Transportation Research Part C 130 (2021) 103275

Contents lists available at ScienceDirect

Transportation Research Part C

journal homepage: www.elsevier.com/locate/trc

FADS: A framework for autonomous drone safety using temporal logic-based trajectory planning☆

Yash Vardhan Pant [a,*], Max Z. Li [b], Alena Rodionova [c], Rhudii A. Quaye [c], Houssam Abbas [d], Megan S. Ryerson [e,c], Rahul Mangharam [c]

[a] Department of Electrical Engineering and Computer Science, University of California, Berkeley, CA, USA
[b] Department of Aeronautics and Astronautics, Massachusetts Institute of Technology, Cambridge, MA, USA
[c] Department of Electrical and Systems Engineering, University of Pennsylvania, Philadelphia, PA, USA
[d] Department of Electrical and Computer Engineering, Oregon State University, Corvalis, OR, USA
[e] Department of City and Regional Planning, University of Pennsylvania, Philadelphia, PA, USA

ARTICLE INFO

Keywords:
Unmanned Aerial Systems
Urban Air Mobility
Signal Temporal Logic
Robust Trajectory Lanning

ABSTRACT

In this work, we present an integrated Framework for Autonomous Drone Safety (FADS). The demand for safe and efficient mobility of people and goods is growing rapidly, in line with the growth in population in US urban centers. In response, new technologies to meet these urban mobility demands are also rapidly maturing in preparation for future full-scale deployment. As surface congestion increases and the technology surrounding unmanned aerial systems (UAS) matures, more people are looking to the urban airspace and Urban Air Mobility (UAM) as a piece of the puzzle to promote mobility in cities. However, the lack of coordination between UAS stakeholders, federal UAS safety regulations, and researchers developing UAS algorithms continues to be a critical barrier to widespread UAS adoption. FADS takes into account federal UAS safety requirements, UAM challenge scenarios, contingency events, as well as stakeholder-specific operational requirements. FADS formalizes these requirements, through Signal Temporal Logic (STL) representations, and a trajectory planning optimization for multi-rotor UAS fleets guarantees robust and continuous-time satisfaction of the requirements and mission objectives. The intuitive FADS user interface makes it easy to plan missions in a variety of environments; we demonstrate this through several rural and urban environment-based case studies. FADS holistically integrates high-level stakeholder objectives with low-level trajectory planning; combined with a user-friendly interface, FADS reduces the complexity of stakeholder coordination within the UAM context.

1. Introduction

The rapid technological developments in the domain of unmanned aerial systems (UAS) have led federal and local governments to wrestle with three diverging yet intertwined forces related to UAS development: (1) the increase in accessibility and affordability of UAS lead to increased private-user and commercial interests in UAS technology (Heaton et al., 2018; New York Times, 2018); (2) the

☆ This article belongs to the Virtual Special Issue on IG005579: VSI: VSI:UAM.

* Corresponding author.
E-mail addresses: yashpant@berkeley.edu (Y.V. Pant), maxli@mit.edu (M.Z. Li), nellro@seas.upenn.edu (A. Rodionova), quayerhu@seas.upenn.edu (R.A. Quaye), habbas@seas.upenn.edu (H. Abbas), mryerson@design.upenn.edu (M.S. Ryerson), rahulm@seas.upenn.edu (R. Mangharam).

https://doi.org/10.1016/j.trc.2021.103275
Received 15 October 2020; Received in revised form 7 May 2021; Accepted 18 June 2021
Available online 30 July 2021
0968-090X/© 2021 Elsevier Ltd. All rights reserved.

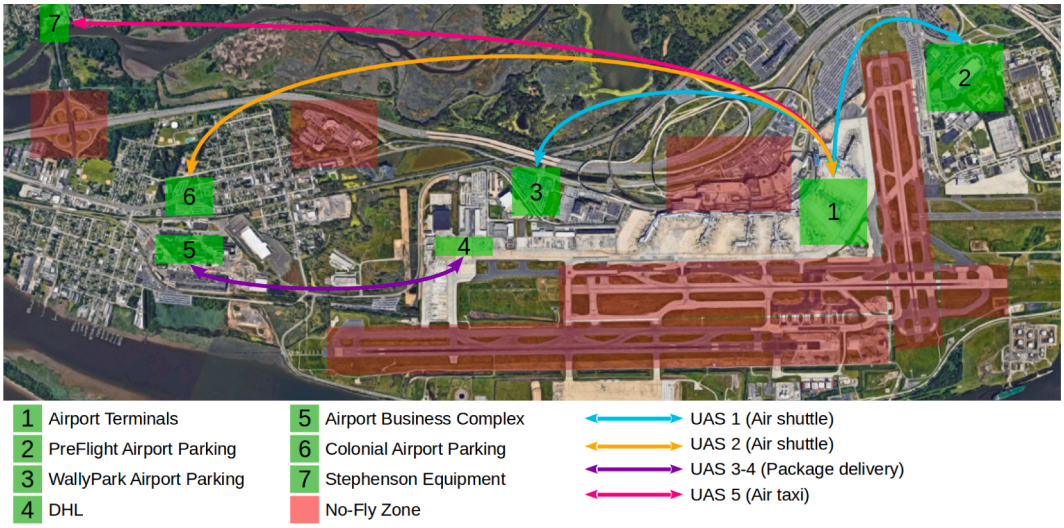

Fig. 1. UAM operating concept in the vicinity of a major hub airport (Philadelphia International Airport). Here, five UAS carrying people and goods are tasked with three heterogeneous missions. Same-color arrows indicate paths and locations of regions to be visited by UAS. We elaborate on this urban airspace case study in Section 7.

proliferation of UAS technology raises safety and ethical-use concerns, leading to a patchwork of regulations and advisories issued by the Federal Aviation Administration (FAA) (Federal Aviation Administration, 2013; Federal Aviation Administration, 2015); (3) the haphazard but heavily restrictive regulations on UAS operations (Office of the Federal Register, 2018; Federal Aviation Administration, 2018), combined with difficulties in enforcement and possible security vulnerabilities (Rodday et al., 2016), create an environment where widespread adoption of UAS for multipurpose missions is unlikely. While FAA forecasts for UAS purchases highlight a vigorously growing market (1.9 million in 2016 to 4.3 million by 2020 for hobbyist UAS purchases, and 0.6 million in 2016 to 2.7 million by 2020 for commercial UAS purchases) (Federal Aviation Administration, 2016), due to the interactions brought forth by the three diverging trends, the US public sector has yet to see a stable application of UAS technology outside of limited-scope trials, despite its potential for large advancements in efficiency and functionality in an array of applications spanning package delivery, agriculture operations, and transportation (Hamilton, 2018; Crown Consulting, 2018).

The interest in the urban airspace, or urban air mobility (UAM), is strong given that the urban airspace is replete with salient applications of UAS and UAS swarm technologies (The National Academies, 2020). However, the combined obstacles of a lack of technological and legislative certifications of UAS operational safety hinder this vision for a low-altitude, high-density urban UAS airspace. In this work, we tackle these obstacles via a two-pronged approach: (1) a short-term methodological solution that generates robust and safe UAS trajectories, and (2) a formalized database of UAS safety requirements that serves to verify other autonomous UAS algorithms.

Example 1. (*Multiple heterogeneous missions in an urban airspace*) We consider an example of UAS operations[1] in an urban environment consisting of a major hub airport, sensitive airport-related infrastructure (e.g., runways), along with nearby office buildings and major traffic intersections. Here, multiple UAS conduct heterogeneous missions within a much more constrained and complex low-altitude airspace. This urban case study represents a vision for what UAM could resemble: multiple UAS carrying out missions involving last-mile package delivery, aerial surveillance, and shuttling people on fixed routes (Hamilton, 2018; Crown Consulting, 2018). Fig. 1 shows the mission workspace and the regions that define the mission profiles. Specifically, five UAS carry out three types of missions within this urban infrastructure environment:

1. UAS 1 and 2 are autonomous air shuttles that ferry passengers between different parking areas and the airport terminal.
2. UAS 3 and 4 perform last-mile package delivery from the DHL ramp area to the airport business complex.
3. UAS 5 is an autonomous air taxi that picks up passengers from the terminal and takes them to their desired location.

The UAS have to stay away from the no-fly zones corresponding to the runways, airport infrastructure, and regions with heavy pedestrian or surface traffic as well as maintain separation from each other. In addition, UAS altitudes are capped below an altitude ceiling in order to avoid interfering with commercial flights. Later sections describe how these requirements are represented in a mathematically sound manner by encoding them as a Signal Temporal Logic (STL) specification. We describe this case study in detail in Section 7, and demonstrate how our framework can successfully generate safe, robust trajectories for the UAS to complete their

[1] Videos of simulations in this paper can be found at https://www.youtube.com/playlist?list=PL10P-R0IRhJw2iUhVlLz_IJAdfJUyhGcv

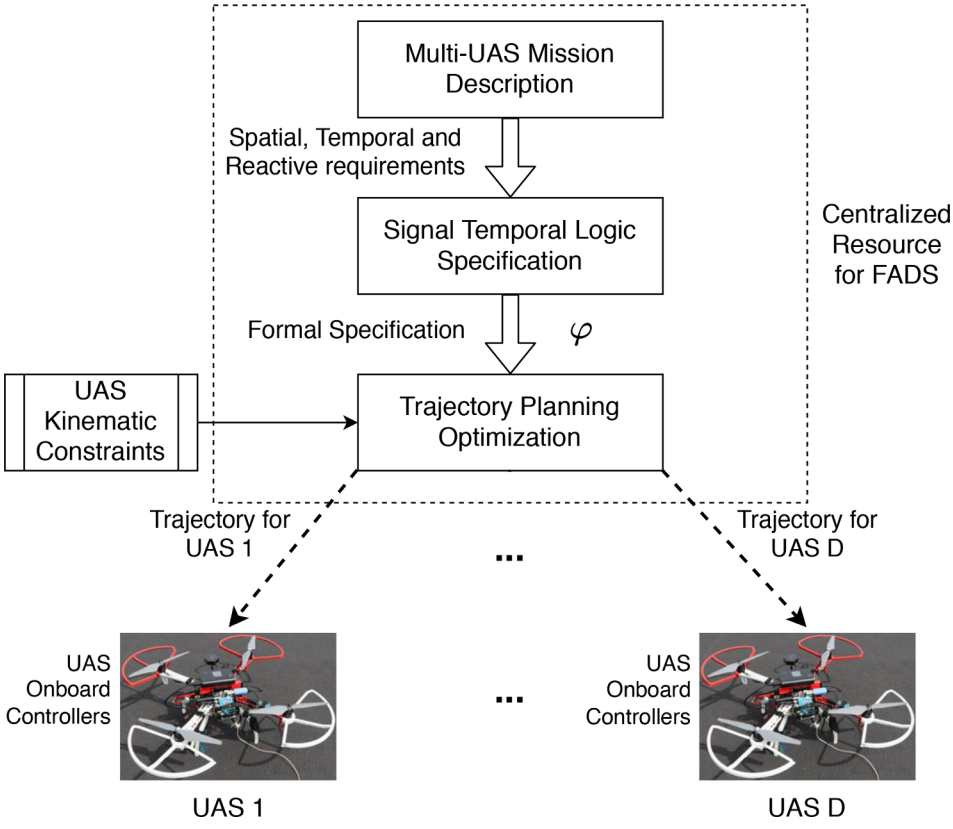

Fig. 2. Outline of our framework, FADS. The approach formalizes mission and operational requirements using Signal Temporal Logic, and then solves an optimization to generate trajectories for each UAS in the mission such that they robustly satisfy the mission.

missions.

1.1. Motivation: A unified model for UAS deployment

The high-level process of eventual UAS deployment can be summarized by four stages (National Aeronautics and Space Administration, 2018): (1) research and development of low-level UAS flight envelopes, trajectory planners, and flight controllers; (2) standardization of operational constraints such as separation standards and contingency procedures; (3) simulation-based and live UAS flight tests and demonstrations that provide feedback in order to assess and re-calibrate prior stages; and (4) safe deployment of UAS technologies in real-world settings for private and commercial purposes. Our work presents a Framework for Autonomous Drone Safety (FADS), addressing the disconnect between stages (1) and (2) that we identify through a literature review. In addition, FADS also provides a simulation environment where users can load custom UAS mission profiles and save the mission profiles to be loaded for later use. These case studies can be used as templates for designing other mission profiles, complete with mission goals, constraints, and other customizable parameters specified through *Signal Temporal Logic* (STL) (Section 3.1). Finally, through the built-in simulation environment, stage (3) testing can take place within FADS as well, the results of which can further inform necessary adjustments to the two previous stages.

Furthermore, in order to establish a roadmap for the four aforementioned stages, NASA has proposed "grand challenges" [2] that seek to better facilitate the translation process from UAS research and technological developments to real-world deployment (National Aeronautics and Space Administration, 2018). Specifically, these grand challenge scenarios were created to benchmark UAS vehicular performance as well as their performance when integrated into the urban airspace. We were motivated by many of the predefined scenarios from "Phase 2" of the NASA UAM grand challenge, and FADS directly addresses the following grand challenge scenarios:

- Construction of a simulated UAM airspace that complies with current regulations and concepts of operations (*NASA UAM grand challenge scenario 2*)

[2] We note that the NASA UAM grand challenge has since been renamed as the *Advanced Air Mobility National Campaign* (National Aeronautics and Space Administration, 2020).

- Trajectory robustness criterion to satisfy contingency and redundancy requirements (*NASA UAM grand challenge scenarios 2 and 5*)
- Trajectory and flight planning taking into account airspace, vehicular, and air traffic management constraints (*NASA UAM grand challenge scenario 3*)
- Separation and conflict resolution (*NASA UAM grand challenge scenario 8*)
- Mission Task Elements (MTEs), developed jointly with the FAA, that benchmark vehicular performance during various phases of flight such as climb/cruise/descent and contingency landings (*All scenarios in NASA UAM grand challenge*)

1.2. Overview of FADS

Fig. 2 shows an overview of our Framework for Autonomous Drone Safety (FADS). The given mission and operational requirements are first mathematically represented using Signal Temporal Logic[3] (Section 4). We then formulate and solve a trajectory planning optimization (Section 5) to select (time-stamped) waypoints and generate trajectories for each UAS in the mission such that they robustly satisfy the STL specification. These trajectories are sent to the UAS, which use off-the-shelf position and velocity tracking and low-level (attitude) controllers (Pant et al., 2021) to track them. The development of these controllers is beyond the scope of this work.

1.3. Contributions

The main contributions of our work are as follows:

1. FADS enables us to take into account mission requirements, possibly over a fleet of UAS, as well as high-level operational requirements (e.g., FAA Part 107 and other regulations (Federal Aviation Administration, 2013; Federal Aviation Administration, 2018; Federal Aviation Administration, 2020)) in a mathematically unambiguous manner by representing them using STL.

Example 2For example, a constraint on vehicular performance imposed by Federal Aviation Administration (2018) stipulates that the ground speed of the UAS must be less than or equal to 44 meters per second during flight. This high-level constraint maps one-to-one with the following mission planning statement in STL: $\varphi = \square_{[0,T]}(\|v(t)\| \leqslant 44)$. Here, $v(t) \in \mathbb{R}^3$ is the velocity of the UAS at time t, and the *always* ($\square$) operator requires the UAS speed to be $\|v(t)\| \leqslant 44\, ms^{-1}, \forall t \in [0, T]$.

More details on STL are in Section 3, and present instances of how UAS operating requirements can be represented mathematically using STL in Section 4.

2. The trajectory planning optimization that is at the heart of our approach ensures that: a) All UAS satisfy these requirements in continuous time, b) the generated trajectories respect kinematic constraints for each, c) the *robustness* measure associated with these trajectories is maximized such that bounded deviations from these planned trajectories do not violate the mission or operational requirements.
3. Market feasibility and acceptability studies regarding the adoption of UAM have emphasized the importance of specific use cases when evaluating various UAM concepts (Hamilton, 2018; Crown Consulting, 2018). Through two case studies in urban and rural scenarios, we show the applicability of our approach as well as its ability to generate trajectories for STL specifications defined over long time horizons (of the order of minutes) that are too complex for state-of-art Mixed Integer Programming-based tools.
4. Finally, we provide an easy-to-use Graphical User Interface and implementation of our method[4] that enables the use of our approach without requiring in-depth familiarity with Temporal Logics or the underlying trajectory planning optimization.

This paper builds upon our work in the conference publications (Pant et al., 2018; Pant et al., 2019) and shows how this approach to trajectory planning can be used for a wide variety of UAM missions when the workspace, e.g., static obstacles, regions to visit, and paths of uncontrolled agents, is known *a priori*. Furthermore, we emphasize that our approach can handle missions with high complexity in terms of the time scale (e.g., 11 min for the case study in Section 7, corresponding to 6600 time steps at 10 Hz), number of UAS performing different missions, and behavioral requirements. To the best of our knowledge, there is no other approach (also see Section 2) that can generate trajectories that satisfy the STL specifications corresponding to these missions, as will be seen in Sections 6 and 7.

1.4. Paper organization

The remainder of the paper is organized as follows: We examine a selection of the pertinent literature related to UAS trajectory planning, STL, and current UAS deployment cases in Sections 2.1, 2.2, and 2.3, respectively. We then present the theoretical framework for the low-level planning and STL specifications in Section 3 and how these specifications are mapped to high-level safety and mission objectives in Section 4. We introduce and discuss the results of two case studies in Sections 6 (rural airspace) and 7 (urban airspace). A summary of our FADS framework and future research directions is presented in Section 8.

[3] Translating plain-text statements to STL is beyond the scope of this work, however simple missions can be graphically represented and translated to STL using our work in Pant et al. (2019). Also see Section 5.2.

[4] https://github.com/yashpant/FlyByLogic.

1.5. Frequently used notation

We use $p = [p_x, p_y, p_z]^\intercal \in \mathbb{R}^3$ to denote the position of a UAS in 3-D co-ordinate space (vector or matrix transpose denoted by $\intercal$), and correspondingly $v \in \mathbb{R}^3$ and $a \in \mathbb{R}^3$ for the velocity and acceleration respectively. Other notations are introduced as required in the following sections.

2. Literature review

The three areas that we cover within this literature review are: (1) UAS trajectory planning and conflict resolution; (2) temporal logic and mission planning; (3) examples of real-world UAS deployments. Throughout each subsection, we elaborate on how FADS either improves, extends, and/or combines past research and operational concepts.

2.1. UAS trajectory planning and conflict resolution

There are numerous studies leveraging a variety of methods that tackle UAS-specific trajectory conflict resolution, detect-and-avoid algorithms, and trajectory optimization. UAS trajectory conflict resolution in a three-dimensional setting with multiple UAS is formulated in the context of an optimal control problem in Borrelli et al. (2006), where the optimal control problem is solved via two methods: As a nonlinear programming problem, and as a mixed integer linear programming problem. A mixed integer nonlinear program formulation of the trajectory optimization problem for UAS is presented in Ragi and Mittelmann (2017). The ability to do online as well as offline planning is introduced by Besada-Portas et al. (2010), where the trajectory planning is taken care of by evolutionary-type algorithms.

While resolving pairwise trajectory conflicts between two UAS is a crucial design feature for any trajectory planning model, the ability to take into account environmental obstacles is also important, particularly for UAS operating in urban settings. Previously, models such as the one presented in Mellinger et al. (2012) provide optimal trajectories for quad-rotor systems that take into account environmental obstacles, based on a mixed integer quadratic programming formulation. Similarly, (Lin and Saripalli, 2017) uses a sample-based trajectory planning heuristic that seeks to avoid collisions between the UAS and moving obstacles; these moving obstacles could include commercial aircraft and helicopters that share the urban airspace. FADS incorporates UAS-on-UAS separation and UAS-on-infrastructure (*i.e.* environmental obstacles) constraints with the required dimensions as indicated by current FAA specifications and the NASA UAM grand challenge metrics.

2.2. Temporal logic-based planning

The problem of planning and control for multi-agent systems has also been looked at through the lens of Temporal Logic. The most studied approach involves discretization of the workspace into a grid (Saha et al., 2014), and working with simplifying abstractions of the dynamics of the agents (Desai et al., 2017; Aksaray et al., 2016). The resulting solutions, which mostly deal with Linear Temporal Logic (LTL) specifications, have guarantees on correctness only on the discrete abstraction of the underlying continuous system. These methods also in general cannot deal with temporal operators with a bounded time horizon, limiting the kind of missions that can be expressed in them, e.g. a UAS must reach a region within the next 8 to 10 s cannot be specified with fragments of LTL that do not have the *next* operator, like LTL_X (Fainekos et al., 2005; Kloetzer and Belta, 2008).

The planning method in this paper does not rely on discretizing the behavior of the robotic agents, and can leverage the full expressivity of Signal Temporal Logic (STL) (Section 3.1), allowing us to specify bounded time requirements of the form above, and more (e.g. see Section 7). Our method relies on optimizing a notion of *robustness* (Fainekos, 2008) associated with the STL specification. The closest method is the sub-gradient based approach of (Abbas et al. (2014),Raman et al. (2014)), however it has only been applied to systems with safety properties. The method of Raman et al. (2014), which relies on encoding the STL specification as constraints in a Mixed-Integer Linear Program (MILP), can leverage the full expressiveness of STL and work with continuous dynamics. However it has been shown in previous work (Pant et al., 2017; Pant et al., 2018) that our approach is computationally faster and can scale to a higher number of UAS; In fact, we show in Section 7.3 that state-of-the-art MIP implementations time out for our case study scenarios, whereas FADS returns a solution within a reasonable time frame. Finally, unlike the other methods discussed, our method also offers continuous-time guarantees on the satisfaction of the STL specifications. A more detailed review of methods for planning and control with STL specifications can be found in Belta and Sadraddini (2019).

2.3. Examples of real-world UAS deployments

Currently, deployments of UAS in live flight situations typically are used to validate research methodologies such as the ones surveyed in Section 2.1, or in a severely limited setting as a means to test commercial applications. In terms of validation of novel UAS trajectory planning algorithms, (Alejo et al., 2012) utilized multiple UAS in order to test and verify their speed profile planning-based conflict resolution methodology. Mellinger et al. (2012) also validated their Mixed Integer Quadratic Program-based optimal trajectory generator via live UAS within an environment with obstacles.

A few companies and industries are also testing the capabilities of various UAS platforms to augment their products and services. However, most of these deployments are oftentimes confined to a small geographical area and not made widely available. A prominent

example is the usage of UAS platforms to deliver food and beverages, limited to select US college campuses (CNN, 2016). Short-distance and urban-to-rural package delivery services via UAS are also undergoing live testing and development; unlike the food and beverage delivery example, these package delivery services are being tested within a closed environment with no potential customers in the loop (CNBC, 2019).

In the next section, we expand on the main contribution of our FADS framework: bundling the low-level trajectory planning and mission specifications with high-level objectives and robustness interpretations, all within the constraints set by the FAA and the NASA UAM grand challenge. We argue that architecture similar to the mission planning interface in FADS will be critical to ensure safety within the urban airspace where multiple simultaneous UAS missions are being carried out.

3. Preliminaries on signal temporal logic

Let $x \in X \subset \mathbb{R}^n$ be the state of a system. In this paper we deal with discrete-time representations of continuous-time UAS trajectories of these states, so we introduce notation to make the sampling period explicit. Let $dt \in \Re_{>0}$ be a sampling period and $T \in \Re_{>0}$ be a trajectory duration. We write $[0 : dt : T] = (0, dt, 2dt, \ldots, (H-1)dt)$ for the sampled time interval s.t. $(H-1)dt = T$ (we assume T is divisible by $H-1$). A discrete-time *trajectory* is then a sequence of states $\mathbf{x} = (x_0, x_{t_1} \ldots, x_{t_{H-1}})$ s.t. for all $t \in [0 : dt : T], x_t \in X$. Given a time domain $\mathbb{T} = [0 : dt : T]$, the signal space $X^{\mathbb{T}}$ is the set of all signals $\mathbf{x} : \mathbb{T} \to X$. For an interval $I \subset \Re_+$ and $t \in \Re_+$, set $t + I = \{t + a \mid a \in I\}$. The max operator is written $\sqcup$ and min is written $\sqcap$. As will be defined in the following subsection, a Signal Temporal Logic *specification* $\varphi : X^{\mathbb{T}} \to \{\top, \bot\}$ is a boolean-valued function that takes in the trajectory[5] $\mathbf{x}$ and evaluates it over the time domain $\mathbb{T}$ to return the boolean true $\top$ when the specification is satisfied by $\mathbf{x}$, or false $\bot$ otherwise.

3.1. Signal Temporal Logic (STL)

We wish to generate UAS trajectories that satisfy a specification expressed in Signal Temporal Logic (STL) (Maler and Nickovic, 2004; Donzé and Maler, 2010). STL is a logic that allows the succinct and unambiguous specification of a wide variety of desired system behaviors over time, such as Abbas and Fainekos, 2013 "The UAS reaches the goal within 10 time units while always avoiding obstacles" and "While the UAS is in Zone 1, it must obey that zone's altitude constraints". Formally, let $M = \{\mu_1, \ldots, \mu_L\}$ be a set of real-valued functions of the state $\mu_k : X \to \Re$. For each μ_k define the *predicate* $p_k := \mu_k(x) \geqslant 0$. Set $AP := \{p_1, \ldots, p_L\}$. Thus each predicate defines a set, namely p_k defines $\{x \in X \mid f_k(x) \geqslant 0\}$. Let $I \subset \Re$ denote a non-singleton interval, $\top$ the Boolean True, p a predicate, $\neg$ and $\wedge$ the Boolean negation and AND operators, respectively, and $\mathcal{U}$ the Until temporal operator. An STL formula φ is built recursively from the predicates using the following grammar:

$$\varphi \;:=\; \top \mid p \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 \mathcal{U}_I \varphi_2$$

Informally, $\varphi_1 \mathcal{U}_I \varphi_2$ means that φ_2 must hold at some point in I, and *until* then, φ_1 must hold without interruption. The disjunction ($\vee$), implication ($\Rightarrow$), Always ($\Box$) and Eventually ($\Diamond$) operators can be defined using the above operators. Formally, the *pointwise semantics* of an STL formula φ define what it means for a system trajectory $\mathbf{x}$ to satisfy φ.

Definition 3.1. (*STL semantics*) Let $\mathbb{T} = [0 : dt : T]$. The boolean truth value of φ w.r.t. the discrete-time trajectory $\mathbf{x} : \mathbb{T} \to X$ at time $t \in \mathbb{T}$ is defined recursively.

$$\begin{aligned}
(\mathbf{x}, t) \vDash \top &\Leftrightarrow \quad \top \\
\forall p_k \in AP, (\mathbf{x}, t) \vDash p_k &\Leftrightarrow \quad \mu_k(x_t) \geqslant 0 \\
(\mathbf{x}, t) \vDash \neg\varphi &\Leftrightarrow \quad \neg(\mathbf{x}, t) \vDash \varphi \\
(\mathbf{x}, t) \vDash \varphi_1 \wedge \varphi_2 &\Leftrightarrow \quad (\mathbf{x}, t) \vDash \varphi_1 \wedge (\mathbf{x}, t) \vDash \varphi_2 \\
(\mathbf{x}, t) \vDash \varphi_1 \mathcal{U}_I \varphi_2 &\Leftrightarrow \quad \exists t' \in [t + I] \cap \mathbb{T} \text{ s.t. } (\mathbf{x}, t') \vDash \varphi_2 \\
&\qquad \wedge \forall t'' \in [t, t') \cap \mathbb{T}, \;\; (\mathbf{x}, t'') \vDash \varphi_1
\end{aligned}$$

We say $\mathbf{x}$ *satisfies* φ if $(\mathbf{x}, 0) \vDash \varphi$.

All formulas that appear in this paper have bounded temporal intervals: $0 \leqslant \inf I < \sup I < +\infty$. To evaluate whether such a *bounded* formula φ holds on a given trajectory, only a finite-length prefix of that trajectory is needed. Its length can be upper-bounded by the *horizon* of φ, $hrz(\varphi) \in \mathbb{N}$, calculable as shown in Raman et al. (2014). For example, the horizon of $\Box_{[0,2]}(\Diamond_{[2,4]} p)$ is $2 + 4 = 6$: we need to observe a trajectory of, at most, length 6 to determine whether the formula holds.

Note: The bounded-time *always* ($\Box_I$) and *eventually* ($\Diamond_I$) operators[6] are used frequently in this paper. Here, $\Box_I \varphi$ is satisfied, i.e. evaluates to $\top$, if and only if the relation φ is $\top$ throughout the time interval I. Similarly, $\Diamond_I \varphi$ is satisfied when $\exists t \in I$ such that φ is $\top$, or the relation φ is satisfied at some time in the interval I.

[5] possibly a concatenation of trajectories of multiple UAS when required

[6] $I = [t_1, t_2]$ is a time interval where $0 \leqslant t_1 < t_2 < \infty$

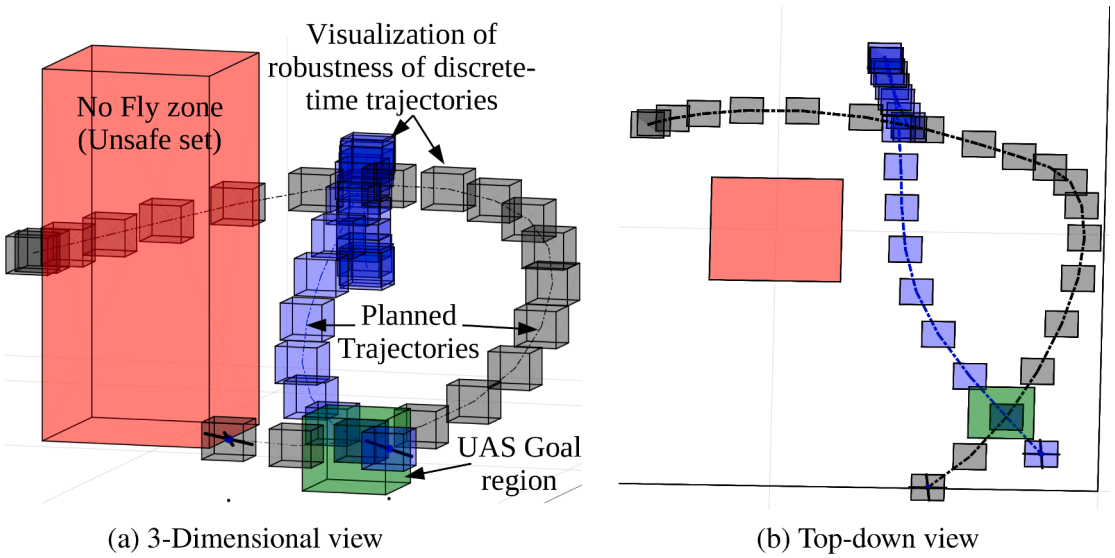

Fig. 3. Robustness value as a bound for tracking discrete time trajectories. For the given trajectories (dashed blue and black), their robustness ρ_φ with respect of a specification φ is such that φ is satisfied as long as the UAS deviate no more than ρ_φ from the trajectories while tracking them. For continuous time trajectories, this would define a *tube* around the trajectories, while in discrete time it would define a sequence of *boxes* as shown here.

3.2. Control using the robust semantics of STL

Designing a controller for the UAS such that the generated trajectories satisfy the STL formula φ is not always enough. In a dynamic environment, where the system must react to new unforeseen events, it is useful to have a margin of maneuverability. That is, it is useful to control the system such that we *maximize* our degree of satisfaction of the formula. When unforeseen events occur, the system can react to them without violating the formula. This degree of satisfaction can be formally defined and computed using the *robust semantics* of temporal logic (Donzé and Maler, 2010; Fainekos and Pappas, 2009).

Definition 3.2. (*Robustness (Donzé and Maler, 2010; Fainekos and Pappas, 2009)*) The *robustness* of STL formula φ relative to $\mathbf{x} : \mathbb{T} \to X$ at time $t \in \mathbb{T}$ is

$$\begin{aligned}
\rho_{\top}(\mathbf{x},t) &= +\infty \\
\rho_{p_k}(\mathbf{x},t) &= \mu_k(x_t)\ \forall p_k \in AP, \\
\rho_{\neg\varphi}(\mathbf{x},t) &= -\rho_{\varphi}(\mathbf{x},t) \\
\rho_{\varphi_1\wedge\varphi_2}(\mathbf{x},t) &= \rho_{\varphi_1}(\mathbf{x},t) \sqcap \rho_{\varphi_2}(\mathbf{x},t) \\
\rho_{\varphi_1 \mathcal{U}_I \varphi_2}(\mathbf{x},t) &= \sqcup_{t' \in [t+I]\cap\mathbb{T}}\big(\rho_{\varphi_2}(\mathbf{x},t') \sqcap \\
&\quad \sqcap_{t'' \in [t,t')\cap\mathbb{T}} \rho_{\varphi_1}(\mathbf{x},t'')\big)
\end{aligned}$$

When $t = 0$, we write $\rho_\varphi(\mathbf{x})$ instead of $\rho_\varphi(\mathbf{x},0)$. The robustness is a real-valued function of $\mathbf{x}$ with the following important property.

Theorem 3.1. (*Fainekos and Pappas, 2009*) *For any $\mathbf{x} \in X^{\mathbb{T}}$ and STL formula φ, if $\rho_\varphi(\mathbf{x},t) < 0$ then $\mathbf{x}$ violates φ at time t, and if $\rho_\varphi(\mathbf{x},t) > 0$ then $\mathbf{x}$ satisfies φ at t. The case $\rho_\varphi(\mathbf{x},t) = 0$ is inconclusive.*

This robustness function is central to our approach, and we note two important observations related to it:

1. In order to satisfy the specification φ, we can compute trajectories $\mathbf{x}$ that *maximize* the robustness ρ_φ. As stated in the theorem above, φ is satisfied when the trajectories achieve positive robustness values. Our approach for doing so is presented in Section 5.
2. Another useful interpretation of robustness is that the larger $\rho_\varphi(\mathbf{x}^*,t)$, the more robust is the behavior of the system: intuitively, $\mathbf{x}^*$ can be disturbed and ρ_φ might decrease but not go negative. In fact, the amount of disturbance that $\mathbf{x}^*$ can sustain is precisely ρ_φ: that is, if $\mathbf{x}^* \vDash \varphi$, then $\mathbf{x}^* + e \vDash \varphi$ for all disturbances $e : \mathbb{T} \to X$ s.t. $\sup_{t\in\mathbb{T}} \|e(t)\| < \rho_\varphi(\mathbf{x}^*)$. Fig. 3 illustrates this, and an example to explain this follows.

3.3. Interpreting the robustness of STL

Example 3. (*Avoiding obstacles*) Consider a safety specification using the *always* ($\Box$) operator:

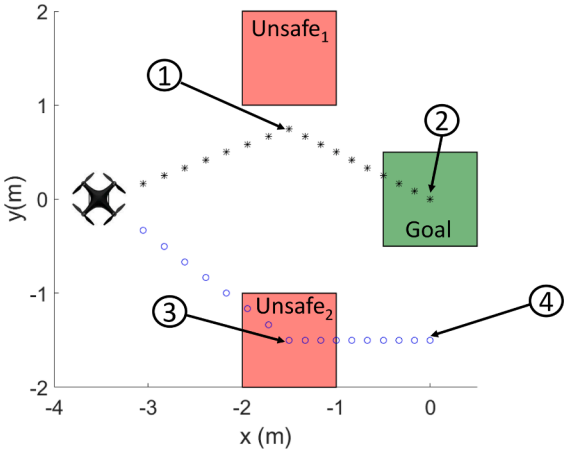

Fig. 4. This illustration shows a UAS and two trajectories, $\mathbf{x}_1$ and $\mathbf{x}_2$.

$$\varphi_{\text{safe}} = \square_{[0,T]}\neg(p \in \text{Unsafe}_1) \wedge \square_{[0,T]}\neg(p \in \text{Unsafe}_2) \tag{1}$$

This states the position p of the UAS should, in the time interval $[0, T]$, never be inside the region given by Unsafe_1 and it should also never be inside Unsafe_2. Fig. 4 shows these regions. Consider the trajectory $\mathbf{x}_1$, shown from time 0 to T seconds. As can be seen, the UAS does indeed avoid the unsafe regions and satisfies the specification, which by Theorem 3.1 implies that the robustness of this trajectory $\mathbf{x}$ with respect to the specification $\varphi_{\text{safe}}, \rho_{\varphi_{\text{safe}}}(\mathbf{x})$ is positive.

In order to further understand this robustness value, let us first compute it. The proposition $p \in \text{Unsafe}_1$ can be written in more details as $(p_x \leqslant -1) \wedge (-p_x \leqslant 2) \wedge (p_y \leqslant 2) \wedge (-p_y \leqslant -1)$. This comes from the representation of the set as a bounded axis-aligned polyhedron in $\mathbb{R}^2$. Following the robustness semantics of Definition 3.2 that states the robustness $\rho_{\varphi_1 \wedge \varphi_2} = \min(\rho_{\varphi_1}, \rho_{\varphi_2})$, the robustness of $p \in \text{Unsafe}$, evaluated at a single point in the trajectory, can be computed as $\rho_{\text{Unsafe}_1}(p) = \min(-1 - p_x, 2 + p_x, 2 - p_y, -1 + p_y)$. As an example, consider the point $[-1.5, 0.75]^\intercal$ marked by ① in Fig. 4. The robustness of this point w.r.t proposition $p \in \text{Unsafe}_1$ is $\rho_{\text{Unsafe}_1} = \min(0.5, 0.5, 1.25, -0.25) = -0.25$. This negative robustness implies that the point $[-1.5, 0.75]^\intercal$ does not satisfy the proposition $p \in \text{Unsafe}_1$, as seen in the figure.

Since a part of the safety specification φ_{safe} asks for $\neg(p \in \text{Unsafe}_1)$, the robustness of this proposition is simply the negative of the robustness of the proposition $p \in \text{Unsafe}_1$ (again see Definition 3.2), or 0.25. To then evaluate the robustness of $\square_{[0,T]}\neg(p \in \text{Unsafe}_1)$, following Definition 3.2, we need to compute the minimum of the robustness of the proposition $\neg(p \in \text{Unsafe}_1)$ over all points from time 0 to T in trajectory $\mathbf{x}_1$, i.e. $\min_{t \in [0,T]}(-\min(-1 - p_x(t), 2 + p_x(t), 2 - p_y(t), -1 + p_y(t)))$. For trajectory $\mathbf{x}_1$, this minimum is achieved by the point ①, hence the robustness of trajectory $\mathbf{x}_1$ w.r.t the specification $\square_{[0,T]}\neg(p \in \text{Unsafe}_1)$ is 0.25. Similarly we can compute the robustness of the specification $\square_{[0,T]}\neg(p \in \text{Unsafe}_2)$. The robustness of the safety specification φ_{safe} is then (using $\rho_{\varphi_1 \wedge \varphi_2} = \min(\rho_{\varphi_1}, \rho_{\varphi_2})$) given by minimum of the robustness of $\square_{[0,T]}\neg(p \in \text{Unsafe}_1)$ and $\square_{[0,T]}\neg(p \in \text{Unsafe}_2)$. For the trajectory $\mathbf{x}_1$, this value is achieved by the point ①, and is hence 0.25.

This value of 0.25 implies that each point in the trajectory $\mathbf{x}_1$ could be moved by at most 0.25 meters along any axis and still the trajectory would satisfy the specification φ_{safe}. Again, focusing on ① helps explain this. If we move ① along the y-axis by up to 0.25 meters, ① still does not enter the set Unsafe_1. Moving it 0.25 meters in the y-axis would bring it to the boundary of the unsafe set, and larger deviations would push it into the unsafe set, violating the requirement that the trajectory never enters this set.

3.4. Smooth approximation of STL robustness

The robustness function ρ_φ defined above is continuous, but not smooth due to the max/min operators in it. In (Pant et al., 2017), we defined a continuously differentiable approximation of robustness $\widetilde{\rho}_\varphi$ with the following property:

Theorem 3.2. (*Pant et al., 2017*) *For a STL specification φ, the smooth robustness function $\widetilde{\rho}_\varphi$ as defined in Pant et al. (2017) is continuously differentiable and $|\rho_\varphi(\mathbf{x}, t) - \widetilde{\rho}_\varphi(\mathbf{x}, t)| \leqslant \delta_\varphi$ where δ_φ can be pre-computed and is independent of the evaluation time t.*

We can use gradient-based approaches to maximize $\widetilde{\rho}_\varphi$, and we exploit this to generate trajectories[7] that satisfy φ, as described in Section 5.

[7] While we work with discrete time representations of continuous trajectories, Theorem 5.1 from our work in Pant et al. (2018) shows how we can still satsify φ in continuous time using our approach.

Table 1
High-level constraints and specifications extracted from FAA Part 107 and other federal regulations governing UAS operations (Federal Aviation Administration, 2013; Federal Aviation Administration, 2015; Federal Aviation Administration, 2018; Office of the Federal Register, 2018), with remarks indicating how they are accounted for in our case study implementations.

High-level constraints and specifications	Remarks
Altitude $\leqslant$400 feet (121 meters) AGL (above ground level) **OR** If altitude $>$ 400 feet, stay inside of 400 feet of a structure	Can be encoded as in (2) In our case studies, we follow the former (Altitude $\leqslant$400 feet) since we assume our operations have an altitude cap.
Groundspeed $\leqslant$100 mph (44 meters/s)	Encoded as a constraint in the optimization for trajectory planning (7)
Avoid manned aircraft at all times	Our approach can handle moving obstacles, given predictions of their trajectory, see: https://youtu.be/cmX7g6QN09c
Avoid airspace directly above people at all times	Observed in case study by marking airspace above busy intersections and populated areas as no-fly zones.
Avoid airspace underneath covered structure at all times (*Alternatively:* Constrain altitude selectively)	Observed in case study by marking convex hull of infrastructure as unsafe set.
IGNORE ALL ABOVE in emergency	The mission representation in STL allows for contingency operation "flags" to be triggered, e.g., (5).
NO INTER-UAS COLLISIONS at all times	Requirement of separation between all UAS is maintained through duration of mission, and is captured as a STL specification, e.g., (6).

4. Mapping mission and operational requirements to STL specifications

A key contribution of the FADS implementation is the integration of high-level constraints such as federally-mandated UAS operational regulations as well as spatio-temporal mission requirements into the low-level trajectory planning and synthesis process. Now that we have presented an overview of STL, control of dynamical systems using STL, and interpreting STL robustness, we proceed to *translate* the high-level requirements into formal STL specifications. We extract a list of important safety constraints and operational limitations, along with comments of how each constraint is taken care of within the case studies in this paper in Table 1.

In translating each row of constraints from Table 1 into its STL formulation, we first define some variables that act as placeholders for physical attributes within the UAS mission environment. Let $p \in \mathbb{R}^3$ be the position of the UAS, and p_z be its z component, i.e. its altitude. T is the mission horizon (in seconds), and G_i is a polyhedron corresponding to object i, e.g. a building. Let $G_i^{x,y}$ be the planar projection of G_i specified in Cartesian coordinates, and CG_i be its convex hull. Let $S_i^{x,y}$ be the planar projection of an area with pedestrians moving about, specified in Cartesian coordinates. Finally, let $E \in \{\top, \bot\}$ be a flag which becomes true $\top$ if there is an emergency, but is false during regular flight. The following are some examples of how UAS operational requirements specified by the FAA can be represented in STL:

The altitude-cap constraint (first row of Table 1) can be formulated as follows:

$$\begin{aligned}\varphi_1 \quad &= \square_{[0,T]}\big(\big(p_z \in [0, 121]\big) \vee \big(\big(p_z \geqslant 121\big) \\ &\wedge \vee_i (p \in (\mathbf{G}_i \oplus 121))\big)\big)\end{aligned} \tag{2}$$

Here, $\oplus$ is the Minkowski sum and $G_i \oplus 121$ results in the polyhedron G_i being extended in all directions by 121 meters (121 m). For this specification, we only consider the requirement that altitude should be within 121 m at all times and we explicitly consider all infrastructure in the area as no-fly zones.

The UAS speed limit in the third row of Table 1 is represented in STL as:

$$\varphi_2 = \square_{[0,T]}\big(\|v\|_2 \leqslant 44\big), \tag{3}$$

Here, $v \in \mathbb{R}^3$ is the UAS velocity, and its 2-norm is the speed of the UAS.

The no-fly zones for avoiding airspace above pedestrians as well as flying under covered structures can be represented in STL using the convex hull of these structures:

$$\varphi_3 = \wedge_i\big(\square_{[0,T]} \neg\big(p \in CG_i\big)\big). \tag{4}$$

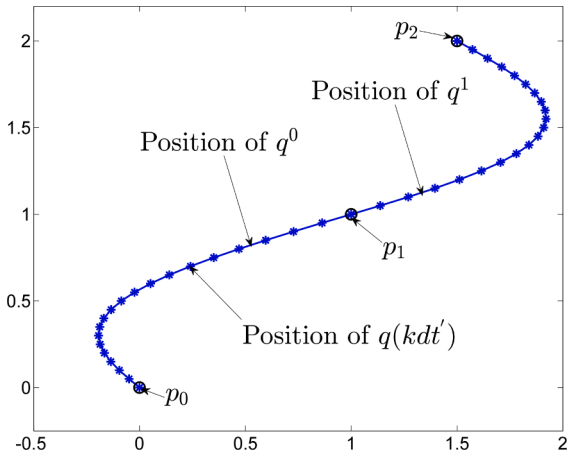

Fig. 5. Planar splines connecting position waypoints p_0, p_1 and p_2. q^0 is the continuous spline (positions, velocities and accelerations) connecting p_0 and p_1 and q_1 is the spline from p_1 to p_2. $q(kdt')$ is the k^{th} sample of q^0, with sampled time dt'.

Switching to a contingency plan in case of an emergency. Let φ_e be the specification corresponding to a contingency. An example is $\varphi_e = \Diamond_{[0,10]}\left(p^z \leqslant 1 \wedge \|\dot{p}\|_2 \leqslant 0.05\right)$, i.e. this contingency response states that within the next 10 s, the UAS altitude must be less than 1 m and its speed must be less than 0.05 ms^{-1}. Let φ_r be the mission specification during regular flight. The last row of Table 1 requires the UAS to ignore its mission and switch to a contingency plan in case of an emergency. This situation can be encoded in STL as follows:

$$\varphi_{contingency} = \left(E \Rightarrow \varphi_e\right) \vee \left(\neg E \wedge \varphi_r\right) \tag{5}$$

Inter UAS collision avoidance. (last row of Table 1), for all UAS d, indexed by the set $\mathscr{D} = \{1, \ldots c, D\}$, can be formulated as follows:

$$\varphi_{\text{Inter UAS}} = \wedge_{d,d' \in \mathscr{D}, d \neq d'} \square_{[0,T]}\left(\left\|p_d - p_{d'}\right\| \geqslant \text{minimum separation}\right) \tag{6}$$

Here, $p_d \in \mathbb{R}^3$ represents the position of UAS d. The above specification requires that in a pairwise manner, all UAS are at least the minimum separation distance away from each other. We note that this minimum separation can be different for each pair of UAS based on their sizes, vehicle class, or other operational factors. While our trajectory generation method described in Section 5 can take different separation minima into account, for ease of notation, we assume that the minimum separation distance between all pairs of UAS is the same.

In the succeeding sections, we present two UAS case studies where we combine the notion of control using the robust semantics of STL (Sections 3.2 and 3.3) along with equivalent mappings from high-level constraints and specifications detailed in this section. The first case study (Section 6) details an infrastructure surveillance mission carried out by a swarm of UAS within a rural environment; the second case study (Section 7) contends with heterogeneous missions carried out within a constrained, urban environment collocated with a major airport. We present the setup as well as discuss the results of both case studies in their respective sections.

5. Fly-by-Logic: Trajectory generation for STL satisfaction

Given an STL specification φ for the mission requirements, we use Algorithm 1 (Pant et al., 2018) to generate trajectories for the UAS involved in the mission such that they satisfy φ. The algorithm relies on solving an optimization (7) to maximize, over these UAS trajectories, the smooth robustness associated with the STL specification. These trajectories, which consist of N jerk-minimizing spline (Mueller et al., 2015) segments $q^0, \ldots c, q^N$, are parameterized by a set of $N+1$ waypoints (positions in $\mathbb{R}^3$) for each of the d UAS. These waypoints are to be reached sequentially every dt seconds and they form variables for the underlying optimization. This formulation results in trajectories with a total flight time of $T = Ndt$ seconds. Fig. 5 shows these waypoints, the continuous time spline trajectories q connecting them, and a high-rate ($dt' \ll dt$) discretization $q(kdt')$ of these trajectories for a single UAS. The detailed formulation is presented in Pant et al. (2018) and reproduced in the appendix (Section A), the rest of this section gives an overview of the approach and the key results. See Fig. 2 for a pictorial representation of the mission planning workflow.

Algorithm 1. Fly-by-Logic: Planning for multi-rotor UAS with STL specifications

(*continued on next page*)

(*continued*)

Data: Specification φ, parameters for optimization $\mathbf{P}_\varphi$ (7) and initial guess for waypoints $\bar{\mathbf{p}}^d$ of each UAS d
Result: For each of d UAS, a sequence of $N+1$ waypoints for jerk-minimizing spline generation $\mathbf{p}^1, \ldots, \mathbf{p}^d$
while $0 \leq k \leq N$ **do**
  At time $t = kdt$:
  Get current position: $p_k^d, \forall d = 1, \ldots, D$
  Set $\bar{\mathbf{p}}^d_{[0:k]} = \mathbf{p}^d_{[0:k]} \forall d$
  Solve $\mathbf{P}_\varphi(\bar{\mathbf{p}}^1, \ldots, \bar{\mathbf{p}}^D)$
  Update $[\mathbf{p}^0, \ldots, \mathbf{p}^D]$
  Give UAS d the next waypoint to track $p_{k+1}^d, \forall d$
  Update $\bar{\mathbf{p}}^d = \mathbf{p}^d$
  $k \leftarrow k+1$
end

5.1. Waypoint selection for Smooth Robustness maximization

Let $\mathbf{w} = [\mathbf{p}^0, \ldots c, \mathbf{p}^D] \in \mathbb{R}^{3(N+1)D}$ be the sequence of waypoints (at rate $1/dt$) over all D UAS in the mission and let $\mathbf{q}$ be the associated trajectories discretized at the higher rate $1/dt'$[8]. From the formulation of the jerk-minimizing splines (Mueller et al., 2015) it was shown in Pant et al. (2018) that we can compute a linear map $L : \Re^{(T/dt)} \rightarrow \Re^{(T/dt')}$ such that $\mathbf{q} = L(\mathbf{x})$. See appendix (Section A.1) for details. Also $v_k^d \in \mathbb{R}^3$ are the velocities of UAS d at time kdt. These are functions of the initial velocity v_0^d and the waypoints $p_0^d, \ldots c, p_k^d$ that the UAS has flown through up to time kdt. See the appendix for more details. Finally, the kinematic constraints on the UAS require that its velocity (for every axis of motion) is within the interval $\left[\underline{v}, \overline{v}\right]$ and acceleration in $\left[\underline{a}, \overline{a}\right]$. The robustness maximization problem[9] is finally:

$$\mathbf{P}_\varphi\left(\bar{\mathbf{p}}^1, \ldots c, \bar{\mathbf{p}}^D\right) : \max_{\mathbf{w}} \quad \widetilde{\rho}_{\varphi_s}\left(L\left(\mathbf{w}\right)\right) \tag{7a}$$

$$\text{s.t. } \forall k = 1, \ldots c, N, \ \forall d = 1, \ldots c, D \tag{7b}$$

$$\mathrm{LB}_v\left(v_{k-1}^d\right) \leqslant p_k^d - p_{k-1}^d \leqslant \mathrm{UB}_v\left(v_{k-1}^d\right), \tag{7c}$$

$$\mathrm{LB}_a\left(v_{k-1}^d\right) \leqslant p_k^d - p_{k-1}^d \leqslant \mathrm{UB}_a\left(v_{k-1}^d\right), \tag{7d}$$

$$\widetilde{\rho}_{\varphi_s}(L(\mathbf{w})) \geqslant \widetilde{\epsilon} \tag{7e}$$

Here, $(\bar{\mathbf{p}}^1, \ldots c, \bar{\mathbf{p}}^D)$ are initial guesses for the waypoints, with the first point for each UAS as their starting positions. Constraints (7c) and (7d) ensure continuous time kinematic feasibility. $\mathrm{LB}_v, \mathrm{UB}_v, \mathrm{LB}_a, \mathrm{UB}_a$ are linear functions of the waypoints that result in the trajectories between them to be kinematically feasible (see Theorem 8.1 in the appendix for details). These kinematic constraints accounting for velocity and acceleration bounds can be different for each UAS. For the sake of simplicity in the simulation case studies, and for ease of notation, we assume in the rest of this paper that all UAS have identical kinematic bounds. In a real-world deployment scenario of our framework, individual stakeholders can specify different UAS vehicle classes, each with different kinematic constraints.

Constraint (7e) ensures that the smooth robustness exceeds a pre-computed lower $\widetilde{\epsilon}$ bound such that the resulting trajectories satisfy the STL specification in continuous time (see Corollary 8.2.1 in the appendix). Note that we replace the mission specification φ with a *strictified* version φ_s (see Section A.3 for details). The main result regarding the properties of this optimization follows:

Theorem 5.1. (*Mission satisfaction and kinematic feasibility*) *A feasible solution to the optimization (7) generates trajectories for the UAS that:*

[8] In our case studies, we set $dt = 1$s and $dt' = 0.05$s
[9] Note that this is an optimization over variables of all the UAS in the STL specification, and is solved in a centralized manner.

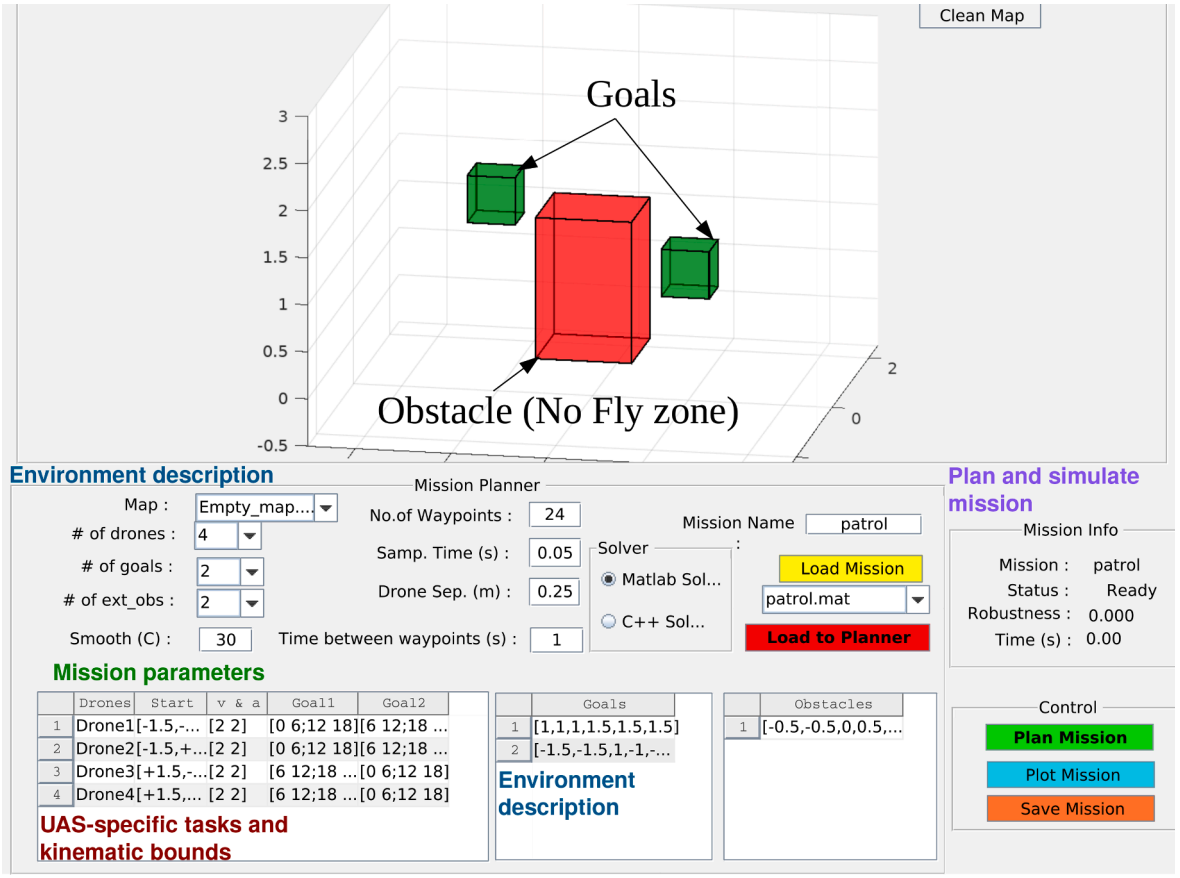

Fig. 6. Annotated overview of the Fly-by-Logic graphic user interface (GUI) for a 4-UAS patrolling mission example. Playback of the simulation, as well as experimental validation on an actual quad-rotor platform can be seen at https://youtu.be/xBQnEweVwZs.

1. *Satisfy the mission specification φ in continuous time.*
2. *Are kinematically feasible for the multi-rotor UAS.*

This theorem is a combination of Theorem 8.1 and Corollary 8.2.1 in the appendix which individually prove the two main results here. For clarity of presentation, the formal statement of these theorems and the proofs are presented in the appendix (Section A.2).

Example 4. In this example, four UAS are tasked with reaching a goal set within 6 s (encoded using the $\Diamond$ operator), while avoiding five static obstacles and one moving obstacle whose trajectory is known *a priori* (encoded using the $\Box$ operator). We denote the position of the moving obstacle by p_{moving}. The four UAS should also always maintain a safe distance δ from each other. In addition to these requirements, one of the UAS (position denoted by p_4) is not allowed into the goal set until the three other UAS have reached it (encoded using the $\mathcal{U}$ operator). This scenario is captured in the following specification:

$$\begin{aligned}\varphi \quad &= \wedge_{d=1}^{4}\wedge_{i=1}^{5}\Box_{[0,6]}\neg\left(p_d \in \text{Unsafe}_i\right)\\ &\wedge_{d,d',d'\neq d}\Box_{[0,6]}\|p_d - p_{d'}\| \geqslant \delta\\ &\wedge\neg\left(p_4 \in \text{Goal}\right)\mathcal{U}_{[0,6]}\left(\wedge_{d=1}^{3}\Diamond_{[0,6]}\left(p_d \in \text{Goal}\right)\right)\\ &\wedge\Diamond_{[0,6]}\left(p_4 \in \text{Goal}\right)\wedge_{d=1}^{4}\Box_{[0,6]}\|p_d - p_{\text{moving}}\| \geqslant \delta\end{aligned} \tag{8}$$

A video of trajectories generated by our method to satisfy (8) can be found at https://youtu.be/cmX7g6QN09c.[10]

5.2. Graphical user interface for mission specification

The task of writing the STL specification for a given mission requires a level of familiarity with the syntax and grammar of STL,

[10] While for this example, we use Algorithm 1 as described, for the case studies in this paper, we use it as a one-shot algorithm where the optimization of (7) is solved offline once to generate trajectories.

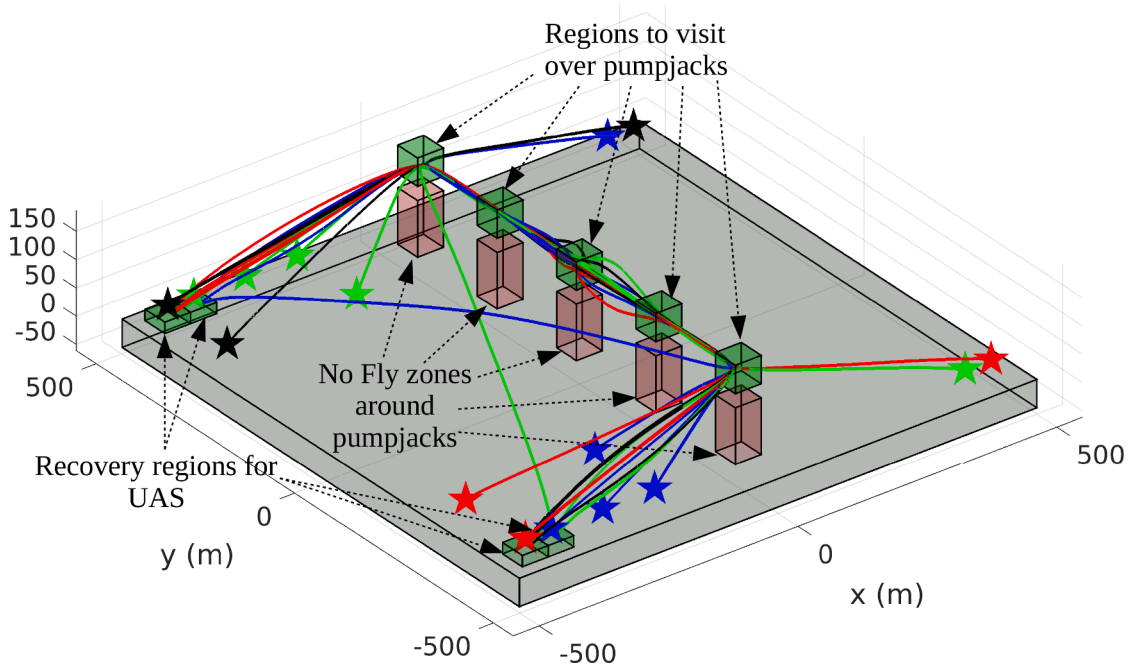

Fig. 7. Trajectories for four UAS tasked with flying over the pumpjacks by reaching all green-colored goal sets within 5 min, avoiding all black-colored obstacles, and landing in the green-colored sets at ground level, i.e., top of the gray box. The figure shows six sets of trajectories for each UAS, generated for six different sets UAS starting positions. The different colored ☆ symbols denote the various starting positions for each UAS. Note that within the set of six different runs, some UAS have the same starting points. This rural case study is outlined in Section 6. Video of the simulation can be found at: https://youtu.be/Xf-7msRzItk.

which could result in a barrier towards the widespread use of the FADS framework presented in this paper. In order to mitigate this, we have developed a graphical interface (Fig. 6) that allows UAS fleet operators to visually specify and plan upcoming missions. These missions correspond to a fragment of STL (Pant et al., 2019), thus allowing our FADS framework to be applied without requiring the end user to be technically familiar with temporal logic and the underlying trajectory planning process. The publicly available version of FADS and associated GUI is available at https://github.com/yashpant/FlyByLogic.

5.3. Solving the trajectory planning optimization and simulation setup

The waypoint selection optimization for trajectory planning (7) was formulated using Casadi (Andersson, 2013) in C++, with Ipopt (Wächter and Biegler, 2020) as the optimization solver. HSL routines (Hsl, 2021) were used as internal linear solvers in Ipopt. The simulation case studies that follow were run on a laptop with a quad-core i7-7600 processor (2.8 Ghz) and 16 Gb RAM running Ubuntu 17.04. For the simulation studies, the waypoints are separated by $T = 10$s, and the higher-rate discretization of the trajectories is done with $dt' = 0.1$s. Since the optimization (7) is over a non-convex objective, we can *multi-start* the optimization procedure by providing different initial trajectories as starting points for the optimization (Pant et al., 2017). The resulting optimizations can then be solved in parallel, and the resultant best solution can be flown by the UAS. For a given application, these initial trajectories can be crafted to incorporate domain-specific knowledge, e.g., rerouting certain UAS onto longer paths to avoid congestion in particular regions of the airspace.

6. Case study: Single mission operations in rural airspace

Consider a low-altitude mission profile (adapted from (Federal Aviation Administration, 2015)) for UAS within a sparsely populated area. Specifically, an operator will deploy UAS in a beyond-line-of-sight setting to survey pumpjacks along an active oil pipeline. The properties and characteristics of this mission profile could serve as a template for other rural use cases for UAS, such as end-point package deliveries, wildfire management and other infrastructure surveillance use cases (Skydio, 2020). Fig. 7 depicts the airspace setting for our low-altitude UAS use case. Note that the allocated location and blocked-off altitudes for this particular mission profile indicate that we do not need to consider interference with commercial aviation. In this use case, we deploy four small multi-rotor UAS tasked with surveying five pumpjacks located along an oil pipeline within a given time frame. In order to carry out a successful surveillance mission, the team of UAS must fly directly over each pumpjack and collect information regarding the pumpjack, *e.g.* taking high-definition aerial photographs.

For safety reasons and in compliance with FAA regulations regarding UAS operations, the mission profile also specifies the

Table 2
Summary of simulation results. Our approach (FADS) generates trajectories that satisfy the requirements of both case studies robustly. The state-of-the-art MILP-based approach BluSTL (Raman et al., 2014) does not return a solution within 2 h for either case study, which we refer to as a timeout.

Mission	Planning time-steps (at 10 Hz)	Robustness (m) [FADS]	Computation time(s) [FADS]	Robustness [BluSTL]	Computation time(s) [BluSTL]
Rural	3000	4.1	139	N/A	Timeout
Urban	6600	5.6	1440	N/A	Timeout

enforcement of pairwise separation requirements. Each UAS must maintain a separation of at least 5 meters from another UAS for the entirety of the mission time interval. Another important operating constraint enforced by the mission profile is that the UAS must stay within the predefined airspace at all times, as well as respect maximum allowable velocities and accelerations. Both physical constraints can be modified *a priori* as needed by the operator. Finally, the mission is considered to be completed once all UAS reaches a predefined landing area, where they will then be recovered by the operator.

6.1. STL formalization of the mission profile

In order to capture this rural case study using STL specifications, we must first define three-dimensional sets that demarcate the various physical attributes of our mission environment. Recall that each UAS must perform a fly-over for each pumpjack; let $Pjack_i$ denote the airspace region directly above each pumpjack $i \in \{1,\ldots,5\}$. Within the STL specifications, each UAS will be required to visit each $Pjack_i$ set within the time interval of the mission.

While $Pjack_i$ denotes airspace regions that UAS will be required to fly to, there may also be predefined regions of the airspace that must *not* be visited by UAS. An example could be telecommunication infrastructures in rural areas that UAS must stay clear of. We will denote such no-fly zones within the mission environment as *NoFly*. For our case study specifically, the no-fly zones include the physical infrastructures for each pumpjack, as well as a safety buffer region around them. Finally, for each of the four UAS $d \in \{1,2,3,4\}$, we specify a *recovery* region within the mission environment, denoted by $Recovery_d$. The sets $Pjack_i$, *NoFly*, and $Recovery_d$ are depicted in Fig. 7.

In addition to the spatial components that we need to specify, we also need to analogously specify the temporal components of the mission profile. More precisely, we will define the main time interval of the mission, as well as sub-intervals during which relevant events must occur. Let $I = [0, T]$ be the main time interval of the mission, during which all five pumpjacks must be surveyed by each of the four UAS. Note that T is the maximum allowable flight time allocated to the mission, specified in our case study in seconds. We also specify a sub-interval of time $I_i^d \subseteqq I$ wherein all four UAS $d \in \{1,2,3,4\}$ must perform a fly-over and survey pumpjack $i \in \{1,\ldots,5\}$. Complimentary to the recovery sets $Recovery_d$, we specify time sub-intervals $I_d \subseteqq I$ wherein each UAS must enter their recovery sets, signifying the end of their mission tasks.

Now that we have defined the spatial and temporal components of the mission environment, we move on to finalize the overall mission specifications by defining the minimum pairwise separation distance as $d_{\min} = 5$ meters. Let the STL specification formula for the mission assigned to each UAS d be denoted by φ_d; we have that the mission for each UAS d is formalized in STL as

$$\begin{aligned}\varphi_d = \; & \wedge_{i=1}^{5}\left(\Diamond_{I_i^d}(p_d \in Pjack_i)\right) \wedge \Box_I(p_d \notin NoFly) \\ & \wedge \Diamond_{I_d}(p_d \in Recovery_d)\end{aligned} \tag{9}$$

This STL formula can be parsed as follows: Each UAS d must visit and fly-over the five pumpjacks ($p_d \in Pjack_i$) within the time sub-intervals allocated ($\Diamond_{I_i^d}$). While completing their flyover, all UAS must stay away from no-fly zones ($p_d \notin NoFly$), and this is enforced throughout the entirety of the main mission time interval ($\Box_I$). Finally, each UAS d must eventually navigate to their recovery sets ($p_d \in Recovery_d$); this must be done within the specified time sub-interval for reaching the recovery set as well ($\Diamond_{I_d}$).

Let $\varphi_{\text{pipeline}}$ be the *overall mission specification* across all UAS, defined in terms of φ_d as well as the required pairwise separation constraints. We can write $\varphi_{\text{pipeline}}$ explicitly as

$$\varphi_{\text{pipeline}} = \left(\wedge_{d=1}^{4}\varphi_d\right) \wedge \left(\wedge_{d,d',d'\neq d}\Box_I \|p_d - p_{d'}\| \geqslant d_{\min}\right) \tag{10}$$

The overall mission specification $\varphi_{\text{pipeline}}$ states that in addition to carrying out the pipeline pumpjack surveillance mission, each unique pair $(d, d') \in \{1,2,3,4\}^2$ of UAS should be separated by at least $d_{\min}$. Our STL specification for this particular mission profile is complete. Finally, we note that the requirements of staying within the predefined mission environment, as well as the predefined bounds on velocity and acceleration, are linear constraints imposed on the state of the UAS. Thus, we directly incorporate these kinematic constraints within the Fly-by-Logic planning algorithm in (7).

6.2. Rural case study results and discussion

We evaluated our approach on this case study for six different sets of initial positions for the four UAS. Table 2 shows the average computation time for the optimization and the associated robustness value. Our approach managed to find trajectories for the four UAS

which satisfy the mission for all six cases with an average robustness value of 4.1 meters. This robustness value shows that the UAS have large buffers within which deviations from their planned trajectories could still satisfy mission requirements. Furthermore, we can solve this problem in about 2 min, demonstrating its applicability. We also compare our approach to BluSTL (Raman et al., 2014), a Mixed Integer Linear Programming (MILP)-based method for control with STL specifications. BluSTL cannot return a solution within 2 h for this problem, possibly because the MILP encoding of the specification $\varphi_{\text{pipeline}}$ results is intractable. Such a result was also noted in Pant et al. (2017), albeit for a different multi-UAS mission specification.

7. Case study: Multi-mission operations in urban airspace

We now formalize the case study containing multiple heterogeneous missions in an urban airspace, which we outlined previously in Example 1. This case study highlights the methodological contributions and implementation improvements over (Pant et al., 2018), allowing us to work in practice with specifications involving nested temporal logic operators as well as long time horizons (of the order of minutes) and large workspaces. The positive results from this case study showcase the applicability of our FADS implementation – UAS trajectories could be successfully and robustly generated, with adherence to high-level constraints and specifications detailed in Table 1.

7.1. Case study setting and mission description

The setting of this case study is in the immediate vicinity of a major airport, specifically Philadelphia International Airport (PHL). We selected PHL to take advantage of three factors around which we could build an UAM case study: (1) the large industrial and business complexes to the west of PHL, (2) the relatively large air cargo operations at PHL operated by multiple cargo carriers, including UPS and DHL, and (3) the presence of multiple parking facilities for airline passengers spread out around PHL. Our case study consists of five UAS conducting different missions around the three aforementioned facilities around PHL, within an 11-min time horizon.

Fig. 1 shows a top-down projection of the case study workspace. We note that the trajectory planning and optimization is performed over the entire three-dimensional space; the two-dimensional top-down view is shown for visual brevity. The areas – predefined sets in $\mathbb{R}^3$ – marked in green demarcate goal sets for the various UAS missions in this case study. These goal sets can be found around the parking areas at PHL (e.g. WallyPark Airport Parking), the air cargo ramps in the PHL airside facilities (e.g. DHL's ramp area), a designated drop-off location at the airport terminal for airline passengers, and external drop-off points for packages and passengers (e.g. Stephenson equipment; Airport Business Complex).

Similarly, the unsafe set, or no-fly zones, are marked in red in Fig. 1. These include the three primary arrival and departure runways at PHL (i.e. 9L/27R, 9R/27L, and 17/35), as well as two land-side areas of PHL with heavy pedestrian or surface traffic. In our case study implementation, the no-fly zones are extended beyond the flight ceiling of the UAS (set at 100 m); this is to ensure that UAS do not attempt to fly over the no-fly zones, as such trajectories are infeasible. Note that the no-fly zones can be easily modified to adhere to various stakeholder needs; for example, it is likely that the runway no-fly zones will need to be expanded to include runway imaginary surfaces that account for the glide-path of arriving aircraft generated by the Instrument Landing System (De Neufville et al., 2013). These can all be easily incorporated and flexibly modified via the FADS user interface shown in Fig. 6.

7.2. STL specifications

With regards to the no-fly zones, the safety specification for all five UAS operating within our case study airspace is:

$$\varphi_{NoFly} = \wedge_{i=1}^{5} \square_{[0,660]} \neg \left(p \in NoFly_i\right) \tag{11}$$

The five UAS will be tasked with carrying out three different types of missions within our case study PHL airspace. Two UAS will be assigned air shuttle operations, operating autonomously on fixed routes. Two additional UAS will be assigned last-mile package delivery missions. The last UAS will be assigned an air taxi mission that operates in an on-demand manner between two passenger drop-off and pick-up locations.

Autonomous air shuttle: The first UAS assigned as an autonomous air shuttle will be shuttling airline passengers between the airport terminals, WallyPark Airport Parking, and PreFlight Airport Parking. The UAS begins at WallyPark Airport Parking, and flies to the drop-off and pick-up location at the PHL terminals. The UAS stays at the airport terminals for 30 s to offload and take on new passengers, before departing for the PreFlight parking area. After another 30-s interval for pick-up and drop-off, the UAS flies back to the airport terminal followed by WallyPark Airport Parking. The STL specification for this particular UAS, together with the no-fly zone requirements, is captured as follows:

$$\begin{aligned} \varphi_{\text{shuttle}-1} = \; & \Diamond_{[0,100]} \square_{[0,30]} \left(p_1 \in \text{Terminal}\right) \wedge \varphi_{NoFly} \wedge \\ & \Diamond_{[130,300]} \square_{[0,30]} \left(p_1 \in \text{PreFlight}\right) \wedge \\ & \Diamond_{[330,500]} \square_{[0,30]} \left(p_1 \in \text{Terminal}\right) \wedge \\ & \Diamond_{[530,630]} \square_{[0,30]} \left(p_1 \in \text{WallyPark}\right) \end{aligned} \tag{12}$$

The second UAS assigned as an air shuttle runs a similar route, but only between Colonial airport parking and the airport terminal. The

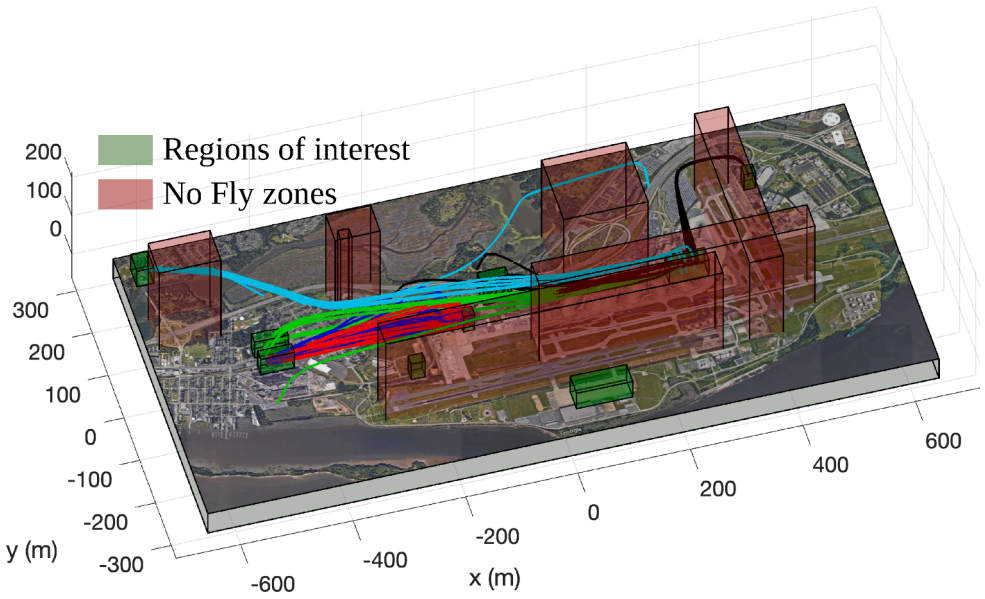

Fig. 8. Mission workspace for the urban airspace case study, and six sets of trajectories for the five UAS corresponding to the various missions being executed by these UAS from six different initial positions. Regions in red represent no fly zones, while those in green represent regions of interest that the UAS will visit during their missions. Also see Fig. 1 for a detailed labeling of this airspace. A video of this case study, with a 10× speed-up, can be found here: https://youtu.be/bC6dleCGW8A.

behavior of this UAS at each of its two stops is analogous to the first UAS: 30 s are allotted for passenger drop-off and pick-up at both Colonial airport parking as well as the airport terminal. Furthermore, we specify that this shuttle route must be run twice during the 11-min time horizon of this example, translating to a particular set of UAS operational velocities. For this air shuttle UAS, the STL specifications is captured as follows:

$$\begin{aligned}\varphi_{\text{shuttle}-2} = \; & \Diamond_{[0,135]}\Box_{[0,30]}\left(p_2 \in \text{Terminal}\right) \wedge \varphi_{NoFly} \wedge \\ & \Diamond_{[165,300]}\Box_{[0,30]}\left(p_2 \in \text{Colonial airport}\right) \wedge \\ & \Diamond_{[330,465]}\Box_{[0,30]}\left(p_2 \in \text{Terminal}\right) \wedge \\ & \Diamond_{[495,630]}\Box_{[0,30]}\left(p_2 \in \text{Colonial airport}\right)\end{aligned} \tag{13}$$

Autonomous last-mile package delivery: Two UAS are tasked with carrying packages from the DHL cargo ramp at PHL to the Airport Business Complex, then returning back to the DHL cargo ramp. The two UAS must wait for 50 s at both the DHL cargo ramp as well as the Airport Business Complex in order to unload and reload packages. Since these two UAS are operating in tandem and executing the same mission, the STL specifications for both UAS are identical, and can be written as follows:

$$\begin{aligned}\varphi_{deliver-1} = \; & \Diamond_{[0,50]}\Box_{[0,50]}\left(p_3 \in \text{Business}\right) \wedge \varphi_{NoFly} \wedge \\ & \Diamond_{[100,150]}\Box_{[0,50]}\left(p_3 \in \text{DHL}\right) \wedge \\ & \Diamond_{[200,250]}\Box_{[0,50]}\left(p_3 \in \text{Business}\right) \wedge \\ & \Diamond_{[300,350]}\Box_{[0,50]}\left(p_3 \in \text{DHL}\right) \wedge \\ & \Diamond_{[400,450]}\Box_{[0,50]}\left(p_3 \in \text{Business}\right) \wedge \\ & \Diamond_{[500,550]}\Box_{[0,50]}\left(p_3 \in \text{DHL}\right) \wedge \\ & \Diamond_{[600,650]}\Box_{[0,10]}\left(p_3 \in \text{Business}\right)\end{aligned} \tag{14}$$

Autonomous air taxi: The final UAS is assigned as an on-demand air taxi that operates between two requested destinations. In our case study, we enact a particular requested trip where the passenger requests to be picked up at the airport terminal, and dropped off at the location of Stephenson equipment, located approximately 4 miles (6,500 m) northwest of PHL. We specify that this air taxi trip must be carried out within the 11-min time horizon; thus, the STL specification for this UAS is captured as follows:

$$\varphi_{\text{air}-\text{taxi}} = \Diamond_{[0,660]}\left(p_5 \in \text{Stephenson}\right) \wedge \varphi_{NoFly} \tag{15}$$

Finally, using the method described in Section 5, the trajectory planning is done for all five UAS in a centralized manner, along with a pairwise separation requirement of at least 5 m between UAS. The overall mission specification can be written as follows:

$$\begin{aligned}\varphi_{\text{PHL}} = \; & \wedge_{d=1}^{2}\varphi_{\text{shuttle}-\text{d}} \wedge \wedge_{d=3}^{4}\varphi_{\text{deliver}-\text{d}} \wedge \varphi_{\text{air}-\text{texi}} \wedge \\ & \wedge_{d,d',d\neq d'}\Box_{[0,660]}\left(\|p_d - p_{d'}\| \geqslant 5\right)\end{aligned} \tag{16}$$

7.3. Urban case study results and discussion

The Fly-by-Logic algorithm embedded as a part of our FADS implementation was able to generate feasible trajectories for all five UAS such that the STL specification of φ_{PHL} was satisfied. We generated six sets of trajectories, corresponding to six instances of the mission with the UAS starting from different initial positions. Fig. 8 shows these trajectories overlaid on the mission workspace. The trajectory generation process is done offline prior to the UAS executing their missions, and this process took on average 24 min to plan trajectories for five UAS with a realistically long mission time horizon of 11 min[11]. Again, as in the previous case study, the MILP-based approach times out on this problem. In addition to the long mission horizon, one of the primary reasons for the high trajectory computation time here (see Table 2) is the $\binom{5}{2}$ inter-UAS separation specifications in (16). However, even in large airspace with comparatively few UAS (e.g., the setting of this urban airspace case study), it is critical to take inter-UAS separation into account: Simulations for UAS trajectory generation where the minimum separation requirement $\varphi_{\text{InterUAS}} = \wedge_{d,d' \in \mathscr{D}, d \neq d'} \square_{[0,660]}(\|p_d - p_{d'}\| \geqslant 5)$ in (16) is ignored resulted in at-risk trajectories where UAS pairs were within 5 meters of each other for an extended duration (see appendix A.4). While ignoring this requirement could result in a reduction of computation times (869s as opposed to 1440s), with even greater reductions possible when the mission specification allows for UAS to plan independently (Rodionova et al., 2021), the resulting trade-off in safety is unacceptable for many safety-critical applications, e.g., operations proximate to a major international airport, as depicted in this urban airspace case study.

Recall the robustness value from Section 3.2; the results from this case study produced an average robustness value of 5.6 meters. In particular, for the mission specification (16), since there is a minimum separation of $5m$ between two UAS, the planned trajectories result in no two UAS being closer than $5 + 5.6 = 10.6m$. For online execution, our approach could be combined with a Detect-and-Avoid (DAA) system (e.g., (Lin and Saripalli, 2017; Rodionova et al., 2021)) to ensure safety, even under unforeseen circumstances. Furthermore, we showed that we could handle nested temporal logic operators within our STL specifications (e.g., (12), (13)), allowing for more nuanced and practically useful UAS behaviors such as hovering in a region for a given time, or repeatedly visiting given regions. We also emphasize that through the FADS GUI (Fig. 6), UAS operators could modify not only the number and characteristics of missions, but also the quantity and dimensions of no-fly zones and goal sets.

In a realistic deployment setting, the UAS operator would execute the FADS framework to generate a complete set of missions that span some time horizon. This set of missions could then be carried out repetitively (e.g. for our urban airspace case study, multiple 11-min segments would be looped continuously), or stitched with other sets of missions for a variety of operating scenarios. This urban airspace case study demonstrates the utility of our FADS implementation to applications outside of a simplistic rural airspace setting with a single mission type. We also include a video of five UAS carrying out this case study in simulation (https://youtu.be/bC6dleCGW8A). Note that although the simulation can be visualized in real time, we impose a 10× playback speed to offset the 11-min mission time horizon.

8. Concluding remarks

Summary: In this paper, we present a framework, FADS, to enable safe operation of UAS fleets tasked with missions that have spatial, temporal, and reactive objectives. We use Signal Temporal Logic (STL) to capture mission requirements as well as FAA safety regulations applicable to the UAS fleet, and develop an optimization-based approach to generate trajectories for the UAS that ensure safe and compliant operations. We also report and attempt to maximize an easy-to-use robustness metric, which directly translates to deviation-tolerant trajectories, and can be used for benchmarking purposes as well.

We demonstrate the applicability of our approach through case studies that involve multiple UAS operating in rural and urban airspace with varying degrees of complexities. Our simulation results show that FADS can successfully generate UAS trajectories for both types of case studies. Furthermore, the high robustness values (on the order of meters) for these trajectories show that bounded deviations from the planned trajectories do not affect the outcome, i.e., satisfaction of mission specifications. In addition, we also highlight some novel technical contributions achieved in the design and completion of the urban airspace case study. Specifically, these technical contributions include the ability to handle long time horizons and to encode specifications with nested STL operators that enable more realistic spatio-temporal vehicle behaviors, both of which are significant improvements upon the state-of-the-art.

Limitations and future work: While FADS improves upon the state-of-the-art in terms of planning for UAS fleets with formal requirements, it still suffers from high computation times as missions become more complicated. For instances, trajectory generation for the urban airspace case study takes ten times as long compared to the rural airspace setting. In such cases, our trajectory planning approach can only be run offline, and not in an online feedback-based manner. Additionally, we could use a multi-start approach for the trajectory planning optimization (see Section 5.3). This allows for the incorporation of domain-specific information in the initial guesses for trajectory solutions, and can potentially lower the computational burden on the optimization solver. Also, instead of jointly optimizing for all UAS in a centralized manner as we do here, future work will aim to develop decentralized approaches that can further improve scalability. While the trajectory generation approach has been tested on real-world robots (Pant et al., 2018), we also

[11] This is the time for the optimization to finish maximizing the robustness of the trajectories. It takes a considerably shorter time to find trajectories that satisfy the mission, i.e. trajectories with a small positive robustness value. This trend was also observed in general in Pant et al. (2017, 2018, 2014).

aim to test the FADS framework in real-world settings and extend it for cases where the full workspace is not known *a priori*. The results and lessons learned from these evaluations could help inform, for example, future zoning policies that may need to consider the impact of UAM on urban residents.

Acknowledgments

This work was supported by STARnet a Semiconductor Research Corporation program sponsored by MARCO and DARPA, NSF MRI-0923518 and the US Department of Transportation University Transportation Center Program.

Appendix A

A.1. Generating trajectories

The mapping L between low-rate $\mathbf{x}[dt]$ and high-rate $\mathbf{y}[dt']$ is implemented by the following trajectory generator, adapted from (Mueller et al., 2015). It takes in a motion duration $T_f > 0$ and a pair of position, velocity and acceleration tuples, called *waypoints*: an *initial* waypoint $q_0 = (p_0, v_0, a_0)$ and a *final* waypoint $q_f = (p_f, v_f, a_f)$. It produces a continuous-time minimum-jerk (time derivative of acceleration) trajectory $q(t) = (p(t), v(t), a(t))$ of duration T_f s.t. $q(0) = q_0$ and $q(T_f) = q_f$. In our control architecture, the waypoints are the elements of the low-rate $\mathbf{x}$ computed by solving (7). The generator of Mueller et al. (2015) formulates the dynamics of a quadrotor UAS in terms of 3D jerk and this allows a decoupling of the equations along three orthogonal jerk axes. By solving three independent optimal control problems, one along each axis, it obtains three minimum-jerk trajectories, each being a spline $q^* : [0, T_f] \to \Re^3$ of the form:

$$\begin{bmatrix} p^*(t) \\ v^*(t) \\ a^*(t) \end{bmatrix} = \begin{bmatrix} \frac{\alpha}{120}t^5 + \frac{\beta}{24}t^4 + \frac{\gamma}{6}t^3 + a_0t^2 + v_0t + p_0 \\ \frac{\alpha}{24}t^4 + \frac{\beta}{6}t^3 + \frac{\gamma}{2}t^2 + a_0t + v_0 \\ \frac{\alpha}{6}t^3 + \frac{\beta}{2}t^2 + \gamma t + a_0 \end{bmatrix} \tag{17}$$

Here, α, β, and γ are scalar *linear* functions of the initial q_0 and final q_f.

We consider the case when the desired initial and endpoint velocities, v_0 and v_f, are free. We also assume $a_0 = a_f = 0$, that is the trajectories get to the end points with constant velocity. The constants in the spline (17) are then (Mueller et al., 2015):

$$\begin{bmatrix} \alpha \\ \beta \\ \gamma \end{bmatrix} = \frac{1}{2T_f^5} \begin{bmatrix} 90 & -15T_f^2 \\ -90T_f & 15T_f^3 \\ 30T_f^2 & -3T_f^4 \end{bmatrix} \begin{bmatrix} p_f - p_0 - v_0T_f \\ v_f - v_0 \end{bmatrix} \tag{18}$$

An example of such a spline (planar) is shown in Fig. 5.

A.2. Kinematic feasibility of the splines

The splines (17) that define the trajectories come from solving an unconstrained optimal control problem, so they are not guaranteed to respect any state and input constraints, and thus might not be *kinematically feasible*. By kinematically feasible, we mean that the quadrotor UAS can be actuated (by the motion controller) to follow the spline. Typically, feasibility requires that the spline velocity and acceleration be within certain bounds. E.g. a sharp turn is not possible at high speed, but can be done at low speed. Therefore, we formally define kinematic feasibility as follows.

Definition 8.1. (*Kinematically feasible trajectories*) Let $\left[\underline{v}, \overline{v}\right]$ be bounds on velocity and $\left[\underline{a}, \overline{a}\right]$ be bounds on acceleration. A trajectory $q : [0, T_f] \to \Re^3$, with $q(t) = (p(t), v(t), a(t))$, is *kinematically feasible* if $v\Big(t\Big) \in \left[\underline{v}, \overline{v}\right]$ and $a\Big(t\Big) \in \left[\underline{a}, \overline{a}\right]$ for all $t \in [0, T_f]$ for each of the three axes of motion.

Assumption 1. We assume that kinematically feasible spline trajectories, as defined here, can be tracked almost perfectly by the position (and attitude) controller. Design of such a controller with a bounded tracking error is presented in Pant et al. (2021) and is beyond the scope of this work.

Here, we derive constraints on the desired end state (p_f, v_f, a_f) such that the resulting trajectory $q(\cdot)$ computed by the generator (Mueller et al., 2015) is kinematically feasible. Since the trajectory generator works independently on each jerk axis, we derive constraints for a one-axis spline given by (17). An identical analysis applies to the splines of other axes. Since a quadrotor UAS can achieve the same velocities and accelerations in either direction along an axis, we take $\underline{v} < 0 < \bar{v} = -\underline{v}$ and $\underline{a} < 0 < \bar{a} = -\underline{a}$.

The spline trajectories generated here have what we call *free end velocities* as $\mathbf{a_f} = \mathbf{a_0} = \mathbf{0}$, and v_f is free to be within kinematic bounds. Without loss of generality $p_0 = 0$. Substituting (18) in (17) and re-arranging terms yields the following expression for the optimal translational state:

$$\begin{aligned} p_t^* &= \left(\frac{90t^5}{240T_f^5} - \frac{90t^4}{48T_f^4} + \frac{30t^3}{12T_f^3}\right)p_f - \left(\frac{90t^5}{240T_f^4} - \frac{90t^4}{48T_f^3} + \frac{30t^3}{12T_f^2} - t\right)v_0 \\ v_t^* &= \underbrace{\left(\frac{90t^4}{48T_f^5} - \frac{90t^3}{12T_f^4} + \frac{30t^2}{4T_f^3}\right)}_{K_3(t)}p_f - \left(\frac{90t^4}{48T_f^4} - \frac{90t^3}{12T_f^3} + \frac{30t^2}{4T_f^2} - 1\right)v_0 \\ a_t^* &= \underbrace{\left(\frac{90t^3}{12T_f^5} - \frac{90t^2}{4T_f^4} + \frac{30t^t}{2T_f^3}\right)}_{K_4(t)}p_f - \left(\frac{90t^3}{12T_f^4} - \frac{90t^2}{4T_f^3} + \frac{30t^t}{2T_f^2}\right)v_0 \end{aligned} \tag{19}$$

Applying the velocity and acceleration bounds $\underline{v} \leqslant v^* \leqslant \bar{v}$ and $\underline{a} \leqslant a^* \leqslant \bar{a}$ to (19) and re-arranging terms yields:

$$\frac{\left(\underline{v} - \left(1 - T_f K_3\left(t\right)\right)v_0\right)}{K_3(t)} \leqslant p_f \leqslant \frac{\left(\bar{v} - \left(1 - T_f K_3\left(t\right)\right)v_0\right)}{K_3(t)},\ \forall t \in \left[0, T_f\right] \tag{20a}$$

$$\underline{a}/K_4\left(t\right) + T_f v_0 \leqslant p_f \leqslant \bar{a}/K_4\left(t\right) + T_f v_0,\ \forall t \in \left[0, T_f\right] \tag{20b}$$

The constraints on p_f are linear in v_0, but parameterized by functions of t. Since t is continuous in $[0, T_f]$, (20) is an infinite system of linear inequalities. Fig. 9 shows these linear bounds for $t = 0, 0.1, 0.2, \ldots c, 1 = T_f$ with $\bar{v} = 1 = -\underline{v}, \bar{a} = 2 = -\underline{a}$.

The infinite system can be reduced to only 2 inequalities:

Lemma 1. *p_f satisfies (20) if it satisfies the following*

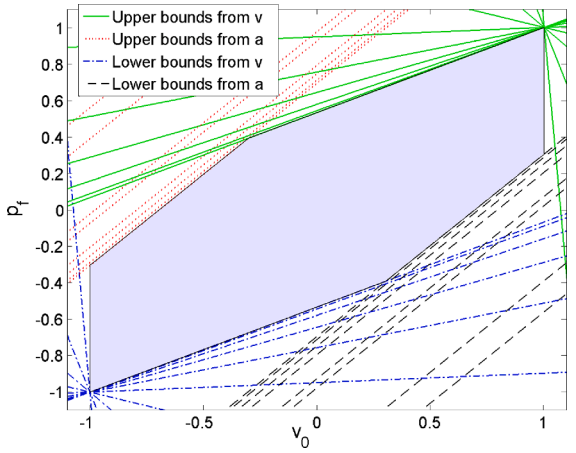

Fig. 9. The upper and lower bounds on p_f due to the acceleration and velocity constraints. Shown as a function of v_0 for $t = 0, 0.1, \ldots, T_f = 1$. The shaded region shows the feasible values of p_f as a function of v_0.

$$\begin{aligned} \frac{\underline{v}-\left(1-T_f K_3\left(T_f\right)\right)v_0}{K_3\left(T_f\right)} &\leqslant p_f \quad \leqslant \frac{\overline{v}-\left(1-T_f K_3\left(T_f\right)\right)v_0}{K_3\left(T_f\right)} \\ T_f v_0+\underline{a}/K_4\left(t'\right) &\leqslant p_f \quad \leqslant T_f v_0+\overline{a}/K_4\left(t'\right) \end{aligned} \tag{21}$$

where t' is a solution of the quadratic equation $\frac{dK_4(t)}{dt}=0$, such that $t' \in [0,T_f]$. *Proof of Lemma 1.* We first prove the upper bound of the first inequality, derived from velocity bounds. The lower bound follows similarly. First, note that the upper bounds $v_0 \mapsto (\overline{v}-(1-T_f K_3(t))v_0)/K_3(t)$ are lines that intersect at $v_0=\overline{v}$ for all t. Indeed, substituting $v_0=\overline{v}$ in the upper bound yields $\overline{v}-(1-T_f K_3(t))\overline{v})/K_3(t)=T_f\overline{v}$ regardless of t (see Fig. 9). Thus the least upper bound is the line with the smallest intercept with the y-axis. Setting $v_0=0$ in (20), the intercept is $\overline{v}/K_3(t)$. This is smallest when $K_3(t)$ is maximized. Since K_3 is monotonically increasing ($\frac{dK_3(t)}{dt}\geqslant 0$), $K_3(t)$ is largest at $t=T_f$. Thus the least upper bound on p_f is $(\overline{v}-(1-T_f K_3(T_f))v_0)/K_3(T_f)$.

We now prove the upper bound for the second inequality, derived from acceleration bounds. The lower bound follows similarly. The upper bounds $t \mapsto \overline{a}/K_4(t)+T_f v_0$ have the same slope, T (see Fig. 9). The least upper bound therefore has the smallest intercept with the y-axis, which is $\overline{a}/K_4(t)$. The smallest intercept, yielding the smallest upper bound, occurs at the t that maximizes K_4. Since $K_4(t)$ is concave in t in the interval $[0,T_f]$, it is maximized at the solution of $\frac{dK_4(t)}{dt}=0$. This establishes the result for $p_0=0$. Refer to Figs. 9 and 10 for the intuition behind this proof. For the general case, simply replace p_f by p_f-p_0 and apply the $p_0=0$ result. Through the decoupling of axes, this result holds for all three jerk axes. □

The main result, following from the above lemma, is as follows:

Theorem 8.1. (*Free endpoint velocity feasibility*) *Given an initial translational state $p_0, v_0 \in \left[\underline{v},\overline{v}\right]$, $a_0=0$, and a maneuver duration T_f, if p_f satisfies*

$$\begin{aligned} \frac{\underline{v}-\left(1-T_f K_3\left(T_f\right)\right)v_0}{K_3\left(T_f\right)} &\leqslant p_f-p_0 \leqslant \frac{\overline{v}-\left(1-T_f K_3\left(T_f\right)\right)v_0}{K_3\left(T_f\right)} \\ T_f v_0+\underline{a}/K_4\left(t'\right) &\leqslant p_f-p_0 \leqslant T_f v_0+\overline{a}/K_4\left(t'\right) \end{aligned} \tag{22}$$

with t' defined as in Lemma 1, then $v^*\left(t\right) \in \left[\underline{v},\overline{v}\right]$ and $a_t^* \in \left[\underline{a},\overline{a}\right]$ for all $t \in [0,T_f]$ and $p^*(T_f)=p_f$.

A.3. Strictification for continuous time guarantees

In general, if the *sampled* trajectory $\mathbf{q}$ satisfies φ, this does not guarantee that the continuous-time trajectory q also satisfies it. For that, we use (Fainekos, 2008, Thm. 5.3.1), which defines a *strictification operator* str: $\varphi \mapsto \varphi_s$ that computes a syntactical variant of φ having the following property:

Theorem 8.2. (*Fainekos, 2008*) *Let dt be the sampling period, and suppose that there exists a constant $\Delta_g \geqslant 0$ s.t. for all t, $\|q(t)-q(t+dt)\| \leqslant \Delta_g dt$. Then $\rho_{\varphi_s}(\mathbf{q})>\Delta_g \Rightarrow (q,0) \vDash \varphi$.*

Intuitively, the stricter φ_s tightens the temporal intervals and the predicates μ_k so it is "harder" to satisfy φ_s than φ. See (Fainekos, 2008, Ch. 5). For the trajectory generator g of Section A.1, Δ_g can be computed given $T_f, \underline{v}, \overline{v}, \underline{a}$ and $\overline{a}$.

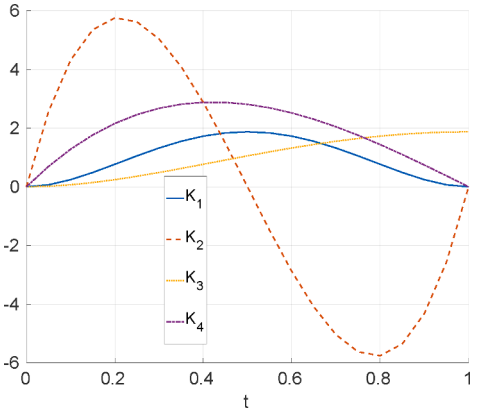

Fig. 10. The functions K_1 to K_4 for $T_f=1$.

We need the following easy-to-prove result, obtained by combining the theorem above and Theorem 3.2, to account for the fact that we optimize a smoothed robustness:

Corollary 8.2.1. *Let δ_{φ_s} be the worst-case approximation error for smooth robustness. If $\widetilde{\rho}_{\varphi_s}(\mathbf{q}) > \widetilde{\epsilon} = \Delta_g + \delta_{\varphi_s}$ then $(q,0) \vDash \varphi$.*

A.4. Urban multi-mission planning without inter-UAV minimum separation requirement

In the case study of Section 7, the trajectory planning optimization is tasked with satisfying the specification φ_{PHL} (see (16)). Since the problem involves 5 UAS in a large airspace, we explore the trajectory generation approach when the inter-UAS minimum separation is ignored.

The table below shows how much time each UAS pair spends violating the minimum separation safety requirement with a minimum distance of 5 meters between UAS pairs (i.e., the specification used in the paper, $\varphi_{\text{Inter UAS}} = \square_{[0,660]}(\left\|p_i - p_j\right\| \geqslant 5)$), where this requirement is now ignored in the trajectory planning stage for the case study (Table 3).

Table 3
Total time and percentage of total flight time where each UAS pair violates the inter-UAS separation requirement $\varphi_{\text{Inter UAS}} = \square_{[0,660]}(\left\|p_i - p_j\right\| \geqslant 5)$ when the trajectory planning is done while excluding this requirement. On the contrary, as demonstrated in the paper, when $\varphi_{\text{Inter UAS}}$ is taken into account, no UAS pairs violate it.

UAS pairs i,j	Total time (s) in violation of $\varphi_{\text{Inter UAS}}$	% time in violation
1, 2	26.1	3.95%
1, 3	0	0%
1, 4	0	0%
1, 5	0.8	0.12%
2, 3	0	0%
2, 4	0	0%
2, 5	2.0	0.30%
3, 4	617.0	93.48%
3, 5	0	0%
4, 5	0	0%

References

Abbas, H., Fainekos, G., 2013. Computing descent direction of MTL robustness for non-linear systems. In: American Control Conference.

Abbas, H., Winn, A., Fainekos, G., Julius, A.A., 2014. Functional gradient descent method for metric temporal logic specifications. In: American Control Conference. https://doi.org/10.1109/ACC.2014.6859453.

Aksaray, D., Jones, A., Kong, Z., Schwager, M., Belta, C., 2016. Q-learning for robust satisfaction of signal temporal logic specifications. In: IEEE Conference on Decision and Control.

Alejo, D., Cobano, J.A., Trujillo, M.A., Viguria, A., Rodriguez, A., Ollero, A., 2012. The speed assignment problem for conflict resolution in aerial robotics. In: 2012 IEEE International Conference on Robotics and Automation, pp. 3619–3624. https://doi.org/10.1109/ICRA.2012.6225026.

Andersson, J., 2013. A General-Purpose Software Framework for Dynamic Optimization (PhD thesis). Arenberg Doctoral School, KU Leuven.

Belta, C., Sadraddini, S., 2019. Formal methods for control synthesis: An optimization perspective. Ann. Rev. Control Robot. Autonom. Syst. 2, 115–140.

Besada-Portas, E., de la Torre, L., de la Cruz, J.M., de Andrés-Toro, B., 2010. Evolutionary trajectory planner for multiple uavs in realistic scenarios. IEEE Trans. Rob. 26 (4), 619–634. https://doi.org/10.1109/TRO.2010.2048610.

Borrelli, F., Subramanian, D., Raghunathan, A.U., Biegler, L.T., 2006. Milp and nlp techniques for centralized trajectory planning of multiple unmanned air vehicles. In: American Control Conference. https://doi.org/10.1109/ACC.2006.1657644.

CNBC, 2019. Amazon wins FAA approval for Prime Air drone delivery fleet. https://www.cnbc.com/2020/08/31/amazon-prime-now-drone-delivery-fleet-gets-faa-approval.html (accessed: 02-Feb-2021).

CNN, 2016. Google drones will deliver Chipotle burritos at Virginia Tech.

Crown Consulting, Urban Air Mobility (UAM) Market Study, 2018. https://ntrs.nasa.gov/api/citations/20190001190/downloads/20190001190.pdf (accessed 01-Feb-2021).

De Neufville, R., Odoni, A.R., Belobaba, P.P., Reynolds, T.G., 2013. Airport systems: Planning, design, and management. McGraw-Hill Education.

Desai, A., Saha, I., Jianqiao, Y., Qadeer, S., Seshia, S.A., 2017. Drona: A framework for safe distributed mobile robotics. In: ACM/IEEE International Conference on Cyber-Physical Systems.

Donzé, A., Maler, O., 2010. Robust satisfaction of temporal logic over real-valued signals, in. In: Proceedings of the International Conference on Formal Modeling and Analysis of Timed Systems.

Federal Aviation Administration, 2016. https://www.faa.gov/news/updates/?newsId=85227 (accessed: 02-Feb-2021).

Fainekos, G., 2008. Robustness of temporal logic specifications (Ph.D. thesis). University of Pennsylvania.

Fainekos, G., Pappas, G., 2009. Robustness of temporal logic specifications for continuous-time signals. In: Theoret. Comput. Sci. Elsevier.

Fainekos, G.E., Kress-Gazit, H., Pappas, G.J., 2005. Hybrid controllers for path planning: A temporal logic approach, in. In: Proceedings of the 44th IEEE Conference on Decision and Control, pp. 4885–4890. https://doi.org/10.1109/CDC.2005.1582935.

Federal Aviation Administration, 2013. Integration of Civil Unmanned Aircraft Systems (UAS) in the National Airspace System (NAS) Roadmap, https://www.faa.gov/uas/resources/policy_library/media/2019_UAS_Civil_Integration_Roadmap_third_edition.pdf (accessed: 01-Feb-2021).

Federal Aviation Administration, 2015. FAA's Approach to Unmanned Aircraft Systems (UAS) Concept Maturation, Validation & Requirements Development. https://www.faa.gov/about/office_org/headquarters_offices/ang/library/events/v_vsummits/v_vsummit2015/presentations/UAS%20Concept%20Maturation-Validation%20and%20Requirements%20Development_Sherri%20Magyarits.pdf (accessed: 02-Feb-2021).

Federal Aviation Administration, 2018. FAA sUAS PART 107: THE SMALL UAS RULE, https://www.faa.gov/documentLibrary/media/Advisory_Circular/AC_107-2.pdf (accessed: 01-Feb-2021).

Federal Aviation Administration, 2020. Concept of Operations v2: Unmanned Aircraft System (UAS) Traffic Management (UTM), https://www.faa.gov/uas/research_development/traffic_management/media/UTM_ConOps_v2.pdf (accessed: 02-Feb-2021).
Hamilton, Booz Allen, 2018. Executive Briefing: Urban Air Mobility (UAM) Market Study. https://www.nasa.gov/sites/default/files/atoms/files/bah_uam_executive_briefing_181005_tagged.pdf (accessed 01-Feb-2021).
Heaton, A., Mills, J., Ansell, D., 2018. Dronehack journalism: Educating & inspiring journalists in the capacities & possibilities of unmanned aerial systems. In: AIAA SciTech Forum.
Hsl. 2021. a collection of fortran codes for large scale scientific computation., http://www.hsl.rl.ac.uk (accessed: 2021-01-22).
Kloetzer, M., Belta, C., 2008. A fully automated framework for control of linear systems from temporal logic specifications. IEEE Trans. Autom. Control 53 (1), 287–297.
Lin, Y., Saripalli, S., 2017. Sampling-based path planning for uav collision avoidance. IEEE Trans. Intell. Transp. Syst. 18 (11), 3179–3192. https://doi.org/10.1109/TITS.2017.2673778.
Maler, O., Nickovic, D., 2004. Monitoring Temporal Properties of Continuous Signals. Springer, Berlin Heidelberg.
Mellinger, D., Kushleyev, A., Kumar, V., 2012. Mixed-integer quadratic program trajectory generation for heterogeneous quadrotor teams. In: 2012 IEEE International Conference on Robotics and Automation, pp. 477–483. https://doi.org/10.1109/ICRA.2012.6225009.
Mueller, M.W., Hehn, M., DÁndrea, R., 2015. A computationally efficient motion primitive for quadrocopter trajectory generation. IEEE Trans. Rob.
National Aeronautics and Space Administration, 2018. NASA Aeronautics Research Mission Directorate (ARMD) Urban Air Mobility (UAM) Grand Challenge Industry Day, https://ntrs.nasa.gov/api/citations/20190003422/downloads/20190003422.pdf (accessed: 02-Feb-2021).
National Aeronautics and Space Administration, Advanced Air Mobility National Campaign, 2020. https://www.nasa.gov/aamnationalcampaign (accessed: 02-Feb-2021).
New York Times, 2018. Schools Offering Drone Programs, but Learning to Fly Is Just the Start, https://www.nytimes.com/2018/04/05/education/learning/schools-drone-programs.html (accessed: 01-23-2021).
Office of the Federal Register, 2018. PART 107 - SMALL UNMANNED AIRCRAFT SYSTEMS, https://www.faa.gov/news/fact_sheets/news_story.cfm?newsId=22615 (accessed: 02-Feb-2021).
Pant, Y.V., Abbas, H., Mangharam, R., 2017. Smooth operator: Control using the smooth robustness of temporal logic. In: IEEE Conference on Control Technology and Applications, pp. 1235–1240.
Pant, Y.V., Abbas, H., Quaye, R.A., Mangharam, R., 2018. 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. IEEE Press, pp. 186–197.
Pant, Y.V., Quaye, R.A., Abbas, H., Varre, A., Mangharam, R., 2019. Fly-by-logic: A tool for unmanned aircraft system fleet planning using temporal logic. In: NASA Formal Methods Symposium. Springer, pp. 355–362.
Pant, Y.V., Yin, H., Arcak, M., Seshia, S.A., 2021. Co-design of Control and Planning for Multi-rotor UAVs with Signal Temporal Logic Specifications. In: American Control Conference arXiv:2009.14363.
Ragi, S., Mittelmann, H.D., 2017. Mixed-integer nonlinear programming formulation of a uav path optimization problem. In: 2017 American Control Conference (ACC), pp. 406–411. https://doi.org/10.23919/ACC.2017.7962987.
Raman, V., Donze, A., Maasoumy, M., Murray, R.M., Sangiovanni-Vincentelli, A., Seshia, S.A., 2014. Model predictive control with signal temporal logic specifications. In: 53rd IEEE Conference on Decision and Control, pp. 81–87. https://doi.org/10.1109/CDC.2014.7039363.
Rodday, N.M., Schmidt, R.D.O., Pras, A., 2016. Exploring security vulnerabilities of unmanned aerial vehicles. In: NOMS 2016, 2016, pp. 993–994. doi: 10.1109/NOMS.2016.7502939.
Rodionova, A., Pant, Y.V., Kurtz, C., Jang, K., Abbas, H., Mangharam, R., 2021. Learning-'N-Flying: A Learning-based, Decentralized Mission Aware UAS Collision Avoidance Scheme. In: ACM Transactions on Cyber Physical Systems. ACM.
Saha, I., Rattanachai, R., Kumar, V., Pappas, G.J., Seshia, S.A., 2014. Automated composition of motion primitives for multi-robot systems from safe ltl specifications. In: IEEE/RSJ International Conference on Intelligent Robots and Systems.
Skydio gains FAA approval to conduct bridge inspections with drones in North Carolina. https://venturebeat.com/2020/10/05/skydio-gains-faa-approval-to-conduct-bridge-inspections-with-drones-in-north-carolina/ (accessed: 2020-10-09).
The National Academies, 2020. Advancing Aerial Mobility: A National Blueprint. The National Academies Press. https://doi.org/10.17226/25646.
Wächter, A., Biegler, L.T., 2020. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. Math. Program.