Learning-to-Fly: Learning-based Collision Avoidance for Scalable Urban Air Mobility

Alëna Rodionova[1,*], Yash Vardhan Pant[2,*], Kuk Jang[1], Houssam Abbas[3], Rahul Mangharam[1]

***Abstract*— With increasing urban population, there is global interest in Urban Air Mobility (UAM), where hundreds of autonomous Unmanned Aircraft Systems (UAS) execute missions in the airspace above cities. Unlike traditional human-in-the-loop air traffic management, UAM requires decentralized autonomous approaches that scale for an order of magnitude higher aircraft densities and are applicable to urban settings. We present Learning-to-Fly (L2F), a decentralized on-demand airborne collision avoidance framework for multiple UAS that allows them to independently plan and safely execute missions with spatial, temporal and reactive objectives expressed using Signal Temporal Logic. We formulate the problem of predictively avoiding collisions between two UAS without violating mission objectives as a Mixed Integer Linear Program (MILP). This however is intractable to solve online. Instead, we develop L2F, a two-stage collision avoidance method that consists of: 1) a learning-based decision-making scheme and 2) a distributed, linear programming-based UAS control algorithm. Through extensive simulations, we show the real-time applicability of our method which is $\approx 6000\times$ faster than the MILP approach and can resolve 100% of collisions when there is ample room to maneuver, and shows graceful degradation in performance otherwise. We also compare L2F to two other methods and demonstrate an implementation on quad-rotor robots.**

I. Introduction

The development of safe and reliable UAS Traffic Management (UTM) is necessary to enable Urban Air Mobility (UAM) [1]. The two fundamental issues here are: a) mission planning for UAS fleets with guarantees on safety and performance, and b) real-time airborne collision avoidance (CA) methods so UAS run by different operators can share the airspace without a priori approval of all flight plans. Tackling the planning and inter-UAS collision avoidance jointly yields a computationally intractable problem as the number of UAS in the airspace increase [2], [3]. So we separate these two aspects in a manner where individual UAS (or those in the same fleet) plan independently, which in turn requires an approach for runtime collision avoidance. The scalability of this will be essential in UAM applications as there will be no central authority to monitor and enforce UAS safety for hundreds of drones per square mile. This stands in contrast to the existing Air-Traffic Control and collision avoidance methods for commercial aviation, like TCAS-II, which was designed to operate in traffic densities of up to 0.3 aircraft per square nautical mile (nmi), i.e., 24 aircraft within a 5 nautical mile radius, which was the highest traffic density envisioned over the next 20 years [4].

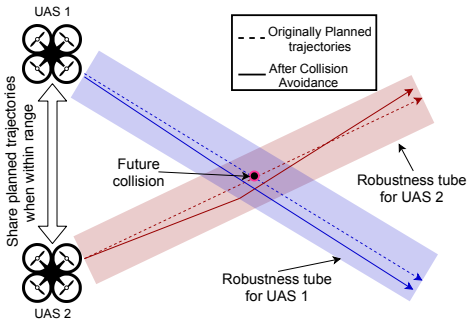

Fig. 1: Two UAS communicating their planned trajectories, and cooperatively maneuvering within their *robustness tubes* to avoid a potential collision in the future.

Airborne collision avoidance however is a complex problem. With high-speed UAS operating at low altitudes in cluttered urban airspace, decisions for collision avoidance need to be made within fractions of a second. The CA system must also be able to take into account the environment (e.g. buildings and other infrastructure, altitude limits, geofenced areas etc.) around it, making the problem harder than simply avoiding inter-UAS collisions.

To overcome the limitations outlined above, we aim to solve the following problems:

Problem 1: **Independent trajectory planning** for an individual UAS (or fleets run by the same operator) to satisfy spatial, temporal and reactive mission objectives specified using Signal Temporal Logic (STL), independently of other UAS that could be in the airspace.

Problem 2: **Real-time pairwise predictive airborne collision avoidance** such that UAS mission requirements satisfied by the trajectories obtained by solving problem 1 are not violated, e.g. UAS make it to their destinations in time while avoiding collisions with each other.

The airborne collision avoidance (Problem 2) poses a larger challenge, and is the primary focus here.

A. Contributions of this work

Our main contribution is *Learning-to-Fly* (L2F)[1], a scheme for real-time, on-the-fly collision avoidance between two UAS whose main features are:

1) *Systematic composition of machine learning and control theory:* We combine learning-based decision-making, and linear programming-based control to solve

[1]Department of Electrical and Systems Engineering, University of Pennsylvania, Philadelphia, PA, USA {alena.rodionova, jangkj, rahulm}@seas.upenn.edu.

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

[3]School of Electrical Engineering and Computer Science, Oregon State University, Corvallis, USA houssam.abbas@oregonstate.edu.

*The authors contributed equally.

[1]Videos of the simulations and demonstrations in this paper can be viewed at https://tinyurl.com/vvvuukh

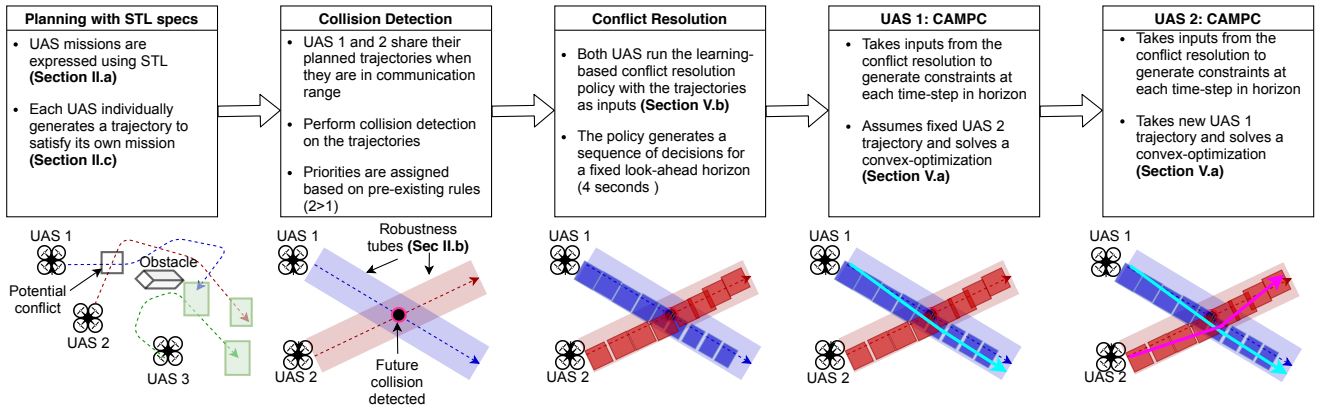

Fig. 2: Step-wise explanation and visualization of the framework. Each UAS generates its own trajectories to satisfy a mission expressed as a Signal Temporal Logic (STL) specification, e.g. regions in green are regions of interest for the UAS to visit, and the obstacle corresponds to infrastructure that all the UAS must avoid. When executing these trajectories, UAS 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 UAS execute a conflict resolution scheme that generates a set of additional constraints that the UAS must satisfy in order to avoid the collision. A co-operative CA-MPC controls the UAS in order to best satisfy these constraints while ensuring each UAS's STL specification is still satisfied. This results in new trajectories (in solid blue and pink) that will avoid the conflict and still stay within the pre-defined robustness tubes.

the problem in a decentralized manner. Unlike many other ad-hoc Machine Learning-based solutions, we provide a sound theoretical justification for our approach in Theorem 5.2. We also provide a sufficient condition for the scheme to work successfully (Theorem 5.1).

2) *A notion of priority among the UAS* can be encoded naturally in L2F, where the UAS with higher priority does not have to deviate from its originally planned trajectory until absolutely necessary.
3) *Computationally lightweight enough for real-time implementation:* Experimental results show that L2F, with a computation time in milliseconds can be used in a real-time implementation at a high-rate (10 Hz).
4) *High performance:* In the best case, L2F successfully results in 2-UAS collision avoidance 100% of the test cases, gracefully degrading to 90% for the worst case. Comparisons with other methods also show the superior performance of L2F.
5) *Enabling fast, independent planning for UAS with temporal logic objectives*, as individual UAS, or fleets of UAS run by the same operator, can plan for themselves without considering other UAS in the airspace while calling upon L2F for on-the-fly collision avoidance. For a 4-UAS case study, we demonstrate a speed up of $3.5\times$ over the centralized planning method of [2].
6) *Proof-of-concept demonstration* on Crazyflie quadrotor robots to show feasibility on real UAS.

B. Overview of approach and paper outline

In this paper, we aim to develop a framework for UAS traffic management (UTM) that solves problems 1 and 2. Figure 2 depicts the proposed planning and control process and indicates the relevant sections in the paper.

1) *Trajectory planning with Signal Temporal Logic (STL) specifications:* Each UAS, j, given the mission as an STL specification φ_j, generates a trajectory that robustly satisfies φ_j. The *robustness value* ρ_{φ_j}, associated with this trajectory, corresponds to the maximum deviation from the planned trajectory such that the UAS j still satisfies its mission φ_j.

Two UAS within communication range share a look-ahead of planned trajectories and if a future collision is detected, new trajectories are needed that still satisfy their original mission specifications. For this, we develop our decentralized approach L2F, which consists of two stages:

2) *Collision detection* and *Conflict resolution:* When a potential collision is detected, a supervised-learning based conflict resolution policy (CR-S), with predefined priority among the two UAS, generates a sequence of discrete decisions corresponding to maneuvers to avoid the collision.
3) *Distributed and co-operative Collision Avoidance MPC (CA-MPC):* The CA-MPC for each UAS takes as input the conflicting trajectories and the output of the conflict resolution policy, and controls the UAS to avoid collision.

In Section VI we evaluate our framework for 2 UAS collision avoidance through extensive simulations and compare its performance to other approaches. Section VII demonstrates a particular UTM framework case study. Finally, in Section VIII we discuss potential future directions.

C. Related work

1) UTM and Automatic Collision Avoidance approaches: The UAS Traffic Management (UTM) problem has been studied in various contexts. In the NASA/FAA Concept of Operations document [5], an airspace allocation scheme is outlined where individual UAS reserve airspace in the form of 4D polygons (space and time), and the polygons of different UAS are not allowed to overlap. Similarly, [6] presents a voxel-based airspace allocation approach. Our approach is less restrictive and allows overlaps in the 4D polygons, but performs maneuvers for collision avoidance when two UAS are on track to a collision (see Fig. 1). TCAS [4] and ACAS [7] systems for collision avoidance in

commercial aircrafts rely on transponders in the two aircrafts to communicate information for the collision avoidance modules. These generate recommendations for the pilots to follow and create vertical separation between aircrafts [8]. In the context of UAS, [9] uses vehicle-to-vehicle communication and tree-search based planning to achieve collision avoidance. ACAS-Xu [10], an automatic collision avoidance scheme for UAS relies on a look-up table to provide high-level recommendations to two UAS that have potentially colliding trajectories. 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, e.g. it can result in situations where ACAS-Xu recommends an action that avoids a nearby UAS but results in the primary UAS going close to a static obstacle. Our method avoids this as CA maneuvers are restricted to keeping UAS inside *robustness tubes* (see Fig. 2) such that mission requirements are not violated. For this reason, ACAS-Xu is currently only being explored for large, high-flying UAS [10] and is not directly applicable to the problem we study here.

2) Multi-agent planning with temporal logic objectives: Many approaches exist for the problem of planning for multiple robotic agents with temporal logic specifications. Most rely on abstract grid-based representations of the workspace [3], [11], or abstract dynamics of the agents [12], [13]. [14] combines a discrete planner with a continuous trajectory generator. Some methods [15], [16], [17] work for subsets of Linear Temporal Logic (LTL) that do not allow for explicit timing bounds on the mission requirements. While [3] uses a subset of LTL, safe-LTL$_f$ that allows them to express reach-avoid specifications with explicit timing constraints. However, in addition to a discretization of the workspace, they also restrict motion to a simple, discrete set of motion primitives. The predictive control method of [18] allows for using the expressiveness of the complete grammar STL for mission specifications. It handles a continuous workspace and linear dynamics of robots, however its reliance on mixed-integer encoding (similar to [19], [20]) for the STL specification limit use in planning/control for multiple agents in 3D workspaces as seen in [21]. The approach of [2] instead relies on optimizing a smooth (non-convex) function for generating trajectories for fleets of multi-rotor UAS with STL specifications. In our framework, we use the planning method of [2], but we let each UAS plan independently of each other. We ensure the safe operation of all UAS in the airspace through the use of our predictive collision avoidance scheme.

II. Preliminaries

We use Signal Temporal Logic (STL) to specify the mission objectives that the UAS need to satisfy (Problem 1). This section provides a brief introduction to STL and the trajectory generation approach.

A. Introduction to Signal Temporal Logic

Signal Temporal Logic (STL) [22] is a behavioral specification language that can be used to encode requirements on signals. The grammar of STL [18] allows for capturing a rich set of behavioral requirements using temporal operators, such as *Always* ($\Box$) and *Eventually* ($\Diamond$), as well as logical operators like *And* ($\wedge$), *Or* ($\vee$), and *negation* ($\neg$). With these operators, an STL specification φ is defined over a signal, e.g. over the trajectories of quad-rotor robots, and evaluates to either *True* or *False*. The following example demonstrates STL to capture operational requirements for two UAS:

Example 1: (A two UAS timed reach-avoid problem) Two quad-rotor UAS are tasked with a mission with spatial and temporal requirements in the workspace shown in Fig. 3:

1) The two UAS have to reach a Goal set (shown in green), or a region of interest, within a time of 6 seconds after starting. UAS j (where $j \in \{1, 2\}$), with position denoted by p_j, has to satisfy: $\varphi_{reach,j} = \Diamond_{[0,6]}(p_j \in \text{Goal})$. The *Eventually* operator over the time interval $[0, 6]$ requires UAS j to be inside the set Goal at some point within 6 seconds.
2) In addition, the two UAS 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: $\varphi_{\text{avoid},j} = \Box_{[0,6]}\neg(p_j \in \text{Unsafe})$
3) Finally, the two UAS should also be separated by at least δ meters along every axis of motion: $\varphi_{\text{separation}} = \Box_{[0,6]}||p_1 - p_2||_\infty \geq \delta$

The 2-UAS timed reach-avoid specification is thus:

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

In order to satisfy φ, a planning method generates trajectories $\mathbf{p}_1$ 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 [18]. In this work, we consider specifications such that the horizon is bounded. More details on STL can be found in [22] or [18]. In this paper, we consider discrete-time STL semantics which are defined over discrete-time trajectories.

B. Robustness of STL specifications

For a time domain $\mathbb{T} = [0, T]$ with sampling time dt, the signal space $X^{\mathbb{T}}$ is the set of all signals $\mathbf{x} : \mathbb{T} \to X$. The *Robustness* value [23] ρ_φ of an STL formula φ, with respect to the signal $\mathbf{x}$ that it is defined over, is a real-valued function of $\mathbf{x}$ that has the important following property:

Theorem 2.1: [23] For any $\mathbf{x} \in X^{\mathbb{T}}$ 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.

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

Corollary 1.1: Given a discrete-time trajectory $\mathbf{x}$ such that $\mathbf{x} \models \varphi$ with robustness value $\rho > 0$, then any trajectory $\mathbf{x}'$ that is within ρ of $\mathbf{x}$ at each time step, i.e. $||x_t - x'_t||_\infty < \rho \, \forall t \in \mathbb{T}$, is such that $\mathbf{x}' \models \varphi$ (also satisfies φ).

C. UAS planning with STL specifications

Fly-by-logic [2] generates trajectories by centrally plan ning for fleets of UAS with STL specifications, e.g. th specification $\varphi_{\text{reach-avoid}}$ of example 1. It maximizes a smoothl approximation $\tilde{\rho}_\varphi$ of the robustness function [21] by pickin waypoints (connected via jerk-minimzing splines [24]) fo all UAS through a centralized, non-convex optimization.

While successful in planning for multiple multi-rotor UAS performance degrades as the number of UAS being planne for increases. The non-convex optimization involving th variables of all the UAS becomes harder as the number o variables increase [2] in particular because for J UAS, $\binom{J}{2}$ terms for pair-wise separation between the UAS are needed. Taking this into account, we use the underlying optimization of [2] 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 UAS. Associated with these trajectories, $\mathbf{x}_j$ is a robustness values ρ_{φ_j}. Ignoring the collision avoidance requirement ($\varphi_{\text{separation}}$) in the planning stage allows for the specification of (1) to be decoupled across UAS, but now requires online pairwise UAS collision avoidance if the planned trajectories are in conflict. This is covered in the following section.

Note: In the following sections, $\mathbf{x}$ will refer to a full-state (discrete-time, finite duration) trajectory for a UAS. We will also use $\mathbf{p}$ to refer to the position components in that trajectory, the position trajectory. x_k (and p_k) refer to the components of the trajectory at time step k.

III. Problem formulation: Collision Avoidance

While flying their planned trajectories (from the previous section), two UAS that are within communication range share a look-ahead of their trajectories and check for a potential collision at any time step k in this look-ahead horizon of N time steps. We assume the UAS can communicate with each other in a manner that allows for enough advance notice for avoiding collisions, e.g. using 5G technology. The details of this are beyond the scope of this paper.

Definition 1: **2-UAS Conflict**: Two UAS, 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.

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 $\mathbf{P}_j = \mathbf{p}_j \oplus \mathbb{B}_{\rho_\varphi}$. We say the *radius* of this tube is ρ (in the inf-norm sense). Here $\mathbb{B}_\rho$ is a 3D cube with sides 2ρ and $\oplus$ is the Minkowski sum operation.
See an example of the robustness tube in Figure 3.

Note: As long as a UAS stays within its robustness tube, it will satisfy the STL specification φ for the which the trajectory was generated for (see Corollary 1.1).

[2]A more general polyhedral constraint of the form $H(p_{1,k} - p_{2,k}) < g$ can be used for defining the conflict without loss of generality.

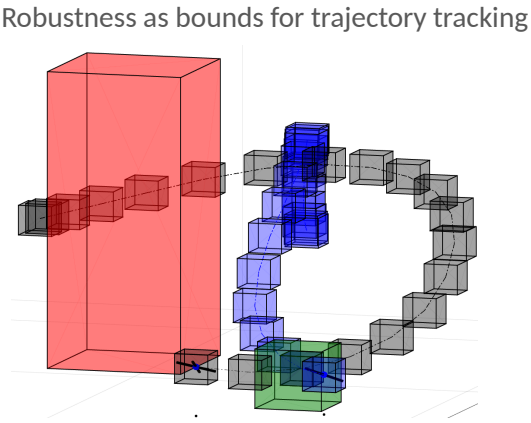

Fig. 3: Discrete time trajectories of two UAS, and their associated robustness tubes (see def. 2) in gray and purple. The trajectories satisfy a reach-avoid specification, see example 1. Unsafe set is in red and the Goal set is in green.

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

Assumption 1: For each of the two UAS 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 just wide enough to have two UAS placed along opposing edges (of a cube at the same time step) 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 Sec. II-C). Now we define the problem of collision avoidance with satisfaction of STL specifications:

Problem 3: Given two planned N-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:

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

$$p'_{j,k} \in \mathbf{P}_j \, \forall j = 1, 2, \, \forall k = 0, \ldots, N \tag{2b}$$

This implies that 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 (see Corollary 1.1).

A. Convex constraints for collision avoidance

Let z_k be the difference in UAS positions at time step k. For two UAS not to be in conflict, we need

$$z_k = p_{1,k} - p_{2,k} \notin \mathbb{B}_\delta \, \forall k \tag{3}$$

This is a non-convex constraint. For a computationally tractable controller formulation which solves problem 3, we define convex constraints that when satisfied imply eq. (3).

The 3D cube $\mathbb{B}_\delta$ can be defined by a set of linear inequality constraints of the form $\widetilde{H}^i z \leq \widetilde{g}^i \, \forall i = 1, \ldots, 6$.Eq. (3) is satisfied when $\exists i \, | \widetilde{H}^i z > \widetilde{g}_i$. Let $H = -\widetilde{H}$ and $g = -\widetilde{g}$, then for any $i \in \{1, \ldots, 6\}$,

$$H^i(p_{1,k} - p_{2,k}) < g^i \Rightarrow (p_{1,k} - p_{2,k}) \notin \mathbb{B}_\delta \tag{4}$$

Intuitively, picking one i at time step k results in a configuration (in position space) where the two UAS are separated

in one of two ways along one of three axis of motion[3], e.g. at a time step k if we select $i|H^i = \begin{bmatrix}0 & 0 & 1\end{bmatrix}$, $g^i = -\delta$, it implies than UAS 2 flies over UAS 1 by δ m, and so on.

IV. Centralized solution: MILP formulation

Let the dynamics of either UAS[4] be of the form $x_{k+1} = Ax_k + Bu_k$. The states $x_k \in \mathbb{R}^6$ here are the positions and velocities in the 3D space, or $x_k = [p_k,\, v_k]^T$ (here p and v are the positions and velocities in the 3D space). The inputs $u_k \in \mathbb{R}^3$ are the thrust, roll and pitch of the UAS. The matrices A and B can be obtained through linearization of the UAS dynamics around hover and discretization in time [25]. Let C be the observation matrix such that $p_k = Cx_k$.

For N steps into the future with a conflict, solving the following receding horizon MILP over the variables of the two UAS would result in new trajectories $\mathbf{p}'_1$, $\mathbf{p}'_2$ that satisfy the minimum separation requirement (3). Let $\mathbf{x}_j \in \mathbb{R}^{6(N+1)}$ be the pre-planned full state trajectories, $\mathbf{x}'_j \in \mathbb{R}^{6(N+1)}$ the new full state trajectories and $\mathbf{u}'_j \in \mathbb{R}^{3N}$ the new controls to be computed for the two UAS ($j = 1, 2$). Let $\mathbf{b} \in \{0,1\}^{6(N+1)}$ be binary decision variables, and M is a large positive number, then the MILP problem is defined as:

$$
\begin{aligned}
\min_{\mathbf{u}'_1,\, \mathbf{u}'_2,\, \mathbf{b}|\mathbf{x}_1,\, \mathbf{x}_2} \; & L(\mathbf{x}'_1,\, \mathbf{u}'_1,\, \mathbf{x}'_2,\, \mathbf{u}'_2) \\
& x'_{j,0} = x_{j,0}\, \forall j \in \{1,2\} \\
& x'_{j,k+1} = Ax'_{j,k} + Bu'_{j,k}\, \forall k \in \{0,\ldots,N-1\},\, \forall j \in \{1,2\} \\
& Cx'_{j,k} \in P_{j,k}\, \forall k \in \{0,\ldots,N\},\, \forall j \in \{1,2\} \\
& H^i C(x'_{1,k} - x'_{2,k}) \leq g_i + M(1 - b_{i,k})\, \forall k \in \{0,\ldots,N\}, \forall i \in \{1,\ldots,6\} \\
& \sum_{i=1}^{6} b_k^i \geq 1\, \forall k \in \{0,\ldots,N\} \\
& u'_{j,k} \in U\, \forall k \in \{0,\ldots,N\},\, \forall j \in \{1,2\} \\
& x'_{j,k} \in X\, \forall k \in \{0,\ldots,N+1\}, \forall j \in \{1,2\}
\end{aligned}
\tag{5}
$$

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 $\mathbb{B}_\delta$. A solution (when it exists) to this MILP results in new trajectories that avoid collisions and stay within their respective robustness tubes of the original trajectories. However, this method relies on solving for a pair of UAS in a centralized manner. Also, it introduces 6 times as many variables (and constraints) as the time horizon of the optimization, which could make the MILP computationally intractable for a real-time implementation. Therefore, we develop a decentralized approach in the following sections.

V. Decentralized solution: Learning-to-Fly

The distributed and co-operative collision avoidance MPC scheme of Section V-A with the conflict resolution algorithm described in Section V-B form the online collision avoidance scheme, *Learning-to-Fly* (L2F), our main contribution.

We assume that the two UAS can communicate their pre-planned N-step trajectories $\mathbf{p}_1$, $\mathbf{p}_2$ to each other (refer to Sec. II-C). Instead of solving the centralized MILP, we want to solve problem 3 by following these steps:

1) **Conflict resolution:** UAS 1 and 2 make a *sequence of decisions*, $\mathbf{d} = (d_0, \ldots, d_N)$ to avoid collision. Each $d_k \in \{1, \ldots 6\}$ represents a particular choice of H and g at time step k, see eq. 4. Section V-B describes our proposed learning-based method for conflict resolution.
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 V-A provides more details on CA-MPC steps 2 and 3.

The overall algorithm is shown in Alg. 1. The visualization of the above steps is presented in Fig. 2. Such decentralized approach differs from the centralized MILP approach, where both the binary decision variables and continuous control variables for each UAS are decided concurrently.

A. Distributed and co-operative collision avoidance MPC

Each UAS $j \in \{1,2\}$ solves the following Collision Avoidance MPC optimization[5]:

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

$$
\begin{aligned}
\min_{\mathbf{u}'_j, \lambda_j | \mathbf{x}_j, \mathbf{x}_{avoid}} & \sum_k \lambda_{j,k} \\
& x'_{j,0} = x_{j,0} \\
& x'_{j,k+1} = Ax'_{j,k} + Bu'_{j,k}\, \forall k = \{0,\ldots,N-1\} \\
& Cx'_{j,k} \in P_{j,k}\, \forall k = \{0,\ldots,N\} \\
& prty_j \cdot H^{d_k} C(x_{avoid,k} - x'_{j,k}) \leq g^{d_k} + \lambda_{j,k}\, \forall k = \{0,\ldots,N\} \\
& \lambda_{j,k} \geq 0\, \forall k = \{0,\ldots,N\} \\
& u'_{j,k} \in U\, \forall k = \{0,\ldots,N\} \\
& x'_{j,k} \in X\, \forall k \in \{0,\ldots,N+1\}
\end{aligned}
\tag{6}
$$

where, $\mathbf{x}_j$ is the pre-planned trajectory of UAS j, $\mathbf{x}_{avoid}$ is the pre-planned trajectory from which UAS j must attain a minimum separation, $prty_j \in \{-1, +1\}$ is the priority of UAS j w.r.t the other UAS in conflict. The decision sequence $\mathbf{d}$ is represented as H^{d_k}, g^{d_k}. 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 (4) w.r.t the pre-planned trajectory $\mathbf{x}_{avoid}$ of the UAS in conflict. The constraints in (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 UAS, see eq. (4).

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

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

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

[5]Enforcing the separation constraint at each time step can lead to a restrictive formulation, especially in cases where the two UAS 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/uex7722`), which is something ACAS-Xu was not designed for.

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

Alg. 1 is designed to be computationally lighter th MILP approach (see Section IV), but unlike the MIL not complete. In Section VI, through extensive simul we show that the L2F approach demonstrates a sign improvement in runtime while maintaining comparabl formance in terms of separation.

Algorithm 1 Learning-to-Fly: Decentralized and coope collision avoidance for two UAS. Also see fig. 2.

Data: Pre-planned trajectories, robustness tubes
Result: Sequence of control signals $\mathbf{u}_1'$, $\mathbf{u}_2'$ for the two
Get $\mathbf{d}$ from conflict resolution
UAS 1 solves CA-MPC optimization (6):
$(\mathbf{x}_1', \mathbf{u}_1', \lambda_1) = \textbf{CA-MPC}_1(\mathbf{x}_1, \mathbf{x}_2, P_1, \mathbf{d}, -1)$
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):
  $(\mathbf{x}_2', \mathbf{u}_2', \lambda_2) = \textbf{CA-MPC}_2(\mathbf{x}_2, \mathbf{x}_1', P_2, d, +1)$
  if $\sum_k \lambda_{2,k} = 0$ **then**
    Done: UAS 2 has created separation
  else
    if $||p_{1,k}' - p_{2,k}'|| \geq \delta \, \forall k = 0, \ldots, N$ **then**
      Done: UAS 1 and UAS 2 created separation
    else
      Not done: UAS still violate eq. 2a
    end
  end
end
Apply control signals $\mathbf{u}_1'$, $\mathbf{u}_2'$ if **Done**; else **Fail**.

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

Definition 5.1 (Zero-slack solution): The solution of the CA-MPC optimization (6), is called the *zero-slack solution* if for a given decision sequence $\mathbf{d}$ either

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

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

The two following theorems make important connections between feasible solutions for MILP and CA-MPC formulations. They are the consequence of the construction of CA-MPC optimizations. We omit the proofs for brevity.

Theorem 5.1 (Sufficient condition for CA): Zero-slack solution of (6) implies that the resulting trajectories for two UAS are non-conflicting and within the robustness tubes of the initial trajectories[6].

[6]Theorem 5.1 formulates a conservative result as (4) is a convex under approximation of the originally non-convex collision avoidance constraint (3). Indeed, non-zero slack $\exists k | \lambda_{2,k} > 0$ does not necessarily imply the violation of the mutual separation requirement (2a). The control signals u_1', u_2' computed by alg. 1 can therefore in some instances still create separation between drones even when the conditions of Theorem 5.1 are not satisfied.

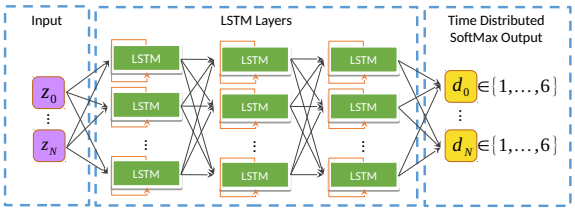

Fig. 4: Proposed LSTM model architecture for CR-S. LSTM layers are shown unrolled over N time steps. The inputs are z_k which are the differences between the planned UAS positions, and the outputs are decisions d_k for conflict resolution at each time k in the horizon.

Theorem 5.2 (Existence of the zero-slack solution): Feasibility of the MILP problem (5) implies the existence of the zero-slack solution of CA-MPC optimization (6).

The Theorem 5.2 states that the binary decision variables b_k^i selected by the feasible solution of the MILP problem (5), when used to select the constraints (defined by H, g) for the CA-MPC formulations for UAS 1 and 2, imply the existence of a zero-slack solution of (6).

B. Learning-based conflict resolution

Motivated by Theorem 5.2, we propose to learn the conflict resolution policy from the MILP solutions. To do so, we use a *Long Short-Term Memory* (LSTM) [26] recurrent neural network augmented with fully-connected layers. LSTMs perform better than traditional recurrent neural networks on sequential prediction tasks [27].

The network is trained to map a difference trajectory $\mathbf{z} = \mathbf{x}_1 - \mathbf{x}_2$ (as in eq. (3)) to a decision sequence $\mathbf{d}$ that de-conflicts pre-planned trajectories $\mathbf{x}_1$ and $\mathbf{x}_2$. For creating the training set, $\mathbf{d}$ is produced by solving the MILP problem 5, i.e. obtaining a sequence of binary decision variables $\mathbf{b} \in \{0,1\}^{6(N+1)}$ and translating it into the decision sequence $\mathbf{d} \in \{1,\ldots,6\}^{N+1}$.

The proposed architecture is presented in Figure 4. 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 such that each value is a decision d_k, $k = \{0,\ldots,N\}$.

VI. Experimental Evaluation

A. Experimental setup

All the simulations were performed on a computer with an AMD Ryzen 7 2700 8-core processor and 16GB RAM, running Ubuntu 18.04. The MILP formulation was implemented in MATLAB using Yalmip [28] with MOSEK v8 as the solver. The learning-based approach was implemented in Python 3 with Tensorflow 1.14 and Keras API and Casadi with QPOASES as the solver. We implemented the CA-MPC using CVXGEN for a measurement of computation times and real-time implementation for experiments of actual hardware.

For the experiments, we set minimum separation to $\delta = 0.1$m. The learning-based CR scheme was trained for $\rho = 0.055$ which is close to the lower bound in assumption 1.

We have generated the data set of 14K training and 10K test conflicting trajectories using the minimum-jerk trajectory generation algorithm from [2]. The time horizon was set to $T = 4$s and $dt = 0.1$s. The initial and final waypoints were

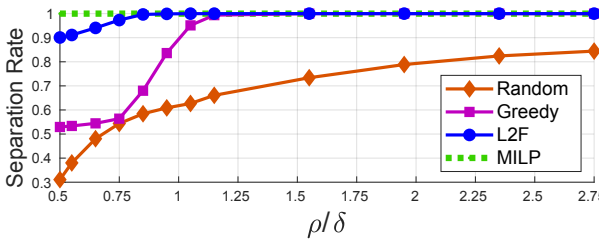

Fig. 5: Model sensitivity analysis with respect to variations fraction ρ/δ, which connects the minimum allowable robustness tube radius ρ to the minimum allowable separation between two UAS δ, see Assumption 1. A higher ρ/δ implies there is more room within the robustness tubes to maneuver within for CA.

sampled uniformly at random from two 3D cubes close to the fixed collision point, initial velocities were set to zero.

We have trained and ran experiments for various network configurations. For each model, 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 Adam optimizer with training rate of 0.001. The model with 3 LSTM layers with 128 neurons each has been chosen as the default learning-based CR model.

B. Results and comparison to other methods

We analyzed three other methods alongside the proposed learning-based approach for L2F.

1) A **random** decision approach which outputs a sequence sampled from the discrete uniform distribution.
2) A **greedy** approach that selects the discrete decisions for which the most distance between the two UAS available at each time step.
3) A centralized **MILP** solution that picks a decision corresponding to a binary decision variable in (5).

For the evaluation, we measured and compared the **separation rate** and the **computation time** over 10K test trajectories. *Separation rate* defines the fraction of given initially conflicting trajectories for which UAS managed to achieve minimum separation.

Figure 5 shows the trade-off between performance in terms of separation rate and ρ/δ fraction, which defines the connection between the robustness tube ρ and the minimum separation δ. Higher ρ/δ implies wider robustness tubes for the UAS to maneuver within, which should make the CA task easier. In the case of $\rho/\delta = 0.5$, where the robustness tubes are just wide enough to fit two UAS (see assumption 1), we see the L2F significantly outperforms the methods (excluding the MILP). As the ratio grows, the performance of all methods improve with L2F still outperforming the others, topping out to achieve a best case separation of 1. The worst-case performance for L2F is 0.9 which is again significantly better than the other approaches.

Table I shows the separation rates for three different ρ/δ values as well as the computation times for conflict resolution schemes plus the CA-MPC optimizations. In terms of separation rate, L2F outperformed the random and the greedy approaches. The centralized MILP outperformed the L2F, however, the computation time for the centralized approach was orders of magnitude higher than L2F. These shows the benefits of L2F compared to other approaches, especially when considering the success-computation time trade-off.

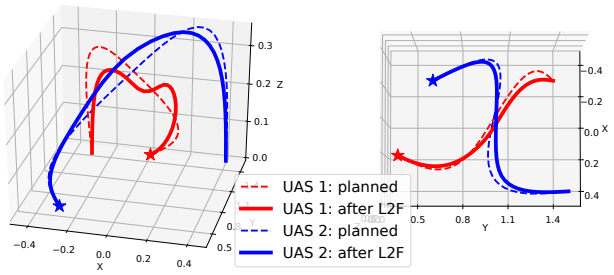

Fig. 6: Trajectories for 2 UAS from different angles. The dashed (planned) trajectories have a collision at the halfway point. The solid ones, generated through L2F method, avoid the collision while remaining within the robustness tube of the original trajectories. Initial UAS positions marked as stars. Playback of the scenario is at `https://tinyurl.com/y8cm65ya`.

CA Scheme	Separation Rate			Computation time	
	ρ/δ = 0.5	ρ/δ = 0.95	ρ/δ = 1.15	Mean	Std
Random	0.311	0.609	0.661	2.02ms	0.17ms
Greedy	0.529	0.836	0.994	3.82ms	0.25ms
L2F	0.901	0.999	1	9.36ms	1.75ms
MILP	1	1	1	68.5s	87.3s

TABLE I: Separation rates and computation times (mean and standard deviation) comparison of different CA schemes. *Separation rate* is the fraction of conflicting trajectories for which separation requirement (2a) is satisfied after CA.

Figure 6 shows an example of two UAS trajectories before and after collision avoidance through L2F method. In addition, in order to evaluate the feasibility of the deconflicted trajectories, we have also ran experiments using two Crazyflie quad-rotor robots. Video recordings of the actual flights and additional simulations can be found at `https://tinyurl.com/yxttq7l5`.

VII. Case Study: Independent Planning and L2F for a 4-UAS Example

Figure 7 depicts a UAS case-study with a reach-avoid mission. Scenario consists of four UAS which must reach desired goal states within 4 seconds while avoiding the wall obstacle and each other. Each UAS $j \in \{1, \dots, 4\}$ specification can be defined as:

$$\varphi_j = \Diamond_{[0,4]}(p_j \in \text{Goal}) \ \wedge \ \Box_{[0,4]} \neg (p_j \in \text{Wall}) \qquad (7)$$

A pairwise separations requirement of 0.1 meters is enforced for all UAS, therefore, the overall mission specification is:

$$\varphi_{\text{mission}} = \bigwedge_{j=1}^{4} \varphi_j \ \wedge \ \bigwedge_{j \neq j'} \Box_{[0,4]} ||p_j - p_{j'}|| \geq 0.1 \qquad (8)$$

First, we solved the planning problem for all four UAS in a centralized manner following approach from [2] Next, we solved the planning problem for each UAS j and its specification φ_j independently, with calling L2F 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 (8).

Simulation results. We have simulated the scenario for 100 different initial conditions. The average computation

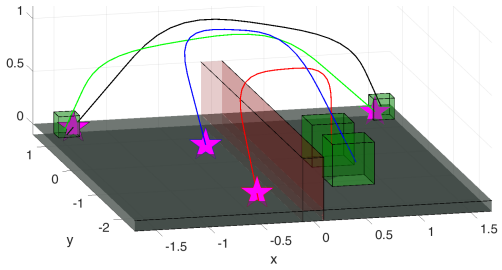

Fig. 7: Workspace for the case study scenario. Trajectories for 4 UAS (magenta stars) reaching their goal sets (green boxes) within 4 seconds, while not crashing into the vertical wall (in red). A pairwise separation requirement of 0.1m is enforced. Simulations are available at `https://tinyurl.com/t8bwwqk`.

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 speed up of $3.5\times$ for the individual UAS planning versus centralized [2].

VIII. Conclusion

Summary. We developed *Learning-to-Fly* (L2F), a two-stage, on-the-fly and predictive collision avoidance approach that combines learning-based decision-making for conflict resolution with decentralized linear programming-based UAS control. Through extensive simulations and demonstrations on real quadrotor drones we show that L2F, with a run-time < 10ms is computationally fast enough for real-time implementation. It is successful in resolving 100% of collisions in most cases, with a graceful degradation to the worst-case performance of 90% when there is little room for the UAS to maneuver. L2F also enables independent UAS planning, speeding up the process compared to centrally planning for all the UAS in the airspace. A 4-UAS case study shows that the independent planning is $3.5\times$-faster.

Limitations and Future Work. While pairwise collision avoidance is sufficient when the airspace density is low, in the future we will extend the approach to cases where more than two UAS could be in conflict with each other. As L2F does not always succeed, we plan to investigate this further and use the failure cases as counterexamples to make the learning-based models better. In general, we expect L2F to be realized within a larger UTM system with additional contingencies (e.g. FAA Lost Link procedures [29]), including the possibility of online re-planning of missions when L2F cannot guarantee collision avoidance.

References

[1] National Academies of Sciences, Engineering, and Medicine, *Advanced Aerial Mobility: A National Blueprint*. The National Academies Press, 2020.

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

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

[4] Federal Aviation Administration, "Introduction to TCAS II, V7.1," https://www.faa.gov/documentlibrary/media/advisory_circular/tcas%20ii%20v7.1%20intro%20booklet.pdf, accessed: 2019-10-19.

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

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

[7] M. J. Kochenderfer, J. E. Holland, and J. P. Chryssanthacopoulos, "Next-generation airborne collision avoidance system," MIT-Lincoln Laboratory, Lexington, US, Tech. Rep., 2012.

[8] J.-B. Jeannin, K. Ghorbal, Y. Kouskoulas, R. Gardner, A. Schmidt, E. Zawadzki, and A. Platzer, "Formal verification of acas x, an industrial airborne collision avoidance system," in *International Conference on Embedded Software*, Piscataway, NJ, USA, 2015.

[9] A. Chakrabarty, C. Ippolito, J. Baculi, K. Krishnakumar, and S. Hening, "Vehicle to vehicle (v2v) communication for collision avoidance for multi-copters flying in utm –tcl4," 01 2019.

[10] G. Manfredi and Y. Jestin, "An introduction to acas xu and the challenges ahead," in *2016 IEEE/AIAA 35th Digital Avionics Systems Conference (DASC)*, Sep. 2016, pp. 1–9.

[11] J. A. DeCastro, J. Alonso-Mora, V. Raman, and H. Kress-Gazit, "Collision-free reactive mission and motion planning for multi-robot systems," in *Springer Proceedings in Advanced Robotics*, 2017.

[12] A. Desai, I. Saha, Y. Jianqiao, S. Qadeer, and S. A. Seshia, "Drona: A framework for safe distributed mobile robotics," in *ACM/IEEE International Conference on Cyber-Physical Systems*, 2017.

[13] D. Aksaray, A. Jones, Z. Kong, M. Schwager, and C. Belta, "Q-learning for robust satisfaction of signal temporal logic specifications," in *IEEE Conference on Decision and Control*, 2016.

[14] X. Ma, Z. Jiao, and Z. Wang, "Decentralized prioritized motion planning for multiple autonomous uavs in 3d polygonal obstacle environments," in *Inter. Conf. on Unmanned Aircraft Systems*, 2016.

[15] M. Kloetzer and C. Belta, "A fully automated framework for control of linear systems from temporal logic specifications," *IEEE Transactions on Automatic Control*, vol. 53, no. 1, pp. 287–297, Feb 2008.

[16] G. E. Fainekos, H. Kress-Gazit, and G. J. Pappas, "Hybrid controllers for path planning: A temporal logic approach," in *Proce. of the 44th IEEE Conf. on Decision and Control*, Dec 2005, pp. 4885–4890.

[17] M. Kloetzer and C. Belta, "Hierarchical abstractions for robotic swarms," in *Proc. of 2006 IEEE Inter. Conf. on Robotics and Automation*, May 2006, pp. 952–957.

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

[19] S. Saha and A. A. Julius, "An milp approach for real-time optimal controller synthesis with metric temporal logic specifications," in *Proceedings of the 2016 American Control Conference (ACC)*, 2016.

[20] S. Karaman and E. Frazzoli, "Linear temporal logic vehicle routing with applications to multi-uav mission planning," *International Journal of Robust and Nonlinear Control*, 2011.

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

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

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

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

[25] Y. V. Pant, H. Abbas, K. Mohta, T. X. Nghiem, J. Devietti, and R. Mangharam, "Co-design of anytime computation and robust control," in *2015 IEEE Real-Time Systems Symposium*, 2015.

[26] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[27] F. A. Gers, N. N. Schraudolph, and J. Schmidhuber, "Learning precise timing with lstm recurrent networks," *Journal of machine learning research*, vol. 3, no. Aug, pp. 115–143, 2002.

[28] J. Löfberg, "Yalmip: Toolbox for modeling and optim. in matlab."

[29] B. Pastakia, J. Won, R. Sollenberger, D. Aubuchon, S. Entis, and L. Thompson, "UAS Operational Assessment: Contingency Operations Simulation Report," Federal Aviation Administration, Tech. Rep., 2015.