Learning-'N-Flying: A Learning-Based, Decentralized Mission-Aware UAS Collision Avoidance Scheme

ALËNA RODIONOVA, University of Pennsylvania, USA
YASH VARDHAN PANT, University of California, Berkeley, USA
CONNOR KURTZ, Oregon State University, USA
KUK JANG, University of Pennsylvania, USA
HOUSSAM ABBAS, Oregon State University, USA
RAHUL MANGHARAM, University of Pennsylvania, USA

Urban Air Mobility, the scenario where hundreds of manned and **Unmanned Aircraft Systems** (UASs) carry out a wide variety of missions (e.g., moving humans and goods within the city), is gaining acceptance as a transportation solution of the future. One of the key requirements for this to happen is safely managing the air traffic in these urban airspaces. Due to the expected density of the airspace, this requires fast autonomous solutions that can be deployed online. We propose Learning-'N-Flying (LNF), a multi-UAS Collision Avoidance (CA) framework. It is decentralized, works on the fly, and allows autonomous **Unmanned Aircraft System** (UAS)s managed by different operators to safely carry out complex missions, represented using Signal Temporal Logic, in a shared airspace. We initially formulate the problem of predictive collision avoidance for two UASs as a mixed-integer linear program, and show that it is intractable to solve online. Instead, we first develop Learning-to-Fly (L2F) by combining (1) learning-based decision-making and (2) decentralized convex optimization-based control. LNF extends L2F to cases where there are more than two UASs on a collision path. Through extensive simulations, we show that our method can run online (computation time in the order of milliseconds) and under certain assumptions has failure rates of less than 1% in the worst case, improving to near 0% in more relaxed operations. We show the applicability of our scheme to a wide variety of settings through multiple case studies.

CCS Concepts: • **Computer systems organization → Robotic control**; • **Computing methodologies → Neural networks;**

Additional Key Words and Phrases: Collision avoidance, unmanned aircraft systems, temporal logic, robustness, neural network, Model Predictive Control

ACM Reference format:
Alëna Rodionova, Yash Vardhan Pant, Connor Kurtz, Kuk Jang, Houssam Abbas, and Rahul Mangharam. 2021. Learning-'N-Flying: A Learning-Based, Decentralized Mission-Aware UAS Collision Avoidance Scheme. *ACM Trans. Cyber-Phys. Syst.* 5, 4, Article 35 (September 2021), 26 pages.
https://doi.org/10.1145/3447624

Authors' addresses: A. Rodionova, K. Jang, and R. Mangharam, University of Pennsylvania, Department of Electrical and Systems Engineering, Philadelphia, 200 S. 33rd Street PA, 19104, USA; emails: {alena.rodionova, jangkj, rahulm}@seas.upenn.edu; Y. V. Pant, University of California, Berkeley, Department of Electrical Engineering and Computer Sciences, Berkeley, 545N Cory Hall, Berkeley, CA 94720, USA; email: yashpant@berkeley.edu; C. Kurtz and H. Abbas, Oregon State University, School of Electrical Engineering and Computer Science, 3101 Kelley Engineering Center Corvallis, OR 97331, USA; emails: {kurtzco, houssam.abbas}@oregonstate.edu.

2378-962X/2021/09-ART35 $15.00
https://doi.org/10.1145/3447624

1 INTRODUCTION

With the increasing footprint and density of metropolitan cities, there is a need for new transportation solutions that can move goods and people around rapidly and without further stressing road networks. **Urban Air Mobility** (UAM) [12] is one such concept quickly gaining acceptance [26] as a means to improve connectivity in metropolitan cities. In such a scenario, hundreds of Autonomous manned and **Unmanned Aircraft Systems** (UASs) will carry goods and people around the city while also performing a host of other missions. A critical step toward making this a reality is safe traffic management of all the UASs in the airspace. Given the high expected UAS traffic density, as well as the short timescales of the flights, **UAS Traffic Management** (UTM) needs to be autonomous and guarantee a high degree of safety and graceful degradation in cases of overload. The first requirement for automated UTM is that its algorithms be able to accommodate a wide variety of missions, since the different operators have different goals and constraints. The second requirement is that as the number of UASs in the airspace increases, the runtimes of the UTM algorithms do not blow up—at least up to a point. The third requirement is that it must provide guaranteed collision avoidance in most use cases and degrade gracefully otherwise; that is, the determination of whether it will be able to deconflict two UASs or not must happen sufficiently fast to alert a higher-level algorithm or a human operator, say, who can impose additional constraints.

In this article we introduce and demonstrate a new algorithm, Learning-'N-Flying (LNF), for multi-UAS planning in urban airspace. LNF starts from multi-UAS missions expressed in **Signal Temporal Logic (STL)**, a formal behavioral specification language that can express a wide variety of missions and supports automated reasoning. In general, a mission will couple various UASs together through mutual separation constraints, and this coupling can cause an exponential blowup in computation. To avoid this, LNF lets every UAS plan independently of others while ignoring the mutual separation constraints. This independent planning step is performed using Fly-by-Logic, our previous UAS motion planner. An online collision avoidance procedure then handles potential collisions on an as-needed basis, i.e., when two UASs that are within communication range detect a future collision between their preplanned trajectories. Even online optimal collision avoidance between two UASs requires solving a **Mixed-Integer Linear Program (MILP)**. LNF avoids this by using a recurrent neural network that maps the current configuration of the two UASs to a sequence of discrete decisions. The network's inference step runs much faster (and its runtime is much more stable) than running an MILP solver. The network is trained offline on solutions generated by solving the MILP. To generalize from two UAS collision avoidance to multi-UAS, we introduce another component to LNF: Fly-by-Logic generates trajectories that satisfy their STL missions and a robustness tube around each trajectory. As long as the UAS is within its tube, it satisfies its mission. To handle a collision between three or more UASs, LNF shrinks the robustness tube for each trajectory in such a way that sequential two-UAS collision avoidance succeeds in deconflicting all the UASs.

We show that LNF is capable of successfully resolving collisions between UASs even within high-density airspaces and short timescales, which are exactly the scenarios expected in UAM. LNF creates opportunities for safer UAS operations and therefore safer UAM.

Contributions of This Work. In this article, we present an online, decentralized, and mission-aware UAS **Collision Avoidance** (CA) scheme that combines machine-learning-based decision-making with **Model Predictive Control (MPC)**. The main contributions of our approach are:

(1) It systematically combines machine-learning-based decision-making[1] with an MPC-based CA controller. This allows us to decouple the usually hard-to-interpret machine learning

[1]With the offline training and fast online application of the learned policy; see Sections 4.2 and 6.2.

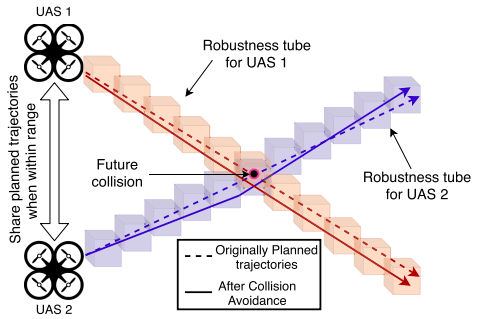


Fig. 1. Two UASs communicating their planned trajectories and cooperatively maneuvering within their *robustness tubes* to avoid a potential collision in the future.

component and the safety-critical low-level controller, and also *repair* potentially unsafe decisions by the ML components. We also present a sufficient condition for our scheme to successfully perform CA.

(2) LNF collision avoidance avoids the live-lock condition where pairwise CA continually results in the creation of collisions between other pairs of UAS.
(3) Our formulation is *mission aware*; i.e., CA does not result in violation of the UAS mission. As shown in [32], this also enables faster STL-based mission planning for a certain class of STL specifications.
(4) Our approach is computationally lightweight with a computation time of the order of 10*ms* and can be used online.
(5) Through extensive simulations, we show that the worst-case failure rate of our method is less than 1%, which is a significant improvement over other approaches including [32].

Related Work. **UTM and Automatic Collision Avoidance Approaches.** CA is a critical component of UTM. The NASA/FAA Concept of Operations [1, 19] present airspace allocation schemes where UASs are allocated airspace in the form of non-overlapping space-time polygons. Our approach is less restrictive and allows overlaps in the polygons but performs online collision avoidance on an as-needed basis. A tree-search-based planning approach for UAS CA is explored in [4]. The next-gen CA system for manned aircrafts, ACAS-X [18], is a learning-based approach that provides vertical separation recommendations. ACAS-Xu [24] relies on a lookup table to provide high-level recommendations to two UASs. It restricts desired maneuvers for CA to the vertical axis for cooperative traffic and the horizontal axis for uncooperative traffic. While we consider only the cooperative case in this work, our method does not restrict CA maneuvers to any single axis of motion. Finally, in its current form, ACAS-Xu also does not take into account any higher-level mission objectives, unlike our approach. This excludes its application to low-level flights in urban settings. The work in [8] presents a decentralized, mission-aware CA scheme but requires time of the order of seconds for the UASs to communicate and safely plan around each other, whereas our approach has a computation times in milliseconds.

Multi-agent Planning with Temporal Logic Objectives. Multi-agent planning for systems with temporal logic objectives has been well studied as a way of safe mission planning. Approaches for this usually rely on grid-based discretization of the workspace [6, 34] or a simplified abstraction of the dynamics of the agents [2, 7]. [22] combines a discrete planner with a continuous trajectory generator. Some methods [10, 16, 17] work for subsets of **Linear Temporal Logic (LTL)** that do not allow for explicit timing bounds on the mission requirements. The work in [34] allows some

explicit timing constraints. However, it restricts motion to a discrete set of motion primitives. The predictive control method of [30] uses the full STL grammar; it handles a continuous workspace and linear dynamics of robots, but its reliance on mixed-integer encoding (similar to [14, 35]) limits its practical use as seen in [27]. The approach of [29] instead relies on optimizing a smooth (non-convex) function for generating trajectories for fleets of multi-rotor UASs with STL specifications. While these methods can ensure safe operation of multi-agent systems, these are all centralized approaches, i.e., require joint planning for all agents and do not scale well with the number of agents. In our framework, we use the planning method of [29], but we let each UAS plan independently of each other in order for the planning to scale. We ensure the safe operation of all UASs in the airspace through the use of our predictive collision avoidance scheme.

Organization of the Article. The rest of the article is organized as follows. Section 2 covers preliminaries on Signal Temporal Logic and trajectory planning. In Section 3 we formalize the two-UAS CA problems, state our main assumptions, and develop a baseline centralized solution via an MILP formulation. Section 4 presents a decentralized learning-based collision avoidance framework for UAS pairs. In Section 5 we extend this approach to support cases when CA has to be performed for three or more UASs. We evaluate our methods through extensive simulations, including three case studies in Section 6. Section 7 concludes the article.

2 PRELIMINARIES: SIGNAL TEMPORAL LOGIC-BASED UAS PLANNING

Notation. For a vector $x = (x_1, \ldots, x_m) \in \mathbb{R}^m$, $\|x\|_\infty = \max_i |x_i|$.

2.1 Introduction to Signal Temporal Logic and Its Robustness

Let $\mathbb{T} = \{0, dt, 2dt, 3dt \ldots\}$ be a discrete time domain with sampling period dt and let $\mathcal{X} \subset \mathbb{R}^m$ be the state space. A *signal* is a function $\mathbf{x} : E \to \mathcal{X}$ where $E \subseteq \mathbb{T}$; the k^{th} element of $\mathbf{x}$ is written x_k, $k \geq 0$. Let $\mathcal{X}^{\mathbb{T}}$ be the set of all signals.

Signal specifications are expressed in STL [23], of which we give an informal description here. An STL formula φ is created using the following grammar:

$$\varphi := \top \mid p \mid \neg\varphi \mid \varphi_1 \vee \varphi_2 \mid \Diamond_{[a,b]}\varphi \mid \Box_{[a,b]}\varphi \mid \varphi_1 \mathcal{U}_{[a,b]}\varphi_2.$$

Here, $\top$ is logical True; p is an atomic proposition, i.e., a basic statement about the state of the system; $\neg$, $\vee$ are the usual Boolean negation and disjunction; $\Diamond$ is Eventually; $\Box$ is Always; and $\mathcal{U}$ is Until. It is possible to define the $\Diamond$ and $\Box$ in terms of Until $\mathcal{U}$, but we make them base operations because we will work extensively with them.

An STL specification φ is interpreted over a signal, e.g., over the trajectories of quad-rotors, and evaluates to either *True* or *False*. For example, operator Eventually ($\Diamond$) augmented with a time interval $\Diamond_{[a,b]}\varphi$ states that φ is *True* at some point within $[a, b]$ time units. Operator Always ($\Box$) would correspond to φ being *True* everywhere within time $[a, b]$. The following example demonstrates how STL captures operational requirements for two UASs:

Example 1 (A Two-UAS Timed Reach-avoid Problem). Two quad-rotor UASs are tasked with a mission with spatial and temporal requirements in the workspace schematically shown in Figure 2:

(1) Each of the two UASs has to reach its corresponding Goal set (shown in green) within a time of 6 seconds after starting. UAS j (where $j \in \{1, 2\}$), with position denoted by $\mathbf{p}_j$, has to satisfy: $\varphi_{\text{reach},j} = \Diamond_{[0,6]}(\mathbf{p}_j \in \text{Goal}_j)$. The *Eventually* operator over the time interval $[0, 6]$ requires UAS j to be inside the set Goal_j at some point within 6 seconds.
(2) The two UASs also have an Unsafe (in red) set to avoid, e.g., a no-fly zone. For each UAS j, this is encoded with *Always* and *Negation* operators:
$\varphi_{\text{avoid},j} = \Box_{[0,6]}\neg(\mathbf{p}_j \in \text{Unsafe})$.

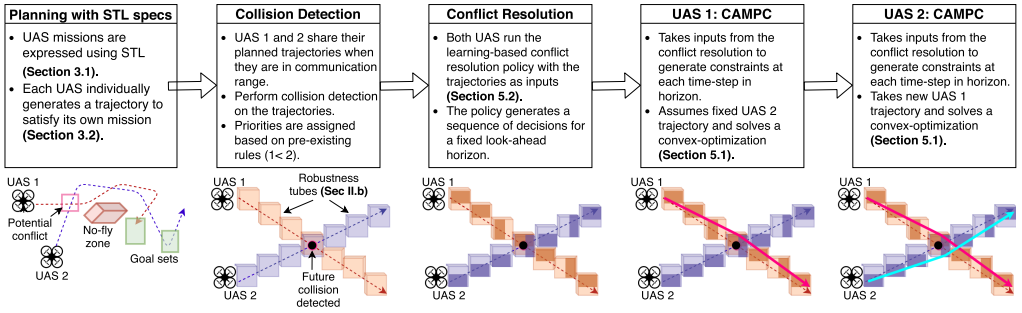


Fig. 2. Step-wise explanation and visualization of the framework. Each UAS generates its own trajectories to satisfy a mission expressed as a Signal Temporal Logic (STL) specification; e.g., regions in green are regions of interest for the UAS to visit, and the no-fly zone corresponds to infrastructure that all the UASs must avoid. When executing these trajectories, UASs communicate their trajectories to others in range to detect any collisions that may happen in the near future. If a collision is detected, the two UASs execute a conflict resolution scheme that generates a set of additional constraints that the UASs must satisfy to avoid the collision. A cooperative CA-MPC controls the UASs to best satisfy these constraints while ensuring each UAS's STL specification is still satisfied. This results in new trajectories (in solid pink and blue) that will avoid the conflict and still stay within the predefined robustness tubes.

(3) Finally, the two UASs should be separated by at least δ meters along every axis of motion: $\varphi_{\text{separation}} = \square_{[0,6]} ||\mathbf{p}_1 - \mathbf{p}_2||_\infty \geq \delta$.

The two-UAS timed reach-avoid specification is thus

$$\varphi_{\text{reach-avoid}} = \bigwedge_{j=1}^{2} (\varphi_{\text{reach},j} \wedge \varphi_{\text{avoid},j}) \wedge \varphi_{\text{separation}}. \tag{1}$$

To satisfy φ, a planning method generates trajectories $\mathbf{p}_1$ and $\mathbf{p}_2$ of a duration at least $hrz(\varphi) = 6$s, where $hrz(\varphi)$ is the time *horizon* of φ. If the trajectories satisfy the specification, i.e., $(\mathbf{p}_1, \mathbf{p}_2) \models \varphi$, then the specification φ evaluates to *True*; otherwise it is *False*. In general, an upper bound for the time horizon can be computed as shown in [30]. In this work, we consider specifications such that the horizon is bounded. More details on STL can be found in [23] or [30]. In this article, we consider discrete-time STL semantics, which are defined over discrete-time trajectories.

The *Robustness* value [9] $\rho_\varphi(\mathbf{x})$ of an STL formula φ with respect to the signal $\mathbf{x}$ is a real-valued function of $\mathbf{x}$ that has the important following property:

Theorem 2.1 ([9]). *(i) For any $\mathbf{x} \in \mathcal{X}^{\mathbb{T}}$ and STL formula φ, if $\rho_\varphi(\mathbf{x}) < 0$, then $\mathbf{x}$ violates φ, and if $\rho_\varphi(\mathbf{x}) > 0$, then $\mathbf{x}$ satisfies φ. The case $\rho_\varphi(\mathbf{x}) = 0$ is inconclusive.*

(ii) Given a discrete-time trajectory $\mathbf{x}$ such that $\mathbf{x} \models \varphi$ with robustness value $\rho_\varphi(\mathbf{x}) = r > 0$, then any trajectory $\mathbf{x}'$ that is within r of $\mathbf{x}$ at each time step, i.e., $||x_k - x'_k||_\infty < r$, $\forall k \in \mathbb{H}$, is such that $\mathbf{x}' \models \varphi$ (also satisfies φ).

2.2 UAS Planning with STL Specifications

Fly-by-logic [27, 29] generates trajectories by centrally planning for fleets of UASs with STL specifications, e.g., the specification $\varphi_{reach\text{-}avoid}$ of example 1. It maximizes the robustness function by picking waypoints for all UASs through a centralized, non-convex optimization.

While successful in planning for multiple multi-rotor UASs, performance degrades as the number of UASs increases, in particular because for N UAS, $\binom{N}{2}$ terms are needed for specifying the pairwise separation constraint $\varphi_{separation}$. For these reasons, the method cannot be used for

real-time planning. In this work, we use the underlying optimization of [29] to generate trajectories but ignore the mutual separation requirement, allowing each UAS to independently (and in parallel) solve for their own STL specification. For the timed reach-avoid specification (1) in example 1, this is equivalent to each UAS generating its own trajectory to satisfy $\varphi_j = \varphi_{reach,j} \wedge \varphi_{avoid,j}$, independently of the other UASs. Ignoring the collision avoidance requirement $\varphi_{separation}$ in the planning stage allows for the specification of (1) to be decoupled across UASs. Therefore, this approach requires *online* UAS collision avoidance. This is covered in the following section.

3 PROBLEM FORMULATION: MISSION-AWARE UAS COLLISION AVOIDANCE

We consider the case where two UASs flying preplanned trajectories are required to perform collision avoidance if their trajectories are on path for a *conflict*.

Definition 1 (Two-UAS Conflict). Two UASs, with discrete-time positions $\mathbf{p}_1$ and $\mathbf{p}_2$, are said to be in *conflict* at time step k if $||p_{1,k} - p_{2,k}||_\infty < \delta$, where δ is a predefined minimum separation distance.[2] Here, $p_{j,k}$ represents the position of UAS j at time step k.

While flying their independently planned trajectories, two UASs that are within communication range share an H-step look-ahead of their trajectories and check for a potential conflict in those H steps. We assume the UASs can communicate with each other in a manner that allows for enough advance notice for avoiding collisions, e.g., using 5G technology. While the details of this are beyond the scope of this article, we formalize this assumption as follows:

Assumption 1. *The two UASs in conflict have a communication range that is at least greater than their n-step forward reachable set [5] ($n \geq 1$).*[3] *That is, the two UASs will not collide immediately in at least the next n-time steps, enabling them to communicate with each other to avoid a collision. Here n is potentially dependent on the communication technology being used.*

Definition 2 (Robustness Tube). Given an STL formula φ and a discrete-time position trajectory $\mathbf{p}_j$ that satisfies φ (with associated robustness ρ), the (discrete) *robustness tube* around $\mathbf{p}_j$ is given by $\mathbf{P}_j = \mathbf{p}_j \oplus \mathbb{B}_\rho$, where $\mathbb{B}_\rho$ is a 3D cube with sides 2ρ and $\oplus$ is the Minkowski sum operation ($A \oplus B := \{a + b \mid a \in A, b \in B\}$). We say the *radius* of this tube is ρ (in the inf-norm sense).

Robustness tube defines the space around the UAS trajectory, such that as long as the UAS stays within its robustness tube, it will satisfy the STL specification for which it was generated. See examples of the robustness tubes in Figures 1 and 2.

The following assumption relates the minimum allowable radius ρ of the robustness tube to the minimum allowable separation δ between two UASs.

Assumption 2. *For each of the two UASs in conflict, the radius of the robustness tube is greater than $\delta/2$, i.e., $\min(\rho_1, \rho_2) \geq \delta/2$, where ρ_1 and ρ_2 are the robustness of UAS 1 and 2, respectively.*

This assumption defines the case where the radius of the robustness tube is wide enough to have two UASs placed along opposing edges of their respective tubes and still achieve the minimum separation between them. We assume that all the trajectories generated by the independent planning have sufficient robustness to satisfy this assumption (see Section 2.2). Now we define the problem of collision avoidance with satisfaction of STL specifications:

[2] A more general polyhedral constraint of the form $M(p_{1,k} - p_{2,k}) < q$ can be used for defining the conflict.
[3] This set can be computed offline as we know the dynamics and actuation limits for each UAS.

Problem 1. *Given two planned H-step UAS trajectories $\mathbf{p}_1$ and $\mathbf{p}_2$ that have a conflict, the collision avoidance problem is to find a new sequence of positions $\mathbf{p}'_1$ and $\mathbf{p}'_2$ that meet the following conditions:*

$$||p'_{1,k} - p'_{2,k}|| \geq \delta, \forall k \in \{0, \ldots, H\} \tag{2a}$$

$$p'_{j,k} \in P_{j,k}, \forall k \in \{0, \ldots, H\}, \forall j \in \{1, 2\}. \tag{2b}$$

That is, we need a new trajectory for each UAS such that they achieve minimum separation distance and also stay within the robustness tube around their originally planned trajectories.

Convex Constraints for Collision Avoidance. Let $z_k = p_{1,k} - p_{2,k}$ be the difference in UAS positions at time step k. For two UASs not to be in conflict, we need

$$z_k \notin \mathbb{B}_{\delta/2}, \ \forall k \in \{0, \ldots, H\}. \tag{3}$$

This is a non-convex constraint. For a computationally tractable controller formulation that solves Problem 1, we define convex constraints that when satisfied imply Equation (3). The 3D cube $\mathbb{B}_{\delta/2}$ can be defined by a set of linear inequality constraints of the form $\widetilde{M}^i z \leq \widetilde{q}^i$, $\forall i \in \{1, \ldots, 6\}$. Equation (3) is satisfied when $\exists i \, | \widetilde{M}^i z > \widetilde{q}_i$. Let $M = -\widetilde{M}$ and $q = -\widetilde{q}$; then $\forall i \in \{1, \ldots, 6\}$:

$$M^i (p_{1,k} - p_{2,k}) < q^i \Rightarrow (p_{1,k} - p_{2,k}) \notin \mathbb{B}_{\delta/2}. \tag{4}$$

Intuitively, picking one i at time step k results in a configuration (in position space) where the two UASs are separated in one of two ways along one of three axes of motion.[4] For example, if at time step k we select i with corresponding $M^i = [0, 0, 1]$ and $q^i = -\delta$, it implies that UAS 2 flies over UAS 1 by δ meters, and so on.

A Centralized Solution via a MILP Formulation. Here, we formulate a MILP to solve the two-UAS CA problem of problem 1 in a predictive, receding horizon manner. For the formulation, we consider an H-step look ahead that contains the time steps where the two UASs are in conflict. Let the dynamics of either UAS[5] be of the form $x_{k+1} = Ax_k + Bu_k$. At each time step k, the UAS state is defined as $x_k = [p_k, v_k]^T \in \mathbb{R}^6$, where p and v are the UAS positions and velocities in the 3D space. Let C be the observation matrix such that $p_k = Cx_k$. The inputs $u_k \in \mathbb{R}^3$ are the thrust, roll, and pitch of the UASs. The matrices A and B are obtained through linearization of the UAS dynamics around hover and discretization in time; see [21] and [28] for more details. Let $\mathbf{x}_j \in \mathbb{R}^{6(H+1)}$ be the preplanned full state trajectories, $\mathbf{x}'_j \in \mathbb{R}^{6(H+1)}$ the new full state trajectories, and $\mathbf{u}'_j \in \mathbb{R}^{3H}$ the new controls to be computed for the UAS $j = 1, 2$. Let $\mathbf{b} \in \{0, 1\}^{6(H+1)}$ be binary decision variables, and μ is a large positive number; then the MILP problem is defined as

$$\min_{\mathbf{u}'_1, \mathbf{u}'_2, \mathbf{b}} J(\mathbf{x}'_1, \mathbf{u}'_1, \mathbf{x}'_2, \mathbf{u}'_2)$$

$$x'_{j,0} = x_{j,0}, \ \forall j \in \{1, 2\}$$

$$x'_{j,k+1} = Ax'_{j,k} + Bu'_{j,k}, \ \forall k \in \{0, \ldots, H-1\}, \ \forall j \in \{1, 2\}$$

$$Cx'_{j,k} \in P_{j,k}, \ \forall k \in \{0, \ldots, H\}, \ \forall j \in \{1, 2\}$$

[4] Two ways along one of three axes defines 6 options, $i \in \{1, \ldots, 6\}$.

[5] For simplicity we assume both UASs have identical dynamics associated with multi-rotor robots; however, our approach would work otherwise.

$$
\begin{aligned}
M^i C\left(x'_{1,k} - x'_{2,k}\right) &\leq q_i + \mu\left(1 - b_k^i\right),\ \forall k \in \{0,\ldots,H\}, \forall i \in \{1,\ldots,6\} \\
\sum_{i=1}^{6} b_k^i &\geq 1,\ \forall k \in \{0,\ldots,H\} \\
u'_{j,k} &\in U,\ \forall k \in \{0,\ldots,H\},\ \forall j \in \{1,2\} \\
x'_{j,k} &\in X,\ \forall k \in \{0,\ldots,H+1\}, \forall j \in \{1,2\}.
\end{aligned} \tag{5}
$$

Here b_k^i encodes action $i = 1,\ldots,6$ taken for avoiding a collision at time step k, which corresponds to a particular side of the cube $\mathbb{B}_{\delta/2}$. Function J could be any cost function of interest; we use $J = 0$ to turn Equation (5) into a feasibility problem. A solution (when it exists) to this MILP results in new trajectories $(\mathbf{p}'_1, \mathbf{p}'_2)$ that avoid collisions and stay within their respective robustness tubes of the original trajectories, and hence are a solution to problem 1.

Such optimization is joint over both UASs. It is impractical as it would either require one UAS to solve for both or each UAS to solve an identical optimization that would also give information about the control sequence of the other UAS. Solving this MILP in an online manner is also intractable, as we show in Section 6.2.1.

4 LEARNING-2-FLY: DECENTRALIZED COLLISION AVOIDANCE FOR UAS PAIRS

To solve problem 1 in an online and decentralized manner, we develop our framework, **Learning-to-Fly (L2F)**. Given a predefined priority among the two UASs, this combines a learning-based **conflict resolution** (CR) scheme (running aboard each UAS) that gives us the discrete components of the **Mixed Integer Linear Program** (MILP) formulation (Equation (5)), and a cooperative collision avoidance MPC for each UAS to control them in a decentralized manner. We assume that the two UASs can communicate their preplanned N-step trajectories $\mathbf{p}_1$, $\mathbf{p}_2$ to each other (refer to Section 2.2), and then L2F solves problem 1 by following these steps (also see Algorithm 1):

(1) **Conflict resolution:** UAS 1 and 2 make a *sequence of decisions*, $\mathbf{d} = (d_0,\ldots,d_H)$, to avoid collision. Each $d_k \in \{1,\ldots 6\}$ represents a particular choice of M and q at time step k; see Equation (4). Section 4.2 will describe our proposed learning-based method for picking d_k.
(2) **UAS 1 CA-MPC:** UAS 1 takes the conflict resolution sequence $\mathbf{d}$ from step 1 and solves a convex optimization to try to deconflict, while assuming UAS 2 maintains its original trajectory. After the optimization, the new trajectory for UAS 1 is sent to UAS 2.
(3) **UAS 2 CA-MPC:** (If needed) UAS 2 takes the same conflict resolution sequence $\mathbf{d}$ from step 1 and solves a convex optimization to try to avoid UAS 1's new trajectory. Section 4.1 provides more details on CA-MPC steps 2 and 3.

The visualization of the above steps is presented in Figure 2. Such a decentralized approach differs from the centralized MILP approach, where both the binary decision variables and continuous control variables for each UAS are decided concurrently.

4.1 Distributed and Cooperative Collision Avoidance MPC (CA-MPC)

Let $\mathbf{x}_j$ be the preplanned trajectory of UAS j, $\mathbf{x}_{avoid}$ be the preplanned trajectory of the other UAS to which j must attain a minimum separation, and $prty_j \in \{-1,+1\}$ be the priority of UAS j. Assume a decision sequence $\mathbf{d}$ is given: at each k in the collision avoidance horizon, the UASs are to avoid each other by respecting Equation (4), namely $M^{d_k}(p_{1,k} - p_{2,k}) < q^{d_k}$. Then each UAS $j = 1,2$ solves the following **Collision-Avoidance MPC optimization (CA-MPC)**:

CA-MPC$_j(\mathbf{x}_j,\ \mathbf{x}_{avoid},\ \mathbf{P}_j,\ \mathbf{d},\ prty_j)$:

$$\begin{aligned}
&\min_{\mathbf{u}'_j,\boldsymbol{\lambda}_j} \sum_{k=0}^{H} \lambda_{j,k} \\
x'_{j,0} &= x_{j,0} \\
x'_{j,k+1} &= Ax'_{j,k} + Bu'_{j,k},\ \forall k \in \{0,\dots,H-1\} \\
Cx'_{j,k} &\in P_{j,k},\ \forall k \in \{0,\dots,H\} \\
prty_j \cdot M^{d_k} C\left(x_{avoid,k} - x'_{j,k}\right) &\leq q^{d_k} + \lambda_{j,k},\ \forall k \in \{0,\dots,H\} \\
\lambda_{j,k} &\geq 0,\ \forall k \in \{0,\dots,H\} \\
u'_{j,k} &\in U,\ \forall k \in \{0,\dots,H\} \\
x'_{j,k} &\in X,\ \forall k \in \{0,\dots,H+1\}.
\end{aligned} \tag{6}$$

This MPC optimization tries to find a new trajectory $\mathbf{x}'_j$ for the UAS j that minimizes the slack variables $\lambda_{j,k}$ that correspond to violations in the minimum separation constraint (Equation (4)) w.r.t the pre-planned trajectory $\mathbf{x}_{avoid}$ of the UAS in conflict. The constraints in Equation (6) ensure that UAS j respects its dynamics, input constraints, and state constraints to stay inside the robustness tube. An objective of 0 implies that UAS j's new trajectory satisfies the minimum separation between the two UASs; see Equation (4).[6]

CA-MPC Optimization for UAS 1. UAS 1, with lower priority, $prty_1 = -1$, first attempts to resolve the conflict for the given sequence of decisions $\mathbf{d}$:

$$(\mathbf{x}'_1, \mathbf{u}'_1, \boldsymbol{\lambda}_1) = \textbf{CA-MPC}_1(\mathbf{x}_1, \mathbf{x}_2, \mathbf{P}_1, \mathbf{d}, -1). \tag{7}$$

An objective of 0 implies that UAS 1 alone can satisfy the minimum separation between the two UASs. Otherwise, UAS 1 alone could not create separation and UAS 2 now needs to maneuver as well.

CA-MPC Optimization for UAS 2. If UAS 1 is unsuccessful at collision avoidance, UAS 1 communicates its current revised trajectory $\mathbf{x}'_1$ to UAS 2, with $prty_2 = +1$. UAS 2 then creates a new trajectory $\mathbf{x}'_2$ (w.r.t the same decision sequence $\mathbf{d}$):

$$(\mathbf{x}'_2, \mathbf{u}'_2, \boldsymbol{\lambda}_2) = \textbf{CA-MPC}_2\ (\mathbf{x}_2, \mathbf{x}'_1, \mathbf{P}_2, \mathbf{d}, +1). \tag{8}$$

Algorithm 1 is designed to be computationally lighter than the MILP approach (Equation (5)), but unlike the MILP it is not complete.

The solution of CA-MPC can be defined as follows:

Definition 4.1 (Zero-slack Solution). The solution of the CA-MPC optimization (Equation (6)) is called the *zero-slack solution* if for a given decision sequence $\mathbf{d}$ either

1) there exists an optimal solution of Equation (6) such that $\sum_k \lambda_{1,k} = 0$ or

2) Equation (6) is feasible with $\sum_k \lambda_{1,k} > 0$ and there exists an optimal solution of Equation (6) such that $\sum_k \lambda_{2,k} = 0$.

Theorem 4.1 defines the sufficient condition for CA and Theorem 4.2 makes important connections between the slack variables in CA-MPC formulation and binary variables in MILP. Both

[6]Enforcing the separation constraint at each time step can lead to a restrictive formulation, especially in cases where the two UASs are only briefly close to each other. This does, however, give us an optimization with a structure that does not change over time and can avoid collisions in cases where the UAS could run across each other more than once in quick succession (e.g., https://tinyurl.com/arc-case), which is something ACAS-Xu was not designed for.

ALGORITHM 1: Learning-to-Fly: Decentralized and cooperative collision avoidance for two UASs. Also see Figure 2.

Notation : $(\mathbf{x}_1', \mathbf{x}_2', \mathbf{u}_1', \mathbf{u}_2') = \mathbf{L2F}(\mathbf{x}_1, \mathbf{x}_2, \mathbf{P}_1, \mathbf{P}_2)$
Input: Preplanned trajectories $\mathbf{x}_1$, $\mathbf{x}_2$, robustness tubes $\mathbf{P}_1$, $\mathbf{P}_2$
Output: Sequence of control signals $\mathbf{u}_1'$, $\mathbf{u}_2'$ for the two UASs, updated trajectories $\mathbf{x}_1'$, $\mathbf{x}_2'$
Get $\mathbf{d}$ from conflict resolution
UAS 1 solves CA-MPC optimization (6): $(\mathbf{x}_1', \mathbf{u}_1', \boldsymbol{\lambda}_1) = \mathbf{CA\text{-}MPC}_1(\mathbf{x}_1, \mathbf{x}_2, \mathbf{P}_1, \mathbf{d}, -1)$
if $\sum_k \lambda_{1,k} = 0$ **then**
  Done: UAS 1 alone has created separation; set $\mathbf{u}_2' = \mathbf{u}_2$
else
  UAS 1 transmits solution to UAS 2
  UAS 2 solves CA-MPC optimization (6): $(\mathbf{x}_2', \mathbf{u}_2', \boldsymbol{\lambda}_2) = \mathbf{CA\text{-}MPC}_2(\mathbf{x}_2, \mathbf{x}_1', \mathbf{P}_2, \mathbf{d}, +1)$
  if $\sum_k \lambda_{2,k} = 0$ **then**
    Done: UAS 2 has created separation
  else
    if $||p_{1,k}' - p_{2,k}'|| \geq \delta$, $\forall k = 0, \ldots, H$ **then**
      Done: UAS 1 and UAS 2 created separation
    else
      Not done: UAS still violate Equation (2a)
    end
  end
end
Apply control signals $\mathbf{u}_1'$, $\mathbf{u}_2'$ if **Done**; else **Fail**.

theorems are direct consequences of the construction of CA-MPC optimizations. We omit the proofs for brevity.

Theorem 4.1 (Sufficient Condition for CA). *Zero-slack solution of Equation (6) implies that the resulting trajectories for two UASs are non-conflicting and within the robustness tubes of the initial trajectories.*[7]

Theorem 4.2 (Existence of the Zero-slack Solution). *Feasibility of the MILP problem (Equation (5)) implies the existence of the zero-slack solution of CA-MPC optimization (Equation (6)).*

Theorem 4.2 states that the binary decision variables b_k^i selected by the feasible solution of the MILP problem (Equation (5)), when used to select the constraints (defined by M, q) for the CA-MPC formulations for UAS 1 and 2, imply the existence of a zero-slack solution of Equation (6).

4.2 Learning-Based Conflict Resolution

Motivated by Theorem 4.2, we propose to learn offline the conflict resolution policy from the MILP solutions and then online use already learned policy. To do so, we use a *Long Short-Term Memory* (LSTM) [13] recurrent neural network augmented with fully connected layers. LSTMs perform better than traditional recurrent neural networks on sequential prediction tasks [11].

The network is trained to map a difference trajectory $\mathbf{z} = \mathbf{x}_1 - \mathbf{x}_2$ (as in Equation (3)) to a decision sequence $\mathbf{d}$ that deconflicts preplanned trajectories $\mathbf{x}_1$ and $\mathbf{x}_2$. For creating the training

[7]Theorem 4.1 formulates a conservative result as Equation (4) is a convex under approximation of the originally non-convex collision avoidance constraint (Equation (3)). Indeed, non-zero slack $\exists k | \lambda_{2,k} > 0$ does not necessarily imply the violation of the mutual separation requirement (Equation (2a)). The control signals u_1', u_2' computed by Algorithm 1 can therefore in some instances still create separation between UASs even when the conditions of Theorem 4.1 are not satisfied.

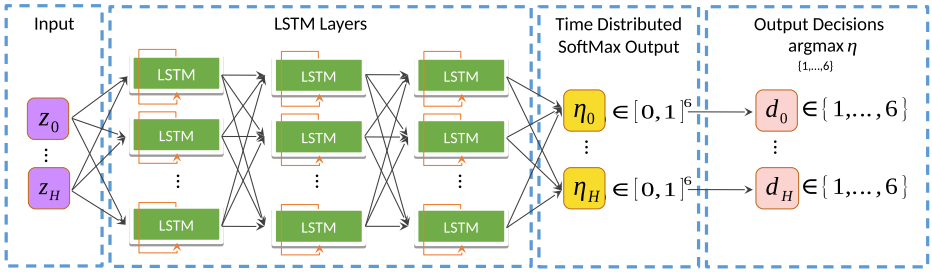


Fig. 3. Proposed LSTM model architecture for CR-S. LSTM layers are shown unrolled over H time steps. The inputs are z_k, which are the differences between the planned UAS positions, and the outputs are decisions d_k for conflict resolution at each time k in the horizon.

set, $\mathbf{d}$ is produced by solving the MILP problem (Equation (5)), i.e., obtaining a sequence of binary decision variables $\mathbf{b} \in \{0,1\}^{6(H+1)}$ and translating it into the decision sequence $\mathbf{d} \in \{1,\ldots,6\}^{H+1}$.

The proposed architecture is presented in Figure 3. The input layer is connected to the block of three stacked LSTM layers. The output layer is a time distributed dense layer with a softmax activation function that produces the class probability estimate $\eta_k = [\eta_k^1,\ldots,\eta_k^6]^\top$ for each $k \in \{0,\ldots,H\}$, which corresponds to a decision $d_k = \text{argmax}_{i=1,\ldots 6}\eta_k^i$.

4.3 Conflict Resolution Repairing

The total number of possible CR decision sequences over a time horizon of H steps is H^6. Learning-based collision resolution produces only one such CR sequence, and since it is not guaranteed to be correct, an inadequate CR sequence might lead to the CA-MPC being unable find a feasible solution of Equation (6), i.e., a failure in resolving a collision. To make the CA algorithm more resilient to such failures, we propose a heuristic that instead of generating only one CR sequence generates a number of slightly modified sequences, aka backups, with an intention of increasing the probability of finding an overall solution for CA. We call it a *CR repairing algorithm*. We propose the following scheme for CR repairing.

4.3.1 Naïve Repairing Scheme for Generating CR Decision Sequences. The naïve repairing algorithm is based on the initial supervised learning CR architecture; see Section 4.2. The proposed DNN model for CR has the output layer with a softmax activation function that produces the class probability estimates $\eta_k = [\eta_k^1,\ldots,\eta_k^6]^\top$ for each time step k; see Figure 3. Discrete decisions were chosen as

$$d_k = \underset{i=1,\ldots 6}{\text{argmax}}\ \eta_k^i, \tag{9}$$

which corresponds to the highest probability class for time step k. Denote such choice of d_k as d_k^1.

Analogously to the idea of top-1 and top-S accuracy rates used in image classification [33], where not only the highest predicted class but also the top S most likely labels count, we define higher-order decisions d_k^s as follows: instead of choosing the highest probability class at time step k, one could choose the second-highest probability class ($s = 2$), third-highest ($s = 3$), up to the sixth-highest ($s = 6$).

Formally, the second-highest probability class choice d_k^2 is defined as

$$d_k^2 = \underset{i=1,\ldots 6,\ i\neq d_k^1}{\text{argmax}}\ \eta_k^i. \tag{10}$$

In the same manner, we define decisions up to d_k^6. The general formula for the sth-highest probability class, decision d_k^s is defined as follows ($s = 1, \ldots, 6$):

$$d_k^s = \underset{i=1,\ldots 6,\ i \neq d_k^j \ \forall j<s}{\operatorname{argmax}} \eta_k^i. \tag{11}$$

Using Equation (11) to generate decisions d_k at time step k, we define the naïve scheme for generating new decision sequences $\mathbf{d}'$ following Algorithm 2.

ALGORITHM 2: Naïve scheme for CR repairing

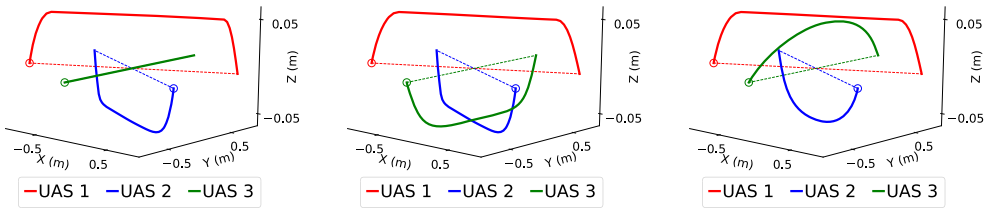
Notation : $(\mathbf{x}_1', \mathbf{x}_2', \mathbf{u}_1', \mathbf{u}_2') = \textbf{Repairing}(\mathbf{x}_1, \mathbf{x}_2, \mathbf{P}_1, \mathbf{P}_2, \Upsilon)$
Input: Preplanned trajectories $\mathbf{x}_1$, $\mathbf{x}_2$, robustness tubes $\mathbf{P}_1$, $\mathbf{P}_2$, original decision sequence $\mathbf{d}$, class probability estimates η, set of collision indices: $\Upsilon = \{k : \ ||p'_{1,k} - p'_{2,k}|| < \delta, \ 0 \leq k \leq H\}$.
Output: Sequence of control signals $\mathbf{u}_1'$, $\mathbf{u}_2'$ for the two UASs, updated trajectories $\mathbf{x}_1'$, $\mathbf{x}_2'$
for $s = 2, \ldots, 6$ **do**
  Define repaired sequence $\mathbf{d}'$ using naïve scheme as follows:
  — $\forall k \notin \Upsilon : \ d'_k = d_k$
  — $\forall k \in \Upsilon : \ d'_k = d_k^s = \operatorname{argmax}_{i=1,\ldots 6,\ i \neq d_k^j \ \forall j<s} \eta_k^i$
  $(\mathbf{x}_1', \mathbf{x}_2', \mathbf{u}_1', \mathbf{u}_2') = \textbf{CA-MPC}(\mathbf{x}_1, \mathbf{x}_2, \mathbf{P}_1, \mathbf{P}_2, \mathbf{d}')$
  if $||p'_{1,k} - p'_{2,k}|| \geq \delta, \ \forall k = 0, \ldots, N$ **then**
    Break: Repaired CR sequence $\mathbf{d}'$ led to UAS 1 and UAS 2 creating separation
  end
end
if $||p'_{1,k} - p'_{2,k}|| \geq \delta, \ \forall k = 0, \ldots, H$ **then**
  $\mathbf{d}' = \mathbf{d}$: Repairing failed. Return trajectories for the original decision sequence.
end

Example 4.3. Let the horizon of interest be only $H = 5$ time steps and the initially obtained decision sequence be $\mathbf{d} = (1, 1, 1, 1, 1)$. Given that the collision was detected at time steps 2 and 3, i.e., $\Upsilon = (2, 3)$, let the second-highest probability decisions be $d_2^2 = 3$ and $d_3^2 = 5$. Then the proposed repaired decision sequence is $\mathbf{d}' = (1, 1, 3, 5, 1)$. If such CR sequence $\mathbf{d}'$ still violates the mutual separation requirement, then the naïve repairing scheme will propose another decision sequence using the third-highest probability decisions d_3. Let $d_2^3 = 2$ and $d_3^3 = 3$; then $\mathbf{d}' = (1, 1, 2, 3, 1)$. If it fails again, the next generated sequence will use fourth-highest decisions, and so on up to the fifth iteration of the algorithm (requires d_k^6 estimates). If none of the sequences managed to create separation, the original CR sequence $\mathbf{d} = (1, 1, 1, 1, 1)$ will be returned.

Other variations of the naïve scheme are possible. For example, one can use an augmented set of collision indices Υ or another order of decisions d_k across the time indices, e.g., replace decisions d_k one by one rather than all d_k for collision indices Υ at once. Moreover, other CR repairing schemes can be efficient and should be explored. We leave it for future work.

5 LEARNING-'N-FLYING: DECENTRALIZED COLLISION AVOIDANCE FOR MULTI-UAS FLEETS

The L2F framework of Section 4 was tailored for CA between two UASs. When more than two UASs are simultaneously on a collision path, applying L2F pairwise for all UASs involved might not necessarily result in all future collisions being resolved. Consider the following example:



(a) L2F for pair UAS 1, UAS 2. Pairwise separations: $\delta_{12} = 0.11$m, $\delta_{13} = 0.06$m, $\delta_{23} = 0.05$m.

(b) L2F for pair UAS 1, UAS 3. Pairwise separations: $\delta_{12} = 0.11$m, $\delta_{13} = 0.11$m, $\delta_{23} = 0.04$m.

(c) L2F for pair UAS 2, UAS 3 results. Pairwise separations: $\delta_{12} = 0.11$m, $\delta_{13} = 0.01$m, $\delta_{23} = 0.1$m.

Fig. 4. Sequential L2F application for the three-UAS scenario. Preplanned colliding trajectories are depicted in dashed lines. Simultaneous collision is detected at point $(0, 0, 0)$. The updated trajectories generated by L2F are depicted in solid color. Initial positions of UAS are marked by "O."

Example 5.1. Figure 4 depicts an experimental setup. Scenario consists of three UASs that must reach desired goal states within 4 seconds while avoiding each other; minimum allowed separation is set to $\delta = 0.1m$. Initially preplanned UAS trajectories have a simultaneous collision across all UASs located at $(0, 0, 0)$. Robustness tube radii were fixed at $\rho = 0.055$ and UAS priorities were set in increasing order, e.g., UASs with a lower index had a lower priority: $1 < 2 < 3$. The first application of L2F led to resolving collision for UAS 1 and UAS 2; see Figure 4(a). The second application resolved collision for UAS 1 and UAS 3 by UAS 3 deviating vertically downward; see Figure 4(b). The third application led to UAS 3 deviating vertically upward, which resolved collision for UAS 2 and UAS 3 but created a re-appeared violation of minimum separation for UAS 1 and UAS 3 in the middle of their trajectories; see Figure 4(c).

To overcome this *live-lock*-like issue, where repeated pairwise applications of L2F only result in new conflicts between other pairs of UASs, we propose a modification of L2F called LNF. The LNF framework is based on pairwise application of L2F but also incorporates a ***Robustness Tube Shrinking* (RTS)** process described in Section 5.1 after every L2F application. The overall LNF framework is presented in Algorithm 3. Section 6.3 presents extensive simulations to show the applicability of the LNF scheme to scenarios where more than two UASs are on collisions paths, including in high-density UAS operations.

ALGORITHM 3: Learning-'N-Flying: Decentralized and cooperative collision avoidance for multi-UAS fleets. Applied in a receding horizon manner by each UAS i.

Input: Preplanned fleet trajectories $\mathbf{x}_i$, initial robustness tubes $\mathbf{P}_i$, UAS priorities
Output: New trajectories $\mathbf{x}'_i$, new robustness tubes $\mathbf{P}'_i$, control inputs $u'_{i,0}$
Each UAS i detects the set of UASs that it is in conflict with: $S = \{j \mid \exists k \; ||p_{i,k} - p_{j,k}|| < \delta, \; 0 \le k \le H\}$
Order S by the UAS priorities
for $j \in S$ **do**
  $(\mathbf{x}'_i, \mathbf{x}'_j, \mathbf{u}'_i, \mathbf{u}'_j) = \mathbf{L2F}(\mathbf{x}_i, \mathbf{x}_j, \mathbf{P}_i, \mathbf{P}_j)$, see Section 4
  if $\Upsilon = \{k : \; ||p'_{i,k} - p'_{j,k}|| < \delta, \; 0 \le k \le H\} \neq \emptyset$ **then**
    $(\mathbf{x}'_i, \mathbf{x}'_j, \mathbf{u}'_i, \mathbf{u}'_j) = \mathbf{Repairing}(\mathbf{x}_i, \mathbf{x}_j, \mathbf{P}_i, \mathbf{P}_j, \Upsilon)$
  end
  $(\mathbf{P}'_i, \mathbf{P}'_j) = \mathbf{RTS}\,(\mathbf{x}'_i, \mathbf{x}'_j, \mathbf{P}_i, \mathbf{P}_j)$
end
Apply controls $u'_{i,0}$ for the initial time step of the receding horizon

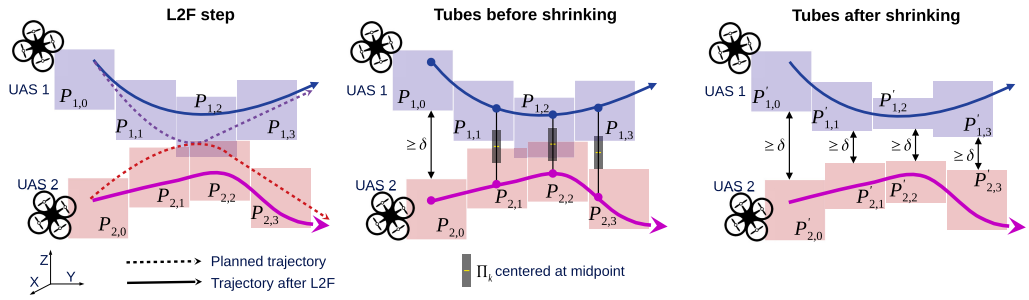


Fig. 5. Visualization of the robustness tube shrinking process.

5.1 Robustness Tube Shrinking

The high-level of idea of RTS is that, when two trajectories are de-collided by L2F, we want to constrain their further modifications by L2F so as not to induce new collisions. In Example 5.1, after collision-free $\mathbf{x}'_1$ and $\mathbf{x}'_2$ are produced by L2F and before $\mathbf{x}'_2$ and $\mathbf{x}_3$ are de-collided, we want to constrain any modification to $\mathbf{x}'_2$ s.t. it does not collide again with $\mathbf{x}'_1$. Since trajectories are constrained to remain within robustness tubes, we simply shrink those tubes to achieve this. The amount of shrinking is δ, the minimum separation. RTS is described in Algorithm 4.

ALGORITHM 4: Robustness tubes shrinking. Also see Figure 5.

Notation : $(\mathbf{P}'_1, \mathbf{P}'_2) = \mathbf{RTS}\,(\mathbf{x}'_1, \mathbf{x}'_2, \mathbf{P}_1, \mathbf{P}_2)$
Input: New trajectories $\mathbf{x}'_1$, $\mathbf{x}'_2$ generated by L2F, initial robustness tubes $\mathbf{P}_1$, $\mathbf{P}_2$
Output: New robustness tubes $\mathbf{P}'_1$, $\mathbf{P}'_2$
Set $msep = \min_{0\le k\le H} ||p'_{1,k} - p'_{2,k}||$
for $k = 0, \ldots, H$ **do**
  if dist$(P_{1,k}, P_{2,k}) \ge \delta$ **then**
    No shrinking required: $P'_{1,k} = P_{1,k},\ P'_{2,k} = P_{2,k}$
  else
    Determine the axis (X, Y, or Z) of maximum separation between $p'_{1,k}$ and $p'_{2,k}$
    Define the 3D box Π_k with edges of size $\min(msep, \delta)$ along the determined axis and infinite edges along other two axes
    Center Π_k at the midpoint between $p'_{1,k}$ and $p'_{2,k}$
    Remove Π_k from both tubes: $P'_{1,k} = P_{1,k} \setminus \Pi_k,\ P'_{2,k} = P_{2,k} \setminus \Pi_k$
  end
end

Example 5.2. Figure 5(a) presents the initial discrete-time robustness tubes and trajectories for UAS 1 and UAS 2. Successful application of L2F resolves the detected collision between initially planned trajectories $\mathbf{p}_1$, $\mathbf{p}_2$, depicted in the dashed line. New non-colliding trajectories $\mathbf{p}'_1$ and $\mathbf{p}'_2$ produced by L2F are in solid color. Figure 5(b) shows that for time step $k = 0$ no shrinking is required since the robustness tubes $P_{1,0}$, $P_{2,0}$ are already δ-separate. For time steps $k = 1, 2, 3$, the axis of maximum separation between trajectories is Z; therefore, boxes Π_k are defined to be of height δ with infinite width and length. Boxes Π_k are drawn in gray; midpoints between the trajectories are drawn in yellow. Figure 5(c) depicts the updated δ-separate robustness tubes $\mathbf{P}'_1$ and $\mathbf{P}'_2$.

Theorem 5.3 (Sufficient Condition for δ-separate Tubes). *Zero-slack solution of Equation (6) implies that robustness tubes updated by the RTS procedure are the subsets of the initial robustness tubes and δ-separate; e.g., for robustness tubes $(\mathbf{P}'_1, \mathbf{P}'_2) = \mathbf{RTS}(\mathbf{x}'_1, \mathbf{x}'_2, \mathbf{P}_1, \mathbf{P}_2)$, the following two properties hold:*

$$\mathbf{dist}(\mathbf{P}'_1, \mathbf{P}'_2) \geq \delta \tag{12}$$

$$\mathbf{P}'_j \subseteq \mathbf{P}_j, \ \forall j \in \{1, 2\}. \tag{13}$$

See the proof in the appendix Section A.

5.2 Combination of L2F with RTS

The three following lemmas define important properties of L2F combined with the shrinking process. Proofs can be found in the appendix Section A.

Lemma 5.1. *Let two trajectories $\mathbf{x}'_1$, $\mathbf{x}'_2$ be generated by L2F and let the robustness tubes $\mathbf{P}'_1$, $\mathbf{P}'_2$ be the updated tubes generated by the RTS procedure from initial tubes $\mathbf{P}_1$, $\mathbf{P}_2$ using the trajectories $\mathbf{x}'_1$, $\mathbf{x}'_2$. Then*

$$\mathbf{p}'_j \in \mathbf{P}'_j, \ \forall j \in \{1, 2\}. \tag{14}$$

Lemma 5.1 states that the RTS procedure preserves trajectory belonging to the corresponding updated robustness tube.

Lemma 5.2. *Let two robustness tubes $\mathbf{P}_1$ and $\mathbf{P}_2$ be δ-separate. Then any pair of trajectories within these robustness tubes are non-conflicting, i.e.:*

$$\forall \mathbf{p}_1 \in \mathbf{P}_1, \ \forall \mathbf{p}_2 \in \mathbf{P}_2, \ ||p_{1,k} - p_{2,k}|| \geq \delta, \ \forall k \in \{0, \ldots, H\}. \tag{15}$$

Using Lemma 5.2, we can now prove that every successful application of L2F combined with the shrinking process results in new trajectories that do not violate previously achieved minimum separations between UASs, unless the RTS process results in an empty robustness tube. In other words, it solves the three-UAS issue raised in Example 5.1. We formalize this result in the context of three UASs with the following Lemma:

Lemma 5.3. *Let $\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3$ be preplanned conflicting UAS trajectories, and let $\mathbf{P}_1$, $\mathbf{P}_2$, and $\mathbf{P}_3$ be their corresponding robustness tubes. Without loss of generality, assume that the sequential pairwise application of L2F combined with RTS has been done in the following order:*

$$(\mathbf{x}'_1, \mathbf{x}'_2) = \mathbf{L2F}\ (\mathbf{x}_1, \mathbf{x}_2, \mathbf{P}_1, \mathbf{P}_2), \qquad (\mathbf{P}'_1, \mathbf{P}'_2) = \mathbf{RTS}\ (\mathbf{x}_1, \mathbf{x}_2, \mathbf{P}_1, \mathbf{P}_2) \tag{16}$$

$$(\mathbf{x}''_1, \mathbf{x}'_3) = \mathbf{L2F}\ (\mathbf{x}'_1, \mathbf{x}_3, \mathbf{P}'_1, \mathbf{P}_3), \qquad (\mathbf{P}''_1, \mathbf{P}'_3) = \mathbf{RTS}\ (\mathbf{x}''_1, \mathbf{x}'_3, \mathbf{P}'_1, \mathbf{P}_3) \tag{17}$$

$$(\mathbf{x}''_2, \mathbf{x}''_3) = \mathbf{L2F}\ (\mathbf{x}'_2, \mathbf{x}'_3, \mathbf{P}'_2, \mathbf{P}'_3), \qquad (\mathbf{P}''_2, \mathbf{P}''_3) = \mathbf{RTS}\ (\mathbf{x}''_2, \mathbf{x}''_3, \mathbf{P}'_2, \mathbf{P}'_3). \tag{18}$$

If all three L2F applications gave zero-slack solutions, then position trajectories $\mathbf{p}''_1, \mathbf{p}''_2, \mathbf{p}''_3$ pairwise satisfy mutual separation requirement, e.g.:

$$\left| |p''_{1,k} - p''_{2,k}| \right| \geq \delta, \ \forall k \in \{0, \ldots, H\} \tag{19}$$

$$\left| |p''_{1,k} - p''_{3,k}| \right| \geq \delta, \ \forall k \in \{0, \ldots, H\} \tag{20}$$

$$\left| |p''_{2,k} - p''_{3,k}| \right| \geq \delta, \ \forall k \in \{0, \ldots, H\}, \tag{21}$$

and are within their corresponding robustness tubes:

$$\mathbf{p}''_j \in \mathbf{P}''_j, \ \forall j \in \{1, 2, 3\}. \tag{22}$$

By induction, we can extend Lemma 5.3 to any number of UASs. Therefore, we can conclude that for any N preplanned UAS trajectories, zero-slack solution of LNF is a sufficient condition for CA; e.g., resulting trajectories generated by LNF are non-conflicting and within the robustness tubes of the initial trajectories. Note that this approach can still fail to find a solution, especially as repeated RTS can result in empty robustness tubes.

Theorem 5.4. *For the case of N UASs, when applied at any time step k, LNF (Algorithm 3) terminates after no more than* $\binom{N}{2}$ *applications of pairwise L2F (Algorithm 1).*

This result follows directly from the inductive application of Lemma 5.3. In experimental evaluations (Section 6.3), we see that this worst-case number of L2F applications is not required often in practice.

6 EXPERIMENTAL EVALUATION OF L2F AND LNF

In this section, we show the performance of our proposed methods via extensive simulations, as well as an implementation for actual quad-rotor robots. We compare L2F and L2F with repair (L2F+Rep) with the MILP formulation of Section 3 and two other baseline approaches. Through multiple case studies, we show how LNF extends the L2F framework to work for scenarios with more than two UASs.

6.1 Experimental Setup

Computation Platform. All the simulations were performed on a computer with an AMD Ryzen 7 2700 eight-core processor and 16GB RAM, running Python 3.6 on Ubuntu 18.04.

Generating Training Data. We have generated the dataset of 14K trajectories for training with collisions between UASs using the trajectory generator in [25]. The look-ahead horizon was set to $T = 4$s and $dt = 0.1$s. Thus, each trajectory consists of $H + 1 = 41$ time steps. The initial and final waypoints were sampled uniformly at random from two 3D cubes close to the fixed collision point; initial velocities were set to zero.

Implementation Details for the Learning-Based Conflict Resolution. The MILP to generate training data for the supervised learning of the CR scheme was implemented in MATLAB using Yalmip [20] with MOSEK v8 as the solver. The learning-based CR scheme was trained for $\rho = 0.055$ and minimum separation $\delta = 0.1$m, which is close to the lower bound in Assumption 2. This was implemented in Python 3.6 with Tensorflow 1.14 and Keras API and Casadi with qpOASES as the solver. For training the LSTM models (with different architectures) for CR, the number of training epochs was set to 2K with a batch size of 2K. Each network was trained to minimize categorical cross-entropy loss using the Adam optimizer [15] with a training rate of $\alpha = 0.001$ and moment exponential decay rates of $\beta_1 = 0.9$ and $\beta_2 = 0.999$. The model with three LSTM layers with 128 neurons each (see Figure 3) was chosen as the default learning-based CR model and is used for the pairwise CA approach of both L2F and LNF.

Implementation Details for the CA-MPC. For the online implementation of our scheme, we implement CA-MPC using CVXgen and report the computation times for this implementation. We then import CA-MPC in Python, interface it with the CR scheme, and run all simulations in Python.

6.2 Experimental Evaluation of L2F

We evaluate the performance of L2F with 10K test trajectories (for pairwise CA) generated using the same distribution of start and end positions as was used for training. Figure 6 shows an example of two UAS trajectories before and after L2F. Successful avoidance of the collision at the midway point on the trajectories can easily be seen on the playback of the scenario available at

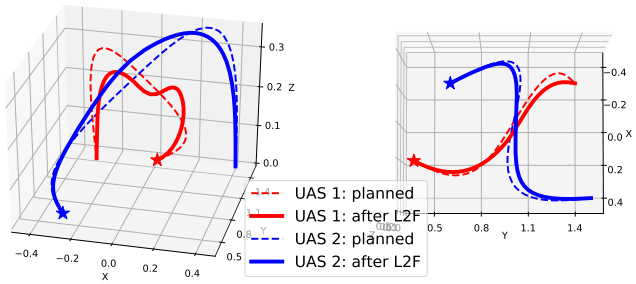


Fig. 6. Trajectories for two UASs from different angles. The dashed (planned) trajectories have a collision at the halfway point. The solid ones, generated through the L2F method, avoid the collision while remaining within the robustness tube of the original trajectories. Initial UAS positions marked as stars. Playback of the scenario is at https://tinyurl.com/l2f-exmpl.

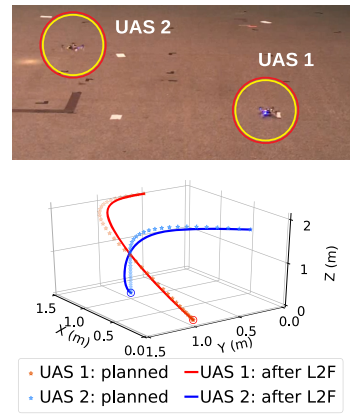


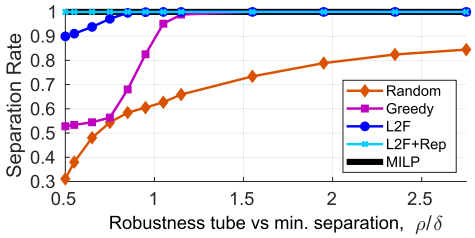
Fig. 7. Trajectories for two Crazyflie quad-rotors before (dotted) and after (solid) L2F. Videos of this are at https://tinyurl.com/exp-cf2.

https://tinyurl.com/l2f-exmpl. To demonstrate the feasibility of the deconflicted trajectories, we also ran experiments using two Crazyflie quad-rotor robots as shown in Figure 7. Videos of the actual flights and additional simulations can be found at https://tinyurl.com/exp-cf2.

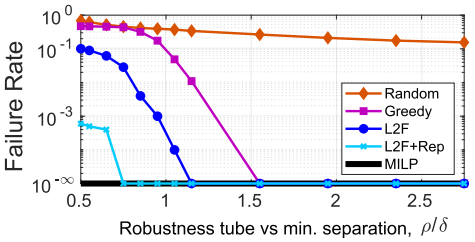
6.2.1 Results and Comparison to Other Methods. We analyzed three other methods alongside the proposed learning-based approach for L2F.

(1) A **random** decision approach that outputs a sequence sampled from the discrete uniform distribution.
(2) A **greedy** approach that selects the discrete decisions that correspond to the direction of the most separation between the two UASs at each time step. For more details see [32].
(3) An **L2F with Repairing** approach following Section 4.3.
(4) A centralized **MILP** solution that picks decisions corresponding to binary decision variables in Equation (5).

For the evaluation, we measured and compared the **separation rate** and the **computation time** for all the methods over the same 10K test trajectories. *Separation rate* defines the fraction of the conflicting trajectories for which UAS managed to achieve minimum separation after a CA approach. Figure 8 shows the impact of the ρ/δ ratio on separation rate. Higher ρ/δ implies wider robustness tubes for the UAS to maneuver within, which should make the CA task easier as is seen in the figure. The centralized MILP has a separation rate of 1 for each case here but is unsuitable for an online implementation with its computation time being over a minute (see Table 1), and we exclude it from the comparisons in the text that follows. In the case of $\rho/\delta = 0.5$, where the robustness tubes are just wide enough to fit two UASs (see Assumption 2), we see that L2F with repairing (L2F+Rep) significantly outperforms the methods. This worst-case performance of L2F with repairing is 0.999, which is significantly better than the other approaches including the original L2F. As the ratio grows, the performance of all methods improves, with L2F+Rep still



(a) *Separation rate* defines the fraction of initially conflicting trajectories for which UAS managed to achieve minimum separation.



(b) *Failure rate* (1-*Separation rate*) defines the fraction of initially conflicting trajectories for which UA could not achieve minimum separation.

Fig. 8. Model sensitivity analysis with respect to variations of fraction ρ/δ, which connects the minimum allowable robustness tube radius ρ to the minimum allowable separation between two UASs δ; see Assumption 2. A higher ρ/δ implies there is more room within the robustness tubes to maneuver for CA.

Table 1. Separation Rates and Computation Times (Mean and Standard Deviation) Comparison of Different CA Schemes

		CA Scheme				
		Random	**Greedy**	**L2F**	**L2F+Rep**	**MILP**
Separation rate	$\boldsymbol{\rho/\delta = 0.5}$	0.311	0.528	0.899	0.999	1
	$\boldsymbol{\rho/\delta = 0.95}$	0.605	0.825	0.999	1	1
	$\boldsymbol{\rho/\delta = 1.15}$	0.659	0.989	1	1	1
Comput. time (ms) (mean ± std)	$\boldsymbol{\rho/\delta = 0.5}$	7.9 ± 0.01	9.7 ± 0.6	9.1 ± 1.3	9.7 ± 3.6	$(98.9 \pm 44.9) \cdot 10^3$
	$\boldsymbol{\rho/\delta = 0.95}$	7.5 ± 0.01	9.3 ± 0.5	8.7 ± 0.5	8.7 ± 0.5	$(82.5 \pm 36.3) \cdot 10^3$
	$\boldsymbol{\rho/\delta = 1.15}$	6.3 ± 1.9	7.1 ± 2.	8.6 ± 0.5	8.7 ± 0.4	$(33.1 \pm 34.9) \cdot 10^3$

Separation rate is the fraction of conflicting trajectories for which the separation requirement (Equation (2a)) is satisfied after CA. *Computation time* estimates the overall time demanded by the CA scheme. MILP reports the time spent on solving Equation (5). Other CA schemes report time needed for CR and CA-MPC together. L2F with repairing includes repairing time as well.

outperforming the others and quickly reaching a separation rate of 1. For $\rho/\delta \geq 1.15$, L2F no longer requires any repair and also has a separation rate of 1.

Table 1 shows the separation rates for three different ρ/δ values as well as the computation times (mean and standard deviation) for each CA algorithm. L2F and L2F+Rep have an average computation time of less than 10ms, making them suited for an online implementation even at our chosen control sampling rate of 10Hz. For all CA schemes excluding MILP, the smaller the ρ/δ ratio, the more UAS 1 alone is unsuccessful at collision avoidance MPC (Equation (7)), and UAS 2 must also solve its CA-MPC (Equation (8)) and deviate from its preplanned trajectory. Therefore, computation time is higher for smaller ρ/δ ratios and lower for higher ρ/δ values. A similar trend is observed for the MILP, even though it jointly solves for both UASs, showing that it is indeed harder to find a solution when the ρ/δ ratio is small.

6.3 Experimental Evaluation of LNF

Next, we carry out simulations to evaluate the performance of LNF, especially in terms of scalability to cases with more than two UASs, and analyze its performance in a wide variety of settings.

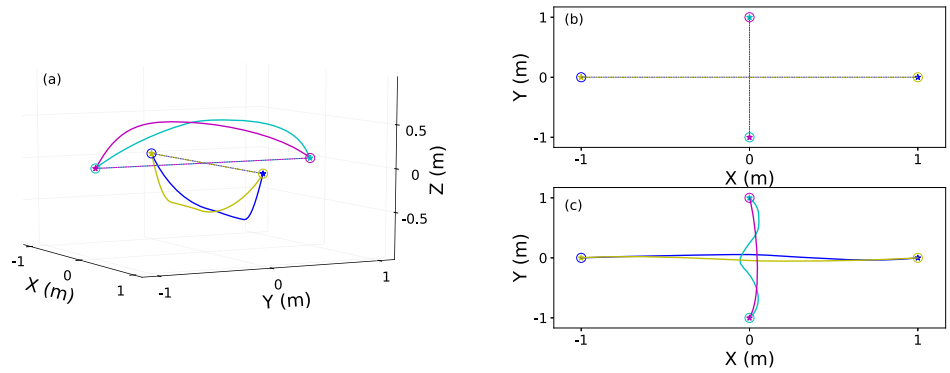


Fig. 9. Four-UAS position swap. (a): 3D representation of the scenario. (b) and (c): 2D projections of the scenario onto the horizontal plane XoY before and after collision avoidance. Initial colliding trajectories are depicted in dashed lines in (a) and (b). Collision is detected at point $(0, 0, 0)$; it involves all four UASs and happens simultaneously across the agents. The updated non-colliding trajectories generated by LNF are depicted in solid color in (a) and (c). Initial positions of UAS are marked by "O" and final positions by "★."

6.3.1 Case Study 1: Four-UAS Position Swap. We recreate the following experiment from [3]. Here, two pairs of UASs must maneuver to swap their positions; i.e., the end point of each UAS is the same as the starting position for another UAS. See the 3D representation of the scenario in Figure 9(a). Each UAS start set is assumed to be a singular point fixed at

$$Goal_1 = (1, 0, 0),\ Goal_2 = (0, 1, 0),\ Goal_3 = (-1, 0, 0),\ Goal_4 = (0, -1, 0), \tag{23}$$

and goal states are antipodal to the start states:

$$Start_j = -Goal_j,\ \forall j \in \{1, 2, 3, 4\}. \tag{24}$$

All four UASs must reach desired goal states within 4 seconds while avoiding each other. With a pairwise separation requirement of at least $\delta = 0.1$ meters, the overall mission specification is

$$\varphi_{mission} = \bigwedge_{j=1}^{4} \Diamond_{[0,4]}(\mathbf{p}_j \in Goal_j) \ \wedge \bigwedge_{j \neq j'} \Box_{[0,4]} ||\mathbf{p}_j - \mathbf{p}_{j'}|| \geq 0.1. \tag{25}$$

Following Section 2.2, initial preplanning is done by ignoring the mutual separation requirement in Equation (25) and generating the trajectory for each UAS $j = \{1, 2, 3, 4\}$ independently with respect to its individual STL specification:

$$\varphi_j = \Diamond_{[0,4]}(\mathbf{p}_j \in Goal_j). \tag{26}$$

Obtained preplanned trajectories contain a joint collision that happens simultaneously (at $t = 2$s; see Figure 10) across all four UASs and located at point $(0, 0, 0)$; see Figure 9(b). For LNF experimental evaluation, the robustness value was fixed at $\rho = 0.055$ and the UAS priorities were set in increasing order; e.g., the UAS with a lower index has the lower priority: $1 < 2 < 3 < 4$.

Simulation Results. The non-colliding trajectories generated by LNF are depicted in Figure 9(c). Playback of the scenario can be found at https://tinyurl.com/swap-pos.

It is observed that the opposite UAS pairs chose to change attitude and pass over each other; see Figure 9(a). Within these opposite pairs, UASs chose to have horizontal deviations to avoid collision; see Figure 9(c). The LNF algorithm performed $\binom{4}{2} = 6$ pairwise applications of L2F (see Theorem 5.4). Such a high number of applications is expected due to the complicated simultaneous nature of the detected collision across the initially preplanned trajectories. No CR repairing

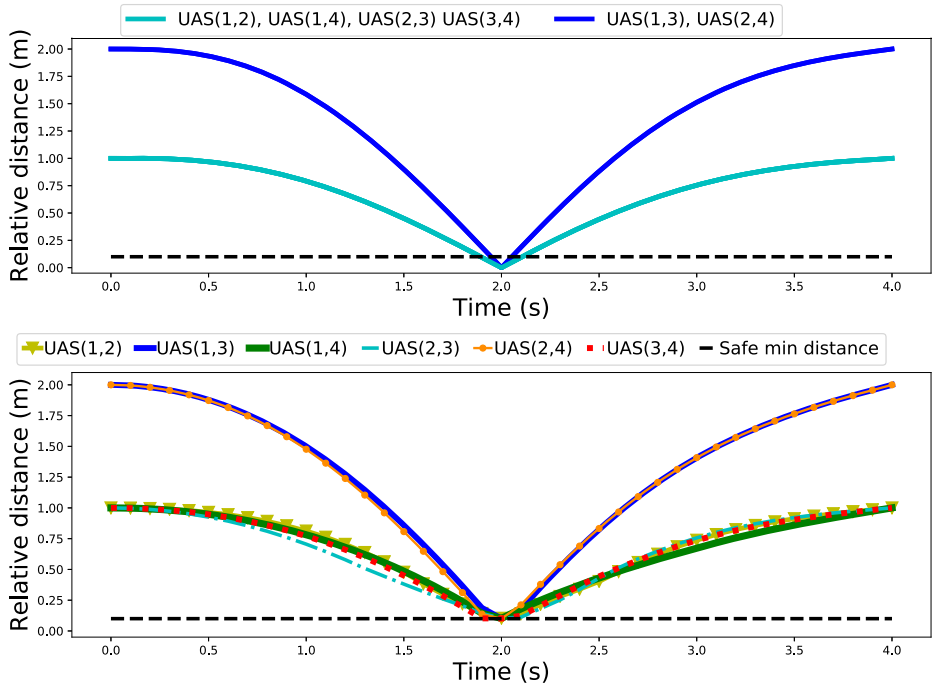


Fig. 10. Four-UAS position swap: Relative distances before (top) and after (bottom) the collision avoidance algorithm. Initial simultaneous collisions across all four UASs are successfully resolved by LNF. Note that the symmetry in the initial positions and trajectories results in multiple UAS pairs with the same relative distances for the time horizon of interest before collision avoidance (top).

was required to successfully produce noncolliding trajectories by the LNF algorithm. It took LNF 37.8ms to perform CA. Figure 10 represents relative distances between UAS pairs before and after collision avoidance. Figure 10 shows that none of the UASs cross the safe minimum separation threshold of 0.1m after LNF; e.g., joint collision has been successfully resolved by LNF.

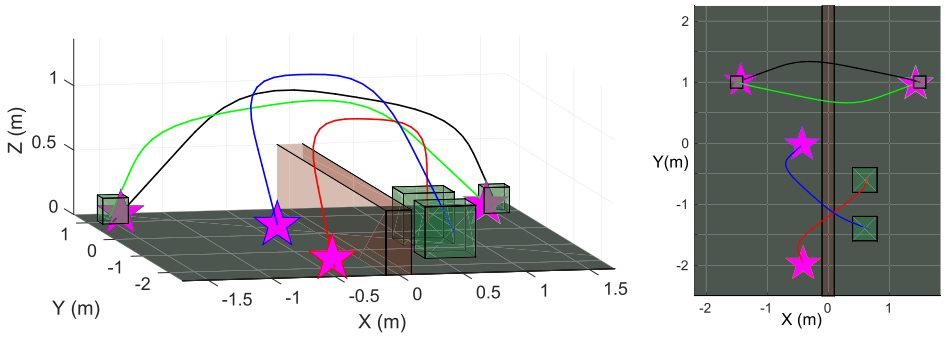
6.3.2 Case Study 2: Four-UAS Reach-Avoid Mission. Figure 11 depicts a multi-UAS case study with a reach-avoid mission. The scenario consists of four UASs that must reach desired goal states within 4 seconds while avoiding the wall obstacle and each other. Each UAS $j \in \{1, \ldots, 4\}$ specification can be defined as:

$$\varphi_j = \Diamond_{[0,4]}(\mathbf{p}_j \in Goal_j) \ \wedge \ \Box_{[0,4]} \neg (\mathbf{p}_j \in Wall). \tag{27}$$

A pairwise separations requirement of $\delta = 0.1$ meters is enforced for all UASs; therefore, the overall mission specification is

$$\varphi_{\text{mission}} = \bigwedge_{j=1}^{4} \varphi_j \ \wedge \ \bigwedge_{j \neq j'} \Box_{[0,4]} ||\mathbf{p}_j - \mathbf{p}_{j'}|| \geq 0.1. \tag{28}$$

First, we solved the planning problem for all four UASs in a centralized manner following the approach from [29]. Next, we solved the planning problem for each UAS j and its specification φ_j independently, with calling LNF on the fly, after planning is complete. This way, independent planning with the online collision avoidance scheme guarantees the satisfaction of the overall mission specification (Equation (28)).



(a) 3D representation of the scenario (b) 2D projection onto XoY

Fig. 11. Reach-avoid mission. Non-colliding trajectories for four UASs generated by LNF. All UASs reach their goal sets (green boxes) within 4 seconds, do not crash into the vertical wall (in red), and satisfy the pairwise separation requirement of 0.1m. Initial UAS positions marked by magenta "⋆." Simulations are available at https://tinyurl.com/reach-av.

Table 2. Reach-Avoid Mission

	Centralized Planning [29]	Decentralized Planning with CA	
		Independent Planning	CA with LNF
Comput. time (mean ± std)(ms)	345.8 ± 87.2	138.6 ± 62.4	9.97 ± 0.4

Computation time (mean and standard deviation) comparison between centralized planning following [29] and decentralized planning (independent planning with LNF) over 100 runs of the scenario.

Simulation Results. We have simulated the scenario for 100 different initial conditions. Computation time results are presented in Table 2. The average computation time to generate trajectories in a centralized manner was 0.35 seconds. The average time per UAS when planning independently (and in parallel) was 0.1 seconds. These results demonstrate a speedup of 3.5× for the individual UAS planning versus centralized [29]. Scenario simulations are available at https://tinyurl.com/reach-av.

6.3.3 Case Study 3: UAS Operations in High-Density Airspace. To verify the scalability of LNF, we perform evaluation of the scenario with high-density UAS operations. The case study consists of multiple UASs flying within the restricted area of 1m^3 while avoiding a no-fly zone of $(0.2)^3 = 0.08m^3$ in the center; see Figure 12. Such a scenario represents a hypothetical constrained and highly populated airspace with heterogeneous UAS missions such as package delivery or aerial surveillance.

Each UAS's j start position $Start_j$ and goal set $Goal_j$ are chosen at (uniform) random on the opposite random faces of the unit cube. Goal state should be reached within a 4-second time interval and the no-fly zone must be avoided during this time interval. Same as in the previous case studies, we first solve the planning problem for each UAS j separately following the trajectory generation approach from [29]. The STL specification for UAS j is captured as follows:

$$\varphi_j = \Diamond_{[0,4]}(\mathbf{p}_j \in Goal_j) \ \wedge \ \Box_{[0,4]}\neg(\mathbf{p}_j \in NoFly). \tag{29}$$

After planning is complete and trajectories $\mathbf{p}_j$ are generated, we call LNF on the fly to satisfy the overall mission specification $\varphi_{\text{mission}} = \bigwedge_{j=1}^{N} \varphi_j \ \wedge \ \varphi_{\text{separation}}$, where N is a number of UASs participating in the scenario and φ_{mission} is the requirement of the minimum allowed pairwise

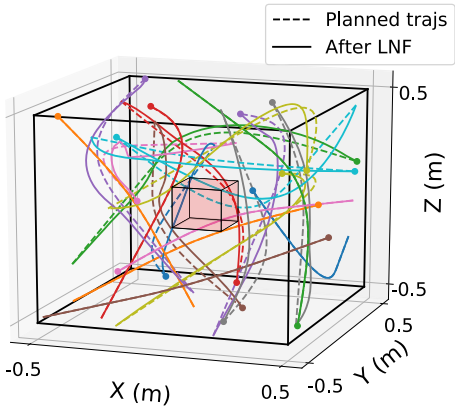


Fig. 12. 3D representation of the unit cube scenario with 20 UASs. All UASs must reach their goal sets within 4 seconds, avoid the no-fly zone, and satisfy pairwise separation requirement of 0.1m. Initially planned trajectories (dashed lines) had five violations of the mutual separation requirement. LNF successfully resolved all detected violations and led to non-colliding trajectories (solid lines). Simulations are available at https://tinyurl.com/unit-cube.

separation of 0.1m between the UAS:

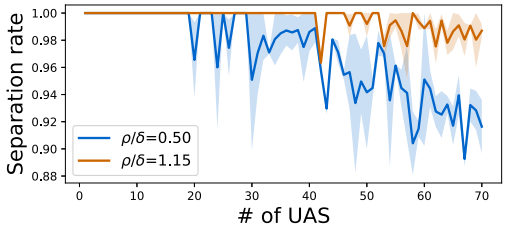
$$\varphi_{\text{separation}} = \bigwedge_{j,j':\ j\neq j'} \square_{[0,4]} ||\mathbf{p}_j - \mathbf{p}_{j'}|| \geq 0.1. \tag{30}$$

We increase the density of the scenario by increasing the number of UASs while keeping the space volume at 1m^3.

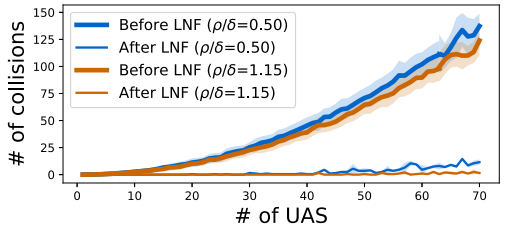
Simulation Results. We ran 100 simulations for various numbers of UASs, each with randomized start and goal positions. Trajectory preplanning was done independently for all UASs, and LNF is tasked with CA. For evaluation, we measure the overall number of minimum separation requirement violations before and after LNF for two different settings of the fraction ρ/δ: narrow robustness tube, $\rho/\delta = 0.5$, and wider tube, $\rho/\delta = 1.15$; see Figure 13. With increasing number of UASs, the number of collisions between initially preplanned trajectories increases (before LNF) and the number of not collisions by LNF, while small, increases as well (Figure 13(b)). The corresponding decay in separation rate over pairs of collisions resolved is faster for the case of $\rho/\delta = 0.5$, which is expected due to less room to maneuver. The separation rate is higher when the ρ/δ ratio is higher; see Figure 13(a). We performed simulations for up to 70 UASs. The average separation rate for 70 UASs is 0.915 for $\rho/\delta = 0.5$ and 0.987 for $\rho/\delta = 1.15$. The results show that LNF can still succeed in scenarios with a high UAS density. Videos of the simulations are available at https://tinyurl.com/unit-cube.

6.3.4 Comparison to MILP-Based Replanning. LNF requires the new trajectories after CA to be be within the robustness tubes of preplanned trajectories to still satisfy other high-level requirements (Problem 1). While this might be restrictive, we show that online replanning is usually not an option in these multi-UAS scenarios. An MILP-based planner, similar in essence to [31], was implemented and treated as a baseline to compare against LNF through evaluations on the scenario of Section 6.3.3. Unlike the decentralized LNF, such MILP planner baseline is centralized as it plans for all the UASs in a single optimization to avoid the *no-fly* zone, reach their destinations, and also avoid each other.

We ran 100 simulations for various numbers of UASs, with each iteration having randomized start and goal positions. Simulations are available at https://tinyurl.com/re-milp. The computation times are presented in Table 3. As the number of UASs increases, it is clear the online replanning is intractable. For example, the baseline takes on average 8.8 seconds to produce trajectories for 20 UASs, in contrast with 73.1 milliseconds for LNF to perform CA. For 50 UASs and higher the MILP baseline solver could not return a single feasible solution, while LNF could. LNF outperforms the



(a) *Separation rate* defines the fraction between the number of initial violations of the minimum separation and the number of resolved violations by LNF.



(b) Number of minimum separation violations before and after LNF, averaged over 100 simulations.

Fig. 13. Unit cube scenario. Model performance analysis with respect to variations in the number of UASs for two different settings of ρ/δ. A higher ρ/δ implies there is more room within the robustness tubes to maneuver for CA. Performance is measured in terms of separation rate (a) and the overall number of minimum separation requirement violations before and after LNF (b). We plot the mean and standard deviation over 100 iterations.

Table 3. *Computation Times* (Mean and Standard Deviation) Demanded by the Replanning Scheme (MILP-Based Replanning or CA with LNF) Averaged over 100 Random Runs

	Replanning Scheme	$N = 10$	$N = 20$	$N = 30$	$N = 40$	$N = 50$
Comp. times	MILP-based planner	0.6 ± 0.1s	8.8 ± 9.6s	175.5 ± 149.9s	1740. ± 129.3s	Timeout
(mean ± std)	CA with LNF	15.2 ± 5.1ms	73.1 ± 23.5ms	117.3 ± 45.6ms	198.7 ± 73.6ms	211.1 ± 82.3ms

Time taken by the MILP-based replanner to encode the problem is not included in the overall computation time. "Timeout" stands for a timeout after 35 minutes.

MILP-based replanning baseline since it can perform CA with small computation times, even for a high number of UASs.

7 CONCLUSIONS

Summary. We presented L2F, an online, decentralized, and mission-aware scheme for pairwise UAS collision avoidance. Through LNF, we extended it to work for cases where more than two UASs are on collision paths, via a systematic pairwise application of L2F and with a set-shrinking approach to avoid live-lock like situations. These frameworks combine learning-based decision-making and decentralized linear optimization-based MPC to perform CA, and we also developed a fast heuristic to *repair* the decisions made by the learning-based component based on the feasibility of the optimizations. Through extensive simulation, we showed that our approach has a computation time of the order of milliseconds and can perform CA for a wide variety of cases with a high success rate even when the UAS density in the airspace is high.

Limitations and Future Work. While our approach works very well in practice, it is not *complete*, i.e., does not guarantee a solution when one exists, as seen in simulation results for L2F. This drawback requires a careful analysis for obtaining the sets of initial conditions over the conflicting UASs such that our method is guaranteed to work. In future work, we aim to leverage tools from formal methods, like falsification, to get a reasonable estimate of the conditions in which our method is guaranteed to work. We will also explore improved heuristics for the set-shrinking in LNF, as well as the CR decision repairing procedure.

APPENDICES

A ROBUSTNESS TUBE SHRINKING

Definition 3. The distance between two sets A and B is defined as

$$\mathbf{dist}(A, B) = \inf \{||a - b||_\infty \mid a \in A,\ b \in B\}. \tag{31}$$

Definition 4. Two robustness tubes $\mathbf{P}_1$ and $\mathbf{P}_2$ are said to be δ-separate from each other if at every time step k the distance between them is at least δ, i.e.:

$$\mathbf{dist}(P_{1,k}, P_{2,k}) \geq \delta \ \forall k = 0, \dots, H. \tag{32}$$

For brevity we use $\mathbf{dist}(\mathbf{P}_1, \mathbf{P}_2) \geq \delta$ for denoting being δ-separate across all time indices $k = 0, \dots, H$.

Proof of Theorem 5.3. By construction of RTS; see Algorithm 4. If initial tubes are δ-separate, then no shrinking is required and therefore, both Properties (13) and (12) are satisfied. If the initial tubes are not δ-separate, Property (13) comes from the fact that for any time step k, $P'_{j,k} = P_{j,k} \setminus \Pi_k$ for UAS $j = 1, 2$. Property (12) is a consequence of the zero-slack solution and Theorem 4.1, which states that resulting trajectories are non-conflicting, $||p'_{1,k} - p'_{2,k}|| \geq \delta$, $\forall k \in \{0, \dots, H\}$; therefore, $msep \geq \delta$. Following Algorithm 4, for any time step k box's Π_k smallest edge is $\min(msep, \delta) = \delta$, and since for both UAS $j = 1, 2$ the tube update is defined as $P'_{j,k} = P_{j,k} \setminus \Pi_k$, the shrinked tubes $P'_{j,k}$ are δ-separate. □

Proof of Lemma 5.1. From the CA-MPC definition (Equation (6)) it follows that $\mathbf{p}'_j \in \mathbf{P}_j$, $\forall j \in \{1, 2\}$. The updated tubes are defined as $\mathbf{P}'_j = \mathbf{P}_j \setminus \boldsymbol{\Pi}$; see Algorithm 4. By the definition of 3D cube $\boldsymbol{\Pi}$, for any time step k, $p'_{j,k} \notin \Pi_k$; therefore, $\mathbf{p}'_j \in \mathbf{P}'_j,\ \forall j \in \{1, 2\}$. □

Proof of Lemma 5.2. Following Definition 4, tubes are δ-separate if $\mathbf{dist}(P_{1,k}, P_{2,k}) \geq \delta,\ \forall k \in \{0, \dots, H\}$. Therefore, due to Equation (31), the following holds:

$$\inf \{||p_{1,k} - p_{2,k}|| \mid p_{1,k} \in P_{1,k},\ p_{2,k} \in P_{2,k}\} \geq \delta. \tag{33}$$

By the definition of the infimum operator, $\forall p_{1,k} \in P_{1,k}, \forall p_{2,k} \in P_{2,k}$:

$$||p_{1,k} - p_{2,k}|| \geq \inf \{||p_{1,k} - p_{2,k}|| \mid p_{1,k} \in P_{1,k},\ p_{2,k} \in P_{2,k}\} \geq \delta, \tag{34}$$

which completes the proof. □

Proof of Lemma 5.3.
(1) Property (21) directly follows from Theorem 4.1.
(2) Due to Theorem 5.3, RTS application (17) leads to tubes $\mathbf{P}''_1$ and $\mathbf{P}'_3$ being δ-separate. RTS (18) leads to $\mathbf{P}''_3 \subseteq \mathbf{P}'_3$. Therefore, $\mathbf{P}''_1$ and $\mathbf{P}''_3$ are δ-separate and following Lemma 5.2, property (20) holds.
(3) Analogously, due to Theorem 5.3, RTS application (16) leads to tubes $\mathbf{P}'_1$ and $\mathbf{P}'_2$ being δ-separate. RTS (17) leads to $\mathbf{P}''_1 \subseteq \mathbf{P}'_1$ and RTS (18) leads to $\mathbf{P}''_2 \subseteq \mathbf{P}'_2$. Therefore, $\mathbf{P}''_1$ and $\mathbf{P}''_2$ are δ-separate and following Lemma 5.2, Property (19) holds.
(4) Tube-belonging Property (22) follows directly from Lemma 5.1.

□

B LINKS TO THE VIDEOS

Table 4 has the links for the visualizations of all simulations and experiments performed in this work.

Table 4. Links for the Videos for Simulations and Experiments

Scenario	Section	Platform	# of UAS	Link
L2F test	Section 6.2	Sim.	2	https://tinyurl.com/l2f-exmpl
Crazyflie validation	Section 6.2	CF 2.0	2	https://tinyurl.com/exp-cf2
Four-UAS position swap	Section 6.3.1	Sim.	4	https://tinyurl.com/swap-pos
Four-UAS reach-avoid mission	Section 6.3.2	Sim.	4	https://tinyurl.com/reach-av
High-density unit cube	Section 6.3.3	Sim.	10, 20, 40	https://tinyurl.com/unit-cube
MILP replanning	Sec 6.3.4	MATLAB	20	https://tinyurl.com/re-milp

"Sim." stands for Python simulations, "CF2.0" for experiments on the Crazyflies.

REFERENCES

[1] Federal Aviation Administration. 2018. Concept of Operations: Unmanned Aircraft System (UAS) Traffic Management (UTM). https://utm.arc.nasa.gov/docs/2018-UTM-ConOps-v1.0.pdf.

[2] Derya Aksaray, Austin Jones, Zhaodan Kong, Mac Schwager, and Calin Belta. 2016. Q-Learning for robust satisfaction of signal temporal logic specifications. In *IEEE Conference on Decision and Control*.

[3] Javier Alonso-Mora, Tobias Naegeli, Roland Siegwart, and Paul Beardsley. 2015. Collision avoidance for aerial vehicles in multi-agent scenarios. *Autonomous Robots* 39, 1 (2015), 101–121.

[4] Anjan Chakrabarty, Corey Ippolito, Joshua Baculi, Kalmanje Krishnakumar, and Sebastian Hening. 2019. Vehicle to Vehicle (V2V) communication for Collision avoidance for Multi-copters flying in UTM–TCL4. https://doi.org/10.2514/6.2019-0690

[5] Mohammed Dahleh, Munther A. Dahleh, and George Verghese. 2004. Lectures on dynamic systems and control. *MIT Lecture Notes* 4, 100 (2004), 1–100.

[6] Jonathan A. DeCastro, Javier Alonso-Mora, Vasumathi Raman, and Hadas Kress-Gazit. 2017. Collision-free reactive mission and motion planning for multi-robot systems. In *Springer Proceedings in Advanced Robotics*.

[7] Ankush Desai, Indranil Saha, Yang Jianqiao, Shaz Qadeer, and Sanjit A. Seshia. 2017. DRONA: A framework for safe distributed mobile robotics. In *ACM/IEEE International Conference on Cyber-Physical Systems*.

[8] Francisco Fabra, Willian Zamora, Julio Sangüesa, Carlos T. Calafate, Juan-Carlos Cano, and Pietro Manzoni. 2019. A distributed approach for collision avoidance between multirotor UAVs following planned missions. *Sensors* 19, 10 (2019), 2404.

[9] G. Fainekos and G. Pappas. 2009. Robustness of temporal logic specifications for continuous-time signals. *Theoretical Computer Science* 410, 42 (2009), 4262–4291.

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

[11] Felix A. Gers, Nicol N. Schraudolph, and Jürgen Schmidhuber. 2002. Learning precise timing with LSTM recurrent networks. *Journal of Machine Learning Research* 3, (2002), 115–143.

[12] Davis L. Hackenberg. 2018. ARMD Urban Air Mobility Grand Challenge: UAM Grand Challenge Scenarios. https://evtol.news/__media/PDFs/eVTOL-NASA-Revised_UAM_Grand_Challenge_Scenarios.pdf. (2018).

[13] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation* 9, 8 (1997), 1735–1780.

[14] S. Karaman and E. Frazzoli. 2011. Linear temporal logic vehicle routing with applications to multi-UAV mission planning. *International Journal of Robust and Nonlinear Control* 21, 12 (2011), 1372–1395.

[15] Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[16] M. Kloetzer and C. Belta. 2006. Hierarchical abstractions for robotic swarms. In *Proc. of 2006 IEEE International Conference on Robotics and Automation*. 952–957. https://doi.org/10.1109/ROBOT.2006.1641832

[17] M. Kloetzer and C. Belta. 2008. A fully automated framework for control of linear systems from temporal logic specifications. *IEEE Transactions on Automatic Control* 53, 1 (Feb. 2008), 287–297. https://doi.org/10.1109/TAC.2007.914952

[18] Mykel J. Kochenderfer, Jessica E. Holland, and James P. Chryssanthacopoulos. 2012. *Next-Generation Airborne Collision Avoidance System*. Technical Report. MIT-Lincoln Laboratory, Lexington, KY.

[19] M. Z. Li, W. R. Tam, S. M. Prakash, J. F. Kennedy, M. S. Ryerson, D. Lee, and Y. V. Pant. 2018. Design and implementation of a centralized system for autonomous unmanned aerial vehicle trajectory conflict resolution. In *Proceedings of IEEE National Aerospace and Electronics Conference*.

[20] Johan Lofberg. 2004. YALMIP: A toolbox for modeling and optimization in MATLAB. In *2004 IEEE International Conference on Robotics and Automation (IEEE Cat. No. 04CH37508)*. IEEE, 284–289.

[21] Teppo Luukkonen. 2011. Modelling and control of quadcopter. *Independent Research Project in Applied Mathematics, Espoo* 22 (2011).
[22] Xiaobai Ma, Ziyuan Jiao, and Zhenkai Wang. 2016. Decentralized prioritized motion planning for multiple autonomous UAVs in 3D polygonal obstacle environments. In *International Conference on Unmanned Aircraft Systems*.
[23] Oded Maler and Dejan Nickovic. 2004. *Monitoring Temporal Properties of Continuous Signals*. Springer, Berlin.
[24] G. Manfredi and Y. Jestin. 2016. An introduction to ACAS Xu and the challenges ahead. In *2016 IEEE/AIAA 35th Digital Avionics Systems Conference (DASC'16)*. 1–9. https://doi.org/10.1109/DASC.2016.7778055
[25] Mark W. Mueller, Markus Hehn, and Raffaello D'Andrea. 2015. A computationally efficient motion primitive for quadrocopter trajectory generation. *IEEE Transactions on Robotics* 31, 6 (2015), 1294–1310.
[26] NASA. 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.
[27] Yash Vardhan Pant, Houssam Abbas, and Rahul Mangharam. 2017. Smooth operator: Control using the smooth robustness of temporal logic. In *2017 IEEE Conference on Control Technology and Applications*. IEEE, 1235–1240.
[28] Yash Vardhan Pant, Houssam Abbas, Kartik Mohta, Truong X. Nghiem, Joseph Devietti, and Rahul Mangharam. 2015. Co-design of anytime computation and robust control. In *2015 IEEE Real-Time Systems Symposium*. IEEE, 43–52.
[29] Yash Vardhan Pant, Houssam Abbas, Rhudii A. Quaye, and Rahul Mangharam. 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, 186–197.
[30] V. Raman, A. Donze, M. Maasoumy, R. M. Murray, A. Sangiovanni-Vincentelli, and S. A. Seshia. 2014. Model predictive control with signal temporal logic specifications. In *53rd IEEE Conference on Decision and Control*. 81–87. https://doi.org/10.1109/CDC.2014.7039363
[31] Vasumathi Raman, Alexandre Donzé, Mehdi Maasoumy, Richard M. Murray, Alberto Sangiovanni-Vincentelli, and Sanjit A. Seshia. 2014. Model predictive control with signal temporal logic specifications. In *53rd IEEE Conference on Decision and Control*. IEEE, 81–87.
[32] Alena Rodionova, Yash Vardhan Pant, Kuk Jang, Houssam Abbas, and Rahul Mangharam. 2020. Learning to Fly - Learning-based Collision Avoidance for Scalable Urban Air Mobility. In *Proceedings of the IEEE International Conference on Intelligent Transportation Systems*. http://arxiv.org/abs/2006.13267.
[33] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. 2015. ImageNet large scale visual recognition challenge. *International Journal of Computer Vision (IJCV)* 115, 3 (2015), 211–252. https://doi.org/10.1007/s11263-015-0816-y
[34] Indranil Saha, Ramaithitima Rattanachai, Vijay Kumar, George J. Pappas, and Sanjit A. Seshia. 2014. Automated composition of motion primitives for multi-robot systems from safe LTL specifications. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*.
[35] S. Saha and A. Agung Julius. 2016. An MILP approach for real-time optimal controller synthesis with Metric Temporal Logic specifications. In *Proceedings of the 2016 American Control Conference (ACC'16)*.

Received August 2020; revised December 2020; accepted January 2021