq \hat{p}_i^{j+1} - \hat{p}_i^j \leq \text{UB}_{\text{jerk}}(\hat{v}_i^j) \tag{9e}$$

$$\tilde{\rho}_\varphi([\hat{\mathbf{p}}_\mathbf{1}, \ldots, \hat{\mathbf{p}}_\mathbf{D}]) \geq \tilde{\epsilon} + \delta \tag{9f}$$

This is a non-convex optimization, with linear constraints [1] and can be solved using off-the-shelf solvers like IPOPT. The UAV trajectories $\hat{\mathbf{p}}_\mathbf{i}$, when discretized in time, are linear functions of the position waypoints $\hat{p}_i^{0:N}$ (details in [1]). Constraints (9c), (9d) are linear functions of the waypoints (and associated waypoint velocities) to ensure that the velocity and acceleration for each axis of motion are within predefined intervals $[v_{\min}, v_{\max}]$ and $[a_{\min}, a_{\max}]$ respectively. These constraints are from [1]. Additionally we develop constraints (9e), also linear in the optimization variables, for the jerk to be within pre-defined intervals $[j_{\min}, j_{\max}]$ as well. The expressions for these constraints are in the Appendix. Finally, the constraint (9f), where $\tilde{\epsilon}$ is the worst case approximation error [2] of $\tilde{\rho}_\varphi$ with respect to ρ_φ, is to ensure that the STL robustness of the trajectories with respect to the specification φ is above a threshold $\delta > 0$. As we will see in the next section, δ is the worst case tracking error achieved by the tracking controller.

Theorem 4.1 (STL satisfaction and kinematic feasibility): A feasible solution to the optimization (9) generates trajectories $\hat{\mathbf{p}}_\mathbf{1}, \ldots, \hat{\mathbf{p}}_\mathbf{D}$ such that they:

1) Satisfy the STL specification φ in continuous time
2) Have bounded velocity, acceleration and jerk (along every axis of motion $l \in \{x, y, z\}$) such that $\forall t \in [0, \text{hrz}(\varphi)]$: $\hat{\mathbf{v}}_l(t) \in [v_{\min}, v_{\max}]$, $\hat{\mathbf{a}}_l(t) \in [a_{\min}, a_{\max}]$ and $\hat{\mathbf{j}}_l(t) \in [j_{\min}, j_{\max}]$ for each UAV.
3) As long as each UAV tracks these trajectories with tracking error $\mathbf{e}(t)$ such that $||\mathbf{e}(t)||_\infty \leq \delta$ $\forall t \in [0, \text{hrz}(\varphi)]$, the specification φ is satisfied.

This trajectory planning approach hence serves as a solution to problem 2. The proofs for the first two points (except the bounded jerk constraints) can be found in [1]. The third point is a direct consequence of Corollary 1.1.

V. Controller and Error Bound Synthesis

In (5), we have seen that the planner states enter the error dynamics. When designing a tracking controller, and computing an error bound (to solve problem 3), the planner states

are treated as uncertain parameters. Additionally, from the second point of Theorem 4.1, we know that these uncertain parameters are bounded: $\hat{\mathbf{x}}(t) \in \hat{X}\ \forall t \in [0, \text{hrz}(\varphi)]$, where $\hat{X} = \mathbb{R}^3 \times [v_{\min}, v_{\max}]^3 \times [a_{\min}, a_{\max}]^3$. With this information in mind, the following proposition proposes a way of computing a tracking controller that ensures boundedness of the tracking error.

Proposition 1: Given error dynamics (5) with mappings $f_e : \mathbb{R}^n \times \mathbb{R}^{\hat{n}} \to \mathbb{R}^n$, $g_e : \mathbb{R}^n \to \mathbb{R}^{n\times m}$, and $\eta \in \mathbb{R}$, $\hat{X} \subseteq \mathbb{R}^{\hat{n}}$, $\underline{u}, \overline{u} \in \mathbb{R}^m$, if there exist a $\mathcal{C}^1$ function $V : \mathbb{R}^n \to \mathbb{R}$, and $k : \mathbb{R}^n \to \mathbb{R}^m$, such that for all $\hat{x} \in \hat{X}$, the following constraints hold,

$$V(e) = \eta \ \Rightarrow\ \frac{\partial V}{\partial e} \cdot (f_e(e, \hat{x}) + g_e(e)k(e)) \leq 0, \tag{10a}$$

$$V(e) \leq \eta \ \Rightarrow\ \underline{u} \leq k(e) \leq \overline{u}, \tag{10b}$$

then the sublevel set $\Omega_\eta^V := \{e \in \mathbb{R}^n : V(e) \leq \eta\}$ is forward invariant.

The sublevel set Ω_η^V is the error bound achieved by the tracking controller k. In practice, it can be difficult to find generic functions V and k satisfying constraints (10). Therefore, we use sum-of-squares (SOS) programming to search for them by restricting them to polynomials: $V \in \mathbb{R}[e]$, and $k \in \mathbb{R}^m[e]$. This in turn requires the error dynamics to be polynomials: $f_e \in \mathbb{R}^n[(e, \hat{a})]$, $g_e \in \mathbb{R}^{n\times m}[e]$, where $\hat{a} := [\hat{a}_x, \hat{a}_y, \hat{a}_z]^\top$. Additionally, define polynomials $d_l = (a_{\max} - \hat{a}_l)(\hat{a}_l + a_{\min})$, $l \in \{x, y, z\}$.

By applying the generalized S-procedure [5] to (10) to get its corresponding SOS constraints, and using the volume of the error bound as the cost function, we obtain the following SOS optimization problem.

$$\begin{aligned}
&\min_{V,k,s_i} \text{volume}(\Omega_\eta^V) \\
&\text{s.t. } s_1^l \in \Sigma[(e, \hat{a})],\ \forall l \in \{x, y, z\},\ s_2 \in \mathbb{R}[(e, \hat{a})], \\
&\quad s_3^j, s_4^j \in \Sigma[e],\ \forall j \in \{1, ..., m\}, \\
&\quad k \in \mathbb{R}^m[e],\ V \in \mathbb{R}[e], && \text{(11a)} \\
&\quad -\frac{\partial V}{\partial e} \cdot (f_e + g_e k) - \sum_{l\in\{x,y,z\}} s_1^l d_l \\
&\quad\quad - (V - \eta)s_2 \in \Sigma[(e, \hat{a})], && \text{(11b)} \\
&\quad \overline{u}_i - k_i + (V - \eta)s_3^j \in \Sigma[e],\ \forall j \in \{1, ..., m\}, && \text{(11c)} \\
&\quad k_i - \underline{u}_i + (V - \eta)s_4^j \in \Sigma[e],\ \forall j \in \{1, ..., m\}, && \text{(11d)}
\end{aligned}$$

where polynomials decision variables s_i are called S-procedure certificates. The optimization (11) is non-convex as there are two groups of decision variables V and (k, s_2, s_3^j, s_4^j) bilinear in each other. We tackle it by using the so called alternating direction algorithm summarized in [14, Algorithm 1]. As a concrete example of the outputs of optimization (11), the tracking controller for u_x (in (3)) is:

$$\begin{aligned}
u_x(e) =& 0.001e_1e_6 + 0.083e_2e_3 + 0.139e_2e_6 + 0.070e_3e_5 \\
&- 0.186e_3e_7 + 0.003e_3e_8 + 0.001e_4e_6 + 0.120e_5e_6 \\
&- 0.062e_6e_7 + 0.001e_6e_8 - 0.001e_1 - 0.289e_2 \\
&- 0.002e_4 - 0.320e_5 - 1.142e_7 + 0.001e_8.
\end{aligned}$$

Once the error bound Ω_η^V is obtained, we can compute the worst-case tracking error δ by solving a convex optimization

$$\min_{\delta>0} \text{ s.t. } \Omega_\eta^V \Rightarrow -\delta \leq e_i \leq \delta,\ i = 1, ..., n. \tag{12}$$

Key result: The worst-case tracking error δ acts as the interface between the control and planning algorithms. Its use is formally stated below:

Theorem 5.1: Given kinematic constraint set $\hat{X}$, resulting error bound Ω_η^V, tacking controller k, and worst-case tracking error δ, if there exists a feasible planned trajectory satisfying the STL specification φ when solving the optimization (9) with $\hat{X}$ and δ, then the trajectories of the nonlinear UAV dynamics (3) under the control of k satisfy φ.

This follows from the controller synthesis procedure, and the trajectory planning guarantees of Theorem 4.1. The co-design framework ensures that planned trajectories satisfy the STL specification φ, and are dynamically feasible for the controller k to track without violating φ. Hence, together the planner and controller serve as a solution to problem 1[2].

Co-design procedure: We now summarize the co-design process, which is outlined in figure 1.

1) Given the kinematic constraint set $\hat{X}$ and the UAV input constraint set U, solve optimizations (11) and (12) to get k and δ (offline).
2) Solve the optimization (9) to obtain planned trajectories $\hat{\mathbf{x}}$ satisfying φ with robustness above δ (offline).
3) Track $\hat{\mathbf{x}}$ for each UAV using k (online).

VI. Simulations

We demonstrate our framework through two simulation examples, adapted from [1]. The only other approach that can handle the full STL semantics, as well as non-linear dynamics is [2]. However these examples, especially the second example with 4 UAVs, are well beyond its computational abilities. We do not perform comparisons to other approaches as they either are for planning only [1], [4], or cannot handle non-linear dynamics [6], [7].

Simulation setup: The trajectory planning algorithm and the controller synthesis was implemented in Matlab, and non-linear UAV dynamics (3) were used for the trajectory tracking. The trajectory planning optimization is implemented using Casadi, and solved using IPOPT as the solver. The trajectory computation times are of the order of a few seconds for both examples. A thorough evaluation of computation times can be found in [1][3]. The SOS optimization (11) is formulated and translated into semi-definite programming (SDP) using SOSOPT, and the SDP is solved using MOSEK. Variables V and k are parametrized using degree-2 polynomials in both examples, and the corresponding SOS optimization takes 1.75×10^3 seconds to solve (offline).

A. Two-UAV Timed Reach Avoid

We start with the mission in Example 1, where two UAVs are tasked with a timed reach-avoid mission (see figure 2).

Results: In the instance shown in figure 2, the UAV initial states correspond to UAV 1 and 2 with positions $[-5, 0, 2.75]^\top$ and $[-5, 2, 4]^\top$ respectively, with zero initial

[2]The control synthesis does not take state constraint X into account directly, however it guarantees $\mathbf{x}(t) \in G\hat{X} \oplus \mathbb{B}_\delta$ where $\mathbb{B}_\delta$ is a box with sides δ, G projects $\hat{X}$ on the state space x and $\oplus$ is the Minkowski sum.

[3]The additional constraints for bounded jerk add little to no overhead

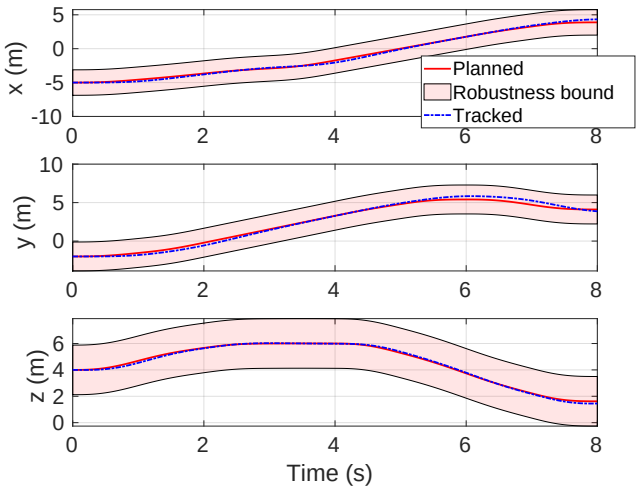

Fig. 5: Position vs time for planned trajectories of UAV 2 flying the timed reach-avoid mission, and after tracking them via the synthesized controller. The actual trajectories (tracked) are within the robustness bound of the planned trajectories, hence ensuring that the specification is satisfied.

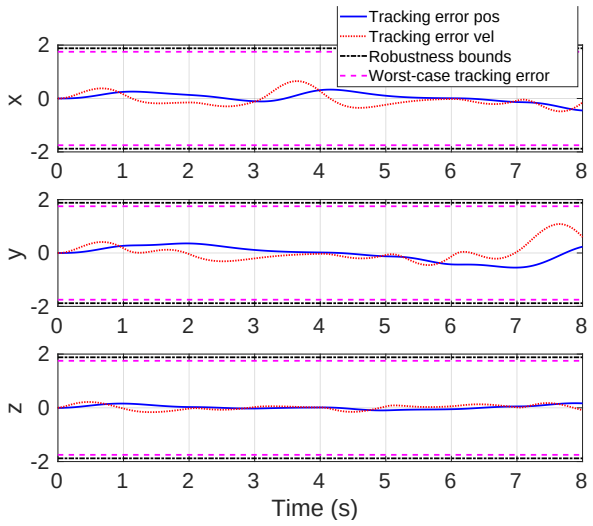

Fig. 6: Tracking errors for position and velocity in the 3 axes of motion for UAV 2 (reach-avoid). Note that the errors are contained within the worst-case bounds, which in turn are smaller than the bounds imposed by the STL robustness of the trajectories.

velocities and orientations. The trajectory planner generated trajectories of robustness $\rho_{\varphi_{\text{reach-avoid}}}(\hat{\mathbf{x}}) = 1.88$. The synthesized controller has a worst case tracking error (in the inf-norm sense) of $\delta = 1.75$, implying the trajectories can be tracked without violating (1). Ω_η^V is shown with the blue ellipsoids.

Figure 5 shows the planned and tracked positions for UAV 2, and figure 6 shows the tracking errors, demonstrating that the tracking is indeed good enough to ensure satisfaction of the underlying STL specification. This holds for both UAVs, for tracking of both positions and velocities, however we do not present additional figures due to lack of space. A video playback of the simulation can be found at `http://bit.ly/TimedRA`

B. *Four-UAV Multi-mission example*

Here, the four UAVs are tasked with performing two types of mission in the workspace shown in figure 7. Two UAVs (1 and 2) have to perform a patrolling mission, twice visiting Zone 1 and Zone 2 between pre-defined time intervals. The other two UAVs (3 and 4) have to visit a Deliver region within the first two seconds to drop off a package, and then reach the Base region. For safety, all UAVs must avoid two Unsafe regions and maintain pairwise separation of at least 0.2m. This mission has a horizon of 20 seconds, and is represented in STL as:

$$
\begin{aligned}
\varphi_{\text{multi-mission}} = {} & \wedge_{i=1}^{2}(\Diamond_{[0,5]}(p_i \in \text{Zone 1}) \wedge \Diamond_{[5,10]}(p_i \in \text{Zone 2}) \\
& \wedge \Diamond_{[10,15]}(p_i \in \text{Zone 1}) \wedge \Diamond_{[15,20]}(p_i \in \text{Zone 2})) \wedge \\
& \wedge \wedge_{i=3}^{4}(\Diamond_{[0,10]}(p_i \in \text{Deliver}) \wedge \Diamond_{[10,20]}(p_i \in \text{Base})) \wedge \\
& \wedge \wedge_{i=1}^{4}(\Box_{[0,20]}(\neg p_i \in \text{Unsafe}_1) \wedge \Box_{[0,20]}(\neg p_i \in \text{Unsafe}_2)) \\
& \wedge_{i,j,i\neq j} \Box_{[0,20]}(||p_i - p_j|| \geq 0.2)
\end{aligned}
$$

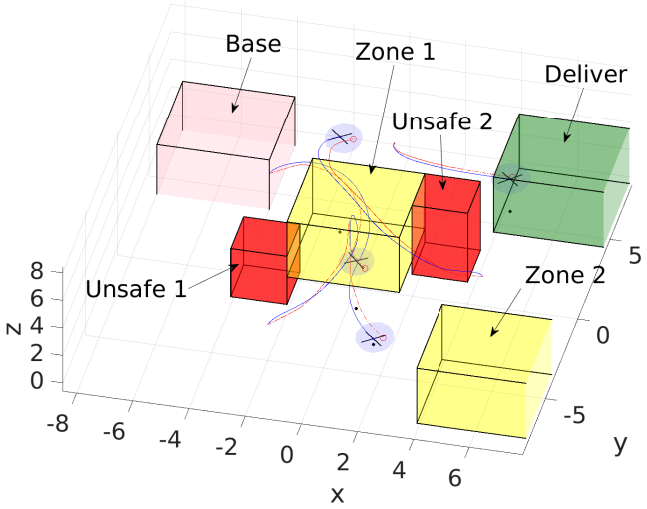

Fig. 7: Workspace for the four UAV multi-mission example. Simulation videos are at `http://bit.ly/MultiMission4UAV`

Results: The trajectory planner generates trajectories that satisfy the specifications with a robustness $\rho_{\varphi_{\text{multi-mission}}}(\hat{\mathbf{x}}) = 1.85 \geq \delta = 1.75$, ensuring that the trajectories can be tracked without violating the specification. Ω_η^V is shown with the blue ellipsoids in figure 7. The planned and tracked trajectories are shown with red and blue curves, respectively. A video playback of the simulation can be found at `http://bit.ly/MultiMission4UAV`

VII. Conclusion

Summary: In this paper, we present an approach for planning and control of (multiple) UAVs carrying out tasks specified using Signal Temporal Logic (STL). This is done by co-designing the trajectory generator and the controller that tracks these trajectories. We showed how this co-design allows us to take into account kinematic constraints, the full non-linear UAV dynamics, and the tracking error while

generating and flying out trajectories that satisfy the STL specification in continuous time. Through simulations, we show the the co-design results in trajectories that satisfy the STL specification and a synthesized controller that tracks them within prescribed bounds.

Limitations and future work: In practice, the computed worst-case tracking bounds for the synthesized controller can be very conservative. This is noticed in the simulations where the tracking error never approaches these bounds. This could result in the trajectory planning optimization being unable to obtain a high enough robustness value with respect to the given STL specification for Theorem 4.1 to hold, e.g. in cases where the UAVs need to fly through narrow openings, and restricts the application of our approach to such settings. To overcome this, we aim to improve our technique by *tightening* the STL specification by a state-dependent function of the tracking error in the planning phase, instead of using the worst-case bounds on tracking error. Additionally, our hierarchical planning and control framework is limited to specifications only over the translational states of the system and cannot handle temporal specifications on the orientations of the UAVs.

REFERENCES

[1] Y. V. Pant, H. Abbas, R. A. Quaye, and R. Mangharam, "Fly-by-logic: control of multi-drone fleets with temporal logic objectives," in *Proceedings of the 9th ACM/IEEE International Conference on Cyber-Physical Systems*, pp. 186–197, IEEE Press, 2018.

[2] Y. V. Pant, H. Abbas, and R. Mangharam, "Smooth operator: Control using the smooth robustness of temporal logic," in *Control Technology and Applications, 2017 IEEE Conf. on*, pp. 1235–1240, IEEE, 2017.

[3] L. Lindemann and D. V. Dimarogonas, "Control barrier functions for signal temporal logic tasks," *IEEE Control Systems Letters*, 2019.

[4] I. Saha, R. Rattanachai, V. Kumar, G. J. Pappas, and S. A. Seshia, "Automated composition of motion primitives for multi-robot systems from safe ltl specifications," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2014.

[5] P. Parrilo, "Structured semidefinite programs and semialgebraic geometry methods in robustness and optimization," 2000.

[6] V. Raman, A. Donze, M. Maasoumy, R. M. Murray, A. Sangiovanni-Vincentelli, and S. A. Seshia, "Model predictive control with signal temporal logic specifications," in *53rd IEEE Conf. on Decision and Control*, pp. 81–87, Dec 2014.

[7] S. Sadraddini and C. Belta, "Robust temporal logic model predictive control," in *Allerton conference*, September 2015.

[8] A. Bemporad and M. Morari, "Control of systems integrating logic, dynamics, and constraints," *Automatica*, vol. 35, no. 3, pp. 407–427, 1999.

[9] 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 preprint arXiv:1809.06746*, 2018.

[10] 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," in *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*, pp. 1517–1522, Dec. 2017.

[11] S. Singh, M. Chen, S. L. Herbert, C. J. Tomlin, and M. Pavone, "Robust tracking with model mismatch for fast and safe planning: an sos optimization approach," *arXiv preprint arXiv:1808.00649*, 2018.

[12] A. Girard and G. J. Pappas, "Hierarchical control system design using approximate simulation," *Automatica*, vol. 45, no. 2, pp. 566 – 571, 2009.

[13] S. W. Smith, M. Arcak, and M. Zamani, "Approximate abstractions of control systems with an application to aggregation," *Automatica*, vol. 119, p. 109065, 2020.

[14] S. W. Smith, H. Yin, and M. Arcak, "Continuous abstraction of nonlinear systems using sum-of-squares programming," in *2019 IEEE 58th Conference on Decision and Control (CDC)*, pp. 8093–8098, 2019.

[15] P.-J. Meyer, H. Yin, A. H. Brodtkorb, M. Arcak, and A. J. Sørensen, "Continuous and discrete abstractions for planning, applied to ship docking," *arXiv e-prints*, p. arXiv:1911.09773, Nov. 2019.

[16] H. Yin, M. Bujarbaruah, M. Arcak, and A. Packard, "Optimization based plannertracker design for safety guarantees," in *2020 American Control Conference (ACC)*, pp. 5194–5200, 2020.

[17] O. Maler and D. Nickovic, *Monitoring Temporal Properties of Continuous Signals*. Springer Berlin Heidelberg, 2004.

[18] G. Fainekos and G. Pappas, "Robustness of temporal logic specifications for continuous-time signals," *Theor. Computer Science*, 2009.

[19] F. Sabatino, "Quadrotor control: modeling, nonlinearcontrol design, and simulation," 2015.

[20] M. W. Mueller, M. Hehn, and R. DÁndrea, "A computationally efficient motion primitive for quadrocopter trajectory generation," in *IEEE Transactions on Robotics*, 2015.

APPENDIX

For the constraints of the trajectory planning optimization (9), consider the min-jerk trajectory segment [20], of time duration T, between a waypoint $\hat{p}^j = [\hat{p}_x^j, \hat{p}_y^j, \hat{p}_z^j]^\top$ with velocity $\hat{v}^j = [\hat{v}_x^j, \hat{v}_y^j, \hat{v}_z^j]^\top$, and the next waypoint with desired position $\hat{p}^{j+1} = [\hat{p}_x^1, \hat{p}_y^1, \hat{p}_z^1]^\top$. We define the functions [1], for $t \in [0, T]$:

$$\begin{aligned} K_3(t) &= (90t^4)/(48T^5) - (90t^3)/(12T^4) + (30t^2)/(4T^3) \\ K_4(t) &= (90t^3)/(12T_f^5) - (90t^2)/(4T^4) + (30t)/(2T^3) \end{aligned} \tag{13}$$

Let $t' = \text{argmax}_{t\in[0,T]} K_t(t)$. We can now define the constraints (for each UAV) that ensure velocity and acceleration are within bounds $[v_{\min}, v_{\max}]$ and $[a_{\min}, a_{\max}]$ respectively, for each axis of motion l:

$$\begin{aligned} \text{LB}_{\text{vel}}(\hat{v}_l^j) &= (v_{\min} - (1 - TK_3(T))\hat{v}_l^j)/K_3(T) \\ \text{UB}_{\text{vel}}(\hat{v}_l^j) &= (v_{\max} - (1 - TK_3(T))\hat{v}_l^j)/K_3(T) \\ \text{LB}_{\text{acc}}(\hat{v}_l^j) &= T\hat{v}_l^j + a_{\min}/K_4(t') \\ \text{UB}_{\text{acc}}(\hat{v}_l^j) &= T\hat{v}_l^j + a_{\max}/K_4(t') \end{aligned} \tag{14}$$

Combining these constraints for all axis of motion gives x, y, z the velocity and acceleration constraints in the optimization (9) of the form $\text{LB}_{\text{vel}}(\hat{v}^j) = [\text{LB}_{\text{vel}}(\hat{v}_x^j), \text{LB}_{\text{vel}}(\hat{v}_y^j), \text{LB}_{\text{vel}}(\hat{v}_z^j)]$ and similarly for the upper bound for velocities and upper/lower bounds for acceleration. These constraints are such that:

$$\begin{aligned} &\text{LB}_{\text{vel}}(\hat{v}^j) \le \hat{p}^{j+1} - \hat{p}^j \le \text{UB}_{\text{vel}}(\hat{v}^j) \\ &\Rightarrow \hat{v}_l^j \in [v_{\min}, v_{\max}]\, \forall t \in [0, T],\ \forall l \in \{x, y, z\}, \text{ and,} \\ &\text{LB}_{\text{acc}}(\hat{v}^j) \le \hat{p}^{j+1} - \hat{p}^j \le \text{UB}_{\text{acc}}(\hat{v}^j) \\ &\Rightarrow \hat{a}_l^j \in [a_{\min}, a_{\max}]\, \forall t \in [0, T],\ \forall l \in \{x, y, z\} \end{aligned} \tag{15}$$

Finally, the constraint that ensures bounded jerk $\hat{\mathbf{j}}_l(t) \in [j_{\min}, j_{\max}]\, \forall t \in [0, T]$ for each axis of motion l are:

$$\begin{aligned} \text{LB}_{\text{jerk}}(\hat{v}_l^j) &= \min((2T^3/30)j_{\min} - 0.5T\hat{v}_l^j,\ (2T^3/30)j_{\min} + T\hat{v}_l^j) \\ \text{UB}_{\text{jerk}}(\hat{v}_l^j) &= \min((2T^3/30)j_{\max} - 0.5T\hat{v}_l^j,\ (2T^3/30)j_{\max} + T\hat{v}_l^j) \end{aligned}$$

The proof follows a similar construction as the proofs in [1]. Note, these constraints are linear in the position waypoints, which are the variables of the trajectory planning optimization (9).