

# A Signal Temporal Logic Approach for Task-Based Coordination of Multi-Aerial Systems: a Wind Turbine Inspection Case Study

Giuseppe Silano<sup>a,f</sup>, Alvaro Caballero<sup>b</sup>, Davide Liuzza<sup>c,d</sup>, Luigi Iannelli<sup>c</sup>, Stjepan Bogdan<sup>e</sup>, Martin Saska<sup>a</sup>

<sup>a</sup>Department of Cybernetics, Faculty of Electrical Engineering, Czech Technical University in Prague, Prague, Czech Republic (emails: {giuseppe.silano, martin.saska}@fel.cvut.cz).

<sup>b</sup>GRVC Robotics Laboratory, University of Seville, Seville, Spain (email: alvarocaballero@us.es).

<sup>c</sup>Department of Engineering, University of Sannio in Benevento, Benevento, Italy (emails: {davide.liuzza, luigi.iannelli}@unisannio.it).

<sup>d</sup>Fusion and Technology for Nuclear Safety and Security Department, Italian National Agency for New Technologies, Energy and Sustainable Economic Development (ENEA), Frascati, Italy (email: davide.liuzza@enea.it).

<sup>e</sup>Faculty of Electrical Engineering and Computing, University of Zagreb, Zagreb, Croatia (email: stjepan.bogdan@fer.hr).

<sup>f</sup>Corresponding author

## Abstract

The paper addresses task assignment and trajectory generation for collaborative inspection missions using a fleet of multi-rotors, focusing on the wind turbine inspection scenario. The proposed solution enables safe and feasible trajectories while accommodating heterogeneous time-bound constraints and vehicle physical limits. An optimization problem is formulated to meet mission objectives and temporal requirements encoded as Signal Temporal Logic (STL) specifications. Additionally, an event-triggered replanner is introduced to address unforeseen events and compensate for lost time. Furthermore, a generalized robustness scoring method is employed to reflect user preferences and mitigate task conflicts. The effectiveness of the proposed approach is demonstrated through MATLAB and Gazebo simulations, as well as field multi-robot experiments in a mock-up scenario.

## Keywords:

Aerial Systems: Applications, Formal Methods in Robotics and Automation, Task and Motion Planning, Multi-Robot Systems.

## 1. Introduction

The increasing use of Unmanned Aerial Vehicles (UAVs) for civilian infrastructure inspections has paved the way for aerial vehicles with greater autonomy, thereby enhancing the safety, speed, and accuracy of these inspections [1]. The capacity for repeated operations enables the monitoring of infrastructure changes over time, providing benefits to a variety of applications such as inspecting oil and gas pipelines [2], wind turbine blades [3], power transmission lines [4], towers and bridges [5].

Presently, the most widespread approach is the use of UAVs that are controlled by highly trained operators. These experts are knowledgeable about the required information and site-specific challenges, such as obstacles, weather conditions, and time of day. On-board cameras and sensors are utilized to gather data which is then processed by specialist teams for further analysis. However, this approach has two major limitations: First, flight operations can be hazardous for operators who must maintain continuous visual contact with the UAVs, especially in environments at height; Second, the process is time-consuming, costly, and prone to human error [6, 7, 8]. Over time, various methods for automating the inspection task have been proposed in the literature [5, 3, 4, 2, 1, 9]. Despite this, significant challenges still exist in making inspections autonomous, including the reliability of navigation systems, radio interference, limited battery life, and unpredictable envi-

ronmental events, which pose risks to mission success [10].

Therefore, there is a clear need for safe and effective techniques that can enhance maintenance and inspection operations, reduce risks, and decrease costs for companies. Formal methods [11] can address these challenges by offering a concise way to express complex mission specifications and time constraints by leveraging their similarity to natural language commands and the use of connectivity operators. Specifically, constraints and objectives are encapsulated within a single temporal formula. Notably, Signal Temporal Logic (STL) [12], particularly in its robust formulation [13], employs the concept of quantitative semantics, meaning that it not only verifies that the system execution satisfies desired requirements, but also provides a metric of how well these requirements are met, referred to as *robustness* [13]. This results in an optimization problem aimed at achieving high robustness scores values through the generation of a trajectory that satisfies the desired specifications while ensuring feasibility within dynamic and constraint margins.

In this paper, wind turbine inspection (see Figure 1) serves as a case scenario for presenting a multi-UAV motion planning approach leveraging on STL specifications for collaborative inspection missions. This scenario illustrates sophisticated planning, involving the coordination of multi-rotor UAVs to avoid collisions with each other and the environment, accomplish diverse time-bound inspection tasks, adhere to the vehicle's dynamical constraints, and ensure comprehensive infrastructure coverage. The mission requirements are formulated as an STL



Figure 1: A multi-rotor UAV conducting an inspection of a wind turbine, capturing videos and pictures of the nacelle, rotor shaft and blades, as well as their surrounding environment, to perform a preliminary remote evaluation [3].

formula, and the robustness of the formula is optimized by setting up a nonlinear, non-convex max-min optimization problem. To manage the complexity of this nonlinear optimization, a hierarchical approach is employed, starting with solving a Mixed-Integer Linear Programming (MILP) planning problem and subsequently feeding the final STL optimizer.

### 1.1. Related work

Extensive research has been done in the fields of autonomous navigation and coordination capabilities for UAVs [5, 6, 7, 10]. The primary objectives of this research have been to: (i) enable UAVs to navigate autonomously and reach safe conditions while avoiding unsafe behaviors (e.g., collisions, crashes), and (ii) enhancing the vehicles' ability to coordinate for data collection and executing appropriate actions. Accurate trajectory planning and task assignment are crucial in both of these areas, not only for individual vehicles, but also for fleets of UAVs with varying characteristics and capabilities that must cooperate, avoid obstacles, and meet mission requirements simultaneously in the same operational area.

The literature on *autonomous navigation* presents a diversity of point-to-point navigation approaches [14], including potential field methods [15] and vehicle-routing problem formulations [16]. Many studies prioritize drone mission endurance to maximize coverage area by accounting for flight times. However, these approaches frequently overlook drone dynamics and physical constraints, resulting in infeasible paths. The problem then becomes a combinatorial optimization challenge, as exemplified by the well-known Vehicle Routing Problem (VRP) [16], which becomes increasingly unsolvable within a reasonable time frame as the complexity exponentially increases with the number of vehicles and variables. In such cases, heuristic methods are often employed to simplify the complexity and identify the most viable solution.

Research has explored various approaches for endowing UAVs with *coordination capabilities* [17, 18, 19, 20]. Some studies have focused on centralized multi-agent algorithms for coordinating and planning safe and dynamically feasible trajectories [17]. Others propose methods to convert the multi-

agent task allocation problem into a search space that can be solved using the Particle Swarm Optimization (PSO) algorithm [18]. Additionally, some approaches employ knowledge transfer techniques between agents to achieve collision avoidance and safety [19]. These approaches can efficiently coordinate and compute trajectories for multiple agents [19, 20], but do not guarantee that vehicles will complete tasks within specified time windows. Furthermore, encoding possibly different complex temporal requirements into the optimization problem, such as a requirement for multiple drones to visit designated regions at specific times with certain behaviors while ensuring safety, requires a systematic approach.

In addressing the motion planning challenge for multi-robot systems, various approaches have been explored in existing literature [20, 21, 22]. Many solutions rely on abstract agent dynamics [20] or abstract grid-based environment representations [21], integrating a discrete planner with a continuous motion generator. While these methods can swiftly compute collision-free motions for numerous agents, they fail to guarantee adherence to physical constraints or task completion within specified time frames. Additionally, they do not offer velocity and acceleration references, leaving the controller responsible for generating these signals. Conversely, when considering multi-rotor models [22], solutions depend on information exchange among agents, posing difficulties in environments with electromagnetic interference, such as in wind turbine inspection tasks [23].

Recently, several studies have explored planning and coordination techniques that incorporate advanced specifications and temporal goals [24, 25, 26]. However, these problems often prove challenging due to their nonlinear, non-convex optimization max-min nature. Control Barrier Functions (CBFs) [24] offer potential for finding robust and computationally efficient trajectories, though they are limited to simpler scenarios and lack soundness and completeness for the STL syntax. For example, if two robots need to visit distinct regions within overlapping time windows, CBFs may lead to infeasible problems [27]. Automata-based methods [25] can reduce complexity by assigning specifications to individual agents. However, constructing the required transition system is a significant burden. Sampling-based techniques [26], encode STL specifications as linear and boolean constraints, offering a solution to the challenge of finding optimal solutions. However, these techniques typically assume that each robot is specialized in specific skills. Most of these methods end up with a dynamic programming formulation, which suffer from dependency on the initial solution [28].

### 1.2. Contributions

In this paper, we introduce a novel approach to multi-UAV task assignment and trajectory generation for collaborative inspection missions, with wind turbine inspection serving as our case study. Our method utilizes STL as a specification language to express the mission's objectives and time constraints. These objectives include collision avoidance between the UAVs and the environment, as well as the completion of time-bound inspection tasks, such as reaching specified target areas and en-

sureing comprehensive coverage of the infrastructure. Addressing these requirements results in a complex nonlinear, non-convex max-min optimization problem typically solved using dynamic programming. However, finding an optimal solution in a reasonable time frame can be challenging due to the solvers’ tendency to get stuck in local optima based on the initial guess [28]. To tackle this challenge, we propose a two-step hierarchical approach that initially simplifies the mission into a Mixed-Integer Linear Programming planning problem formulated on a subset of the inspection objectives and subsequently seeds the global STL optimizer. The approach extends our prior work [29] by:

- Addressing collaborative inspection mission complexities and computing dynamically feasible trajectories with diverse time bounds and vehicle constraints (Section 4.1).
- Introducing a new method for computing the initial guess solution (Section 4.2) considering heterogeneous constraints on vehicle velocity and acceleration.
- Incorporating an event-triggered replanner (Section 4.3) to modify the planned trajectory in case of disturbances or unexpected events, ensuring the new trajectory aligns with the previous optimal solution by compensating for lost time.
- Including a weighted generalization of the STL robust semantics [30] to capture user preferences and address issues arising from conflicting tasks when mission accomplishment cannot be achieved (Section 4.4).
- Assessing the method’s overall performance through MATLAB simulations to evaluate its effectiveness in meeting mission specifications and the efficiency of incorporating the proposed initialization procedure (Section 5). Additionally, validation is conducted via Gazebo simulations and field experiments in a mock-up setting (Section 5.4).

The advantages our approach are twofold. Firstly, we employ an MILP planning formulation that operates on a simplified version of the problem, eliminating the need for linear or linearizable constraints and system dynamics, thereby facilitating the search for a global solution. Secondly, we utilize a concise and unambiguous STL formulation, enabling end-users to specify drone behavior in easily understandable terms, such as target areas to inspect and infrastructure zones to cover, while considering for explicit time requirements. In contrast to existing solutions [24, 25, 26], our approach utilizes a smooth robustness function to incorporate a quantitative robust semantics, a feature not achievable with automata-based methods [25]. Furthermore, our approach addresses the issue of overestimating the dimensions of the robot and obstacles, which can lead to more conservative maneuvers with CBFs [24].

Also, one significant advantage of our approach using STL over point-to-point navigation and vehicle-routing problem formulations, as listed in Section 1.1, is its ability to encapsulate intricate temporal constraints and requirements in a formal and

expressive manner. By integrating natural language commands, temporal and Boolean operators, and task and motion planning, we can devise trajectories that fulfill mission objectives while considering the dynamics and physical constraints of the multi-rotor systems. Moreover, the proposed method allows for handling coordination among multiple drones to ensure the completion of all required inspections without specifying the task (the paper presents a general method applied to a particular case scenario) that a single vehicle must accomplish. Existing methods for planning from STL specifications include abstraction-based methods, mixed-integer optimization, and nonlinear optimization. Abstraction-based methods [31] construct a discrete abstraction of the continuous state space, but suffer from scalability issues. Mixed-integer optimization methods [32, 33] provide soundness and completeness (see Section 3.4) guarantees but become intractable with longer planning horizons. Nonlinear optimization approaches [34, 35] offer increased generality and scalability but are prone to local optima based on initial guesses. Our two-step hierarchical approach addresses these challenges by initially simplifying the mission into a MILP planning problem and then seeding the global STL optimizer.

## 2. Problem Description

This paper explores the problem of utilizing a fleet of multi-rotor UAVs to collaboratively perform inspection missions, with a focus on wind turbine inspection as a case study. This entails capturing videos and pictures of the wind turbine and its surroundings to conduct a preliminary remote evaluation.

The inspection procedure is divided into two distinct tasks: pylon inspection and blade inspection. The *pylon inspection* involves visiting designated areas of interest, modeled as 3D spaces, to assess the structural integrity of mechanical components, such as the nacelle and rotor shaft. The objective of this inspection is to identify potential hazards, like corrosion or damage, that may affect the stability and safety of the pylon. In contrast, the *blade inspection* requires obtaining scans that cover the entire surface of the blade. This need arises due to external factors, such as wind, rain, hail, or bird strikes, that can cause cracks or coating damage, potentially impacting the blades’ performance. To ensure inspection accuracy, the UAVs must maintain a specific distance from the blade and move at a controlled speed to avoid blur effects.

The time needed to complete both inspections may differ based on the wind turbine’s size, which should be considered during the planning phase along with the vehicle’s dynamical constraints. The UAVs in this scenario are assumed to be multi-rotors, primarily quadrotors, with heterogeneous velocity and acceleration limitations. The objective is to plan trajectories for the UAVs to efficiently complete the inspection mission while also satisfying the aforementioned constraints and maintaining a safe distance from obstacles and other UAVs in the environment. It is assumed that a map of the environment, including a polyhedral representation of obstacles like the pylon and blades, is available prior to the inspection mission.

### 3. Preliminaries

This section introduces the fundamental concepts required to grasp the contributions of this paper. We will provide a brief overview of the key system variables, Signal Temporal Logic, and its robustness, smooth robustness approximations, the STL weighted generalization, and the STL motion planning framework upon which the current contribution is built. To enhance readability, a summary of the notation is also provided in Table 1.

#### 3.1. System definition

Let us consider a discrete-time dynamical model of a drone represented in the general form  $x_{k+1} = f(x_k, u_k)$ , where  $x_{k+1}, x_k \in X \subset \mathbb{R}^n$  are the next and current states of the system, respectively, and  $u_k \in \mathcal{U} \subset \mathbb{R}^m$  is the control input. Let us also assume  $f: X \times \mathcal{U} \rightarrow X$  is differentiable in both arguments. Therefore, given an initial state  $x_0 \in X_0 \subset \mathbb{R}^n$  and a time vector  $\mathbf{t} = (t_0, \dots, t_N)^\top \in \mathbb{R}^{N+1}$ , with  $N \in \mathbb{N}_{>0}$  being the number of samples that describe the evolution of the system with sampling period  $T_s \in \mathbb{R}_{>0}$ , we can define the finite control input sequence  $\mathbf{u} = (u_0, \dots, u_{N-1})^\top \in \mathbb{R}^N$  as the input for the system to attain the unique sequence of states  $\mathbf{x} = (x_0, \dots, x_N)^\top \in \mathbb{R}^{N+1}$ . Let us also introduce the notation  $\mathcal{F}_W$  and  $\mathcal{F}_B$  to denote the world frame and body frame reference systems, respectively.

Hence, we can define the state sequence  $\mathbf{x}$  and control input sequence  $\mathbf{u}$  for the  $d$ -th multi-rotor UAV as  ${}^d\mathbf{x} = ({}^d\mathbf{p}^{(1)}, {}^d\mathbf{v}^{(1)}, {}^d\mathbf{p}^{(2)}, {}^d\mathbf{v}^{(2)}, {}^d\mathbf{p}^{(3)}, {}^d\mathbf{v}^{(3)})^\top$  and  ${}^d\mathbf{u} = ({}^d\mathbf{a}^{(1)}, {}^d\mathbf{a}^{(2)}, {}^d\mathbf{a}^{(3)})^\top$ , with  ${}^d\mathbf{p}^{(j)}$ ,  ${}^d\mathbf{v}^{(j)}$ , and  ${}^d\mathbf{a}^{(j)}$  representing vehicle's position, velocity, and acceleration sequences along the  $j$ -axis of the world frame  $\mathcal{F}_W$ , respectively, with  $j = \{1, 2, 3\}$ . Finally, the  $k$ -th elements of  ${}^d\mathbf{p}^{(j)}$ ,  ${}^d\mathbf{v}^{(j)}$ ,  ${}^d\mathbf{a}^{(j)}$ , and  $\mathbf{t}$ , are denoted as  ${}^d p_k^{(j)}$ ,  ${}^d v_k^{(j)}$ ,  ${}^d a_k^{(j)}$ , and  $t_k$ .

The model  $x_{k+1} = f(x_k, u_k)$  considered for the drone is the one provided in [29], where the motion primitives are encoded with splines, and here simply denoted, for each  $j$ -axis, as  $({}^d p_{k+1}^{(j)}, {}^d v_{k+1}^{(j)}, {}^d a_{k+1}^{(j)})^\top = {}^d\mathbf{S}^{(j)}({}^d p_k^{(j)}, {}^d v_k^{(j)}, {}^d a_k^{(j)})$ . Next, we will use a label in the upper left to indicate a specific drone to which the dynamic model refers, while we will not use labels to indicate the vector stack of all drone variables.

#### 3.2. Signal temporal logic

STL concisely and unambiguously describes real-valued signal temporal behavior [12]. Unlike most used planning algorithms [36], STL encapsulates all mission specifications in a single formula  $\varphi$ . For example, the statement ‘‘visit regions A and B every 10s, while always stay at least 1 m away from region C’’ can be expressed as a single STL formula,  $\varphi$ . The syntax and semantics of STL are detailed in [12, 13], but is omitted here for brevity.

In brief, an STL formula  $\varphi$  is constructed from a set of predicates  $p_i$ , where  $i \in \mathbb{N}_0$ , representing atomic prepositions like belonging to a region or comparisons of real values. Formally, let  $M = \{\mu_1, \dots, \mu_L\}$  be a set of real-valued functions of the state,  $\mu_i: X \rightarrow \mathbb{R}$ , and the corresponding set  $AP := \{p_1, \dots, p_L\}$  of predicates. Each predicate defines a set over the system state space, specifically  $p_i$  defines  $\{x \in X \mid \mu_i(x) \geq 0\}$ .

Table 1: Notation–System variables, general symbols, and reference frames.

$\mathcal{F}_W, \mathcal{F}_B$	world and body reference frames
$N, T_s, \mathbf{t}, t_k, \delta$	number of samples, sampling period, time vector and its $k$ -th element, members of the UAV fleet
${}^d\mathbf{x}, {}^d\mathbf{u}$	state and control input sequences of the $d$ -th UAV
$M, AP$	Set of real-valued functions and the corresponding predicates
${}^d\mathbf{p}, {}^d\mathbf{v}, {}^d\mathbf{a}, {}^d\psi,$ ${}^d\bullet_k$	position, velocity, acceleration, and heading of the $d$ -th UAV in $\mathcal{F}_W$ and their $k$ -th elements
${}^d\mathbf{S}^{(j)}$	splines encoding the drone motion primitives
$\varphi, I, p_i, \mu_i, \lambda$	STL formula, generic time interval, $i$ -th predicate and its real-valued function, tunable parameter for $\tilde{\rho}_\varphi(\mathbf{x})$
$\neg, \wedge, \vee, \implies$	negation, conjunction, disjunction, and implication Boolean operators
$\diamond, \square, \circ, \mathcal{U}$	eventually, always, next and until temporal operators
$\rho_\varphi(\mathbf{x}, t_k),$ $\tilde{\rho}_\varphi(\mathbf{x}, t_k)$	robustness and smooth robustness values of the STL formula $\varphi$
${}^d\underline{v}^{(j)}, {}^d\overline{v}^{(j)},$ ${}^d\underline{a}^{(j)}, {}^d\overline{a}^{(j)}$	lower and upper limits for the velocity and acceleration of the $d$ -th UAV along the $j$ -axis
${}^d\varphi_{\text{ws}}, {}^d\varphi_{\text{obs}}, {}^d\varphi_{\text{dis}}$	STL safety requirements
${}^d\varphi_{\text{tr}}, {}^d\varphi_{\text{bla}}$	STL task requirements
${}^d\varphi_{\text{hm}}$	STL mission completion requirement
$T_N, T_{\text{ins}}, T_{\text{bla}}$	mission duration, pylon and blade coverage time intervals
$N_{\text{obs}}, N_{\text{tr}}, N_{\text{bla}}$	number of obstacles, number target areas, and number of blade sides to cover
$\underline{p}_\bullet^{(j)}, \overline{p}_\bullet^{(j)}$	generic vertices of the rectangular regions defining safety, task, and mission requirements
$\Gamma_{\text{dis}}, \Gamma_{\text{bla}}, \varepsilon, \zeta$	mutual distance threshold, minimum required distance away from the blade, maneuverability and safety margins
$\mathcal{G}, \mathcal{V}, \mathcal{E}, \mathcal{W}, \mathcal{D},$ $\mathcal{T}, \mathcal{O}$	graph, set of vertices, graph edges and weights, set of drones, set of target areas and blade extreme points, and set of depots
$\tau, o_\bullet, e_{ij}, w_{ij}, z_{ij}$	cardinality of $\mathcal{T}$ , generic element of $\mathcal{O}$ , $\mathcal{E}$ and $\mathcal{W}$ , integer variable for the MILP solution
$\mathcal{N}_i^{\text{in}}, \mathcal{N}_i^{\text{out}}$	in-neighborhood and out-neighborhood sets of nodes
${}^d\underline{v}^{(j)}, {}^d\overline{v}^{(j)},$ ${}^d\underline{a}^{(j)}, {}^d\overline{a}^{(j)}$	lower and upper limits for the revised velocity and acceleration of the $d$ -th UAV along the $j$ -axis
$\bar{\mathbf{t}}, E, \eta, \bar{t}_k, \hat{t}_k$	event-trigger time vector and its dimension, event threshold value, generic entry of $\bar{\mathbf{t}}$ , and time instance associated with the next task
$\omega\rho_\varphi(\omega, \mathbf{x}, t_k), \omega$	weighted version of the STL robustness score and corresponding weight vector
${}^d T_c, {}^d \omega_c$	thrust and angular velocities of the $d$ -th UAV

STL's grammar uses temporal operators, such as *until* ( $\mathcal{U}$ ), *always* ( $\square$ ), *eventually* ( $\diamond$ ), and *next* ( $\circ$ ), and logical operators like *and* ( $\wedge$ ), *or* ( $\vee$ ), *negation* ( $\neg$ ), and *implication* ( $\implies$ ), that act on atomic propositions over a non-singleton interval  $I \subset \mathbb{R}$ . Thus, an STL formula  $\varphi$  is built recursively from the predicates  $p_i$  and using the grammar, as:

$$\varphi := \top | \neg \varphi | \varphi_1 \vee \varphi_2 | \varphi_1 \wedge \varphi_2 | \square_I \varphi | \diamond_I \varphi | \circ_I \varphi | \varphi_1 \mathcal{U}_I \varphi_2,$$

where  $\varphi_1$  and  $\varphi_2$  are STL formulae. These propositions are simple *true* ( $\top$ ) or *false* ( $\perp$ ) statements, such as belonging to a particular region or comparing real values. An STL formula  $\varphi$  is valid if it evaluates to *true* ( $\top$ ) and invalid ( $\perp$ ) otherwise. For example,  $\varphi_1 \mathcal{U}_I \varphi_2$  requires that  $\varphi_2$  holds within the time interval  $I$  and that  $\varphi_1$  holds uninterrupted until that point.

If a formula  $\varphi$  is *time-bounded*, it contains no unbounded operator. The bound can be interpreted as the horizon of the future predicted system trajectory  $\mathbf{x}$  that is needed to calculate the satisfaction of  $\varphi$ . Generally, to evaluate whether such a formula  $\varphi$  holds on a given trajectory, only a finite-length prefix of that trajectory is needed [37, 38]. In this paper, we will only consider bounded time intervals  $I$ .

### 3.3. Robust signal temporal logic

Uncertainties and unforeseen events can affect the satisfaction of an STL formula  $\varphi$  (see Section 3.2). *Robust semantics* for STL formulae [12, 13, 39] account for these factors by ensuring a margin of satisfaction, measuring how well (poorly) a given specification is satisfied. The *robustness* metric,  $\rho$ , guides the optimization process in finding the best feasible solution to meet mission requirements. It can be formally defined using the following recursive formulae:

$$\begin{aligned} \rho_{p_i}(\mathbf{x}, t_k) &= \mu_i(x(t_k)), \\ \rho_{\neg \varphi}(\mathbf{x}, t_k) &= -\rho_{\varphi}(\mathbf{x}, t_k), \\ \rho_{\varphi_1 \wedge \varphi_2}(\mathbf{x}, t_k) &= \min(\rho_{\varphi_1}(\mathbf{x}, t_k), \rho_{\varphi_2}(\mathbf{x}, t_k)), \\ \rho_{\varphi_1 \vee \varphi_2}(\mathbf{x}, t_k) &= \max(\rho_{\varphi_1}(\mathbf{x}, t_k), \rho_{\varphi_2}(\mathbf{x}, t_k)), \\ \rho_{\square_I \varphi}(\mathbf{x}, t_k) &= \min_{t'_k \in [t_k, t_k+I]} \rho_{\varphi}(\mathbf{x}, t'_k), \\ \rho_{\diamond_I \varphi}(\mathbf{x}, t_k) &= \max_{t'_k \in [t_k, t_k+I]} \rho_{\varphi}(\mathbf{x}, t'_k), \\ \rho_{\circ_I \varphi}(\mathbf{x}, t_k) &= \rho_{\varphi}(\mathbf{x}, t'_k), \text{ with } t'_k \in [t_k, t_k+I], \\ \rho_{\varphi_1 \mathcal{U}_I \varphi_2}(\mathbf{x}, t_k) &= \max_{t'_k \in [t_k, t_k+I]} \left( \min(\rho_{\varphi_2}(\mathbf{x}, t'_k)), \right. \\ &\quad \left. \min_{t''_k \in [t_k, t'_k]} (\rho_{\varphi_1}(\mathbf{x}, t''_k)) \right), \end{aligned}$$

where  $t_k + I$  represents the Minkowski sum of scalar  $t_k$  and time interval  $I$ . These formulae, as said, are recursively defined from *predicates*  $p_i$  and their corresponding real-valued function  $\mu_i(x(t_k))$ , which are true if their robustness value is greater than zero and false otherwise.

The overall behavior of the formula is logical, becoming false if any predicates within it are false. In simpler terms, the STL formula  $\varphi_1 \vee \varphi_2$  is satisfied if  $\varphi_1$  or  $\varphi_2$  is true. Evaluation proceeds by applying logical and temporal operators (such as *always*, *eventually*, *conjunction*, etc.) from the innermost to the outermost part of the formula. For example, this could entail conditions like being inside a target region or

outside an obstacle region, each defined by a specific set of predicates. More comprehensive explanations can be found in references [12, 13, 39]. In this context, we say that  $\mathbf{x}$  satisfies the STL formula  $\varphi$  at time  $t_k$  (shortened as  $\mathbf{x}(t_k) \models \varphi$ ) if  $\rho_{\varphi}(\mathbf{x}, t_k) > 0$ , and violates it if  $\rho_{\varphi}(\mathbf{x}, t_k) \leq 0$ . Furthermore, the value of  $\rho_{\varphi}(\mathbf{x}, t_k)$  represents “how well” the formula is satisfied (if  $\rho_{\varphi}(\mathbf{x}, t_k) > 0$ ) or “how much” is violated (if  $\rho_{\varphi}(\mathbf{x}, t_k) \leq 0$ ), implicitly introducing a robustness criterion.

Hence, we compute control inputs  $\mathbf{u}$ , maximizing robustness  $\rho_{\varphi}(\mathbf{x}, t_k)$  over finite state  $\mathbf{x}$  and input sequences  $\mathbf{u}$ , with  $\mathbf{u}$  and  $\mathbf{x}$  obeying to system dynamics. The optimal sequence for input and states is denoted as  $\mathbf{u}^*$  and  $\mathbf{x}^*$ . A larger  $\rho_{\varphi}(\mathbf{x}^*, t_k)$  signifies a more resilient system behavior against disturbances, allowing the system to withstand greater values of the latter without violating the STL specification. To simplify notation, we will use  $\rho_{\varphi}(\mathbf{x})$  instead of  $\rho_{\varphi}(\mathbf{x}, 0)$  when  $t_k = 0$ .

### 3.4. Smooth approximation

The calculation of  $\rho_{\varphi}(\mathbf{x})$  incorporates non-differentiable functions like min and max. To tackle the computational challenges linked with these non-differentiable functions, it is advantageous to utilize a smooth approximation, represented as  $\tilde{\rho}_{\varphi}(\mathbf{x})$ , of the robustness function  $\rho_{\varphi}(\mathbf{x})$ . This smoothed approximation offers a more manageable and computationally efficient solution. Hence, let  $\lambda \in \mathbb{R}_{>0}$  be a tunable parameter. The smooth approximation of the min and max operators with  $\beta$  predicate arguments is [40]:

$$\begin{aligned} \max(\rho_{\varphi_1}, \dots, \rho_{\varphi_{\beta}}) &\approx \frac{\sum_{i=1}^{\beta} \rho_{\varphi_i} e^{\lambda \rho_{\varphi_i}}}{\sum_{i=1}^{\beta} e^{\lambda \rho_{\varphi_i}}}, \\ \min(\rho_{\varphi_1}, \dots, \rho_{\varphi_{\beta}}) &\approx -\frac{1}{\lambda} \log \left( \sum_{i=1}^{\beta} e^{-\lambda \rho_{\varphi_i}} \right). \end{aligned}$$

Compared to our prior work [29], we adopt an *asymptotically complete* and *smooth everywhere* approximation, which resembles the widely recognized Log-Sum-Exponential approximation [13]. The proposed approximation does not overestimate the max operator and, as a result, is sound. *Soundness* indicates that a suitable sequence  $\mathbf{u}^*$  with strictly positive smooth robustness ( $\tilde{\rho}_{\varphi}(\mathbf{x}) > 0$ ) satisfies the specification  $\varphi$ , whereas a sequence  $\mathbf{u}^*$  with strictly negative smooth robustness ( $\tilde{\rho}_{\varphi}(\mathbf{x}) < 0$ ) violates it. The term *asymptotical completeness* implies that the resulting approximation  $\tilde{\rho}_{\varphi}(\mathbf{x})$  for the final robustness formula can approach the true robustness  $\rho_{\varphi}(\mathbf{x})$  arbitrarily closely as  $\lambda$  tends towards infinity ( $\lambda \rightarrow \infty$ ). Furthermore, the proposed approximation is *smooth everywhere*, possessing infinite differentiability, thus making it viable to use gradient-based optimization algorithms to ascertain a solution [40]. By increasing  $\lambda$ , the approximation can better reflect the true robustness (see Section 3.3).

### 3.5. Weighted signal temporal logic

In many applications, high-level temporal logic specifications may include obligatory or alternative sub-specifications or timings with varying importance or priorities. Traditional

STL lacks the expressivity to specify these preferences. For example, consider  $\varphi = \diamond_I(\mathbf{x} > 0)$ , which is satisfied if  $\mathbf{x}$  becomes greater than 0 within the time interval  $I$ . However, satisfaction at earlier times within this deadline may be more desirable.

Assisting importance and priorities becomes crucial, especially when a formula contains conflicting obligatory subformulae. Therefore, an extension of STL, known as weighted-STL (wSTL) [30, 41, 42], can be employed to enable user preferences such as priorities and importance. The syntax of wSTL is an extension of the STL syntax, and is defined as:

$$\varphi := \top | \perp | \neg \varphi | \bigwedge_{i=1}^N \omega \varphi_i | \bigvee_{i=1}^N \omega \varphi_i | \square_I \omega \varphi | \diamond_I \omega \varphi | \bigcirc_I \omega \varphi | \mathcal{U}_I \omega \varphi_1 \mathcal{U}_I \omega \varphi_2,$$

where the logical *true* ( $\top$ ) and *false* ( $\perp$ ) values, the predicate  $p$ , and all the Boolean and temporal operators have the same interpretation as in STL (as discussed in Section 3.2). The weight  $\omega = [\omega_i]_{i=1}^N \in \mathbb{R}_{>0}^N$  assigns a positive weight  $\omega_i$  to each subformula  $i$  of the  $N$  sub-formulae of the Boolean operators, and  $\omega = [\omega_{t_k}]_{t_k \in I} \in \mathbb{R}_{>0}^{|I|}$  assigns a positive weight  $\omega_{t_k}$  to time  $t_k$  in the interval  $I$  of the temporal operators and  $|I|$  denotes cardinality of  $I$ .

The weights  $\omega$  capture the *importance* of obligatory specifications for conjunctions ( $\wedge$ ) or *priorities* of alternatives for disjunctions ( $\vee$ ). Similarly,  $\omega$  captures the *importance* of satisfaction times for temporal *always* ( $\square$ ) or *priorities* of satisfaction times for temporal *eventually* ( $\diamond$ ) over the interval  $I$ . Throughout the paper, if the weight  $\omega$  associated with an operator (Boolean or temporal) in a wSTL formula  $\omega \varphi$  is constant 1, we drop it from the notation. Thus, STL formulae are wSTL formulae with all weights equal to 1.

Given a wSTL specification  $\omega \varphi$ , the weighted robustness score  ${}^\omega \rho_\varphi(\omega, \mathbf{x}, t_k)$  is recursively defined as:

$$\begin{aligned} {}^\omega \rho_{p_i}(\omega, \mathbf{x}, t_k) &= \mu_i(\omega, x(t_k)), \\ {}^\omega \rho_{\neg \varphi}(\omega, \mathbf{x}, t_k) &= -\rho_\varphi(\omega, \mathbf{x}, t_k), \\ {}^\omega \rho_{\varphi_1 \wedge \varphi_2}(\omega, \mathbf{x}, t_k) &= \min(\rho_{\varphi_1}(\omega, \mathbf{x}, t_k), \rho_{\varphi_2}(\omega, \mathbf{x}, t_k)), \\ {}^\omega \rho_{\varphi_1 \vee \varphi_2}(\omega, \mathbf{x}, t_k) &= \max(\rho_{\varphi_1}(\omega, \mathbf{x}, t_k), \rho_{\varphi_2}(\omega, \mathbf{x}, t_k)), \\ {}^\omega \rho_{\square_I \varphi}(\omega, \mathbf{x}, t_k) &= \min_{t'_k \in [t_k, t_k+I]} \rho_\varphi(\omega, \mathbf{x}, t'_k), \\ {}^\omega \rho_{\diamond_I \varphi}(\omega, \mathbf{x}, t_k) &= \max_{t'_k \in [t_k, t_k+I]} \rho_\varphi(\omega, \mathbf{x}, t'_k), \\ {}^\omega \rho_{\bigcirc_I \varphi}(\omega, \mathbf{x}, t_k) &= \rho_\varphi(\omega, \mathbf{x}, t'_k), \text{ with } t'_k \in [t_k, t_k+I], \\ {}^\omega \rho_{\varphi_1 \mathcal{U}_I \varphi_2}(\omega, \mathbf{x}, t_k) &= \max_{t'_k \in [t_k, t_k+I]} \left( \min(\rho_{\varphi_2}(\omega, \mathbf{x}, t'_k), \right. \\ &\quad \left. \min_{t''_k \in [t_k, t'_k]} (\rho_{\varphi_1}(\omega, \mathbf{x}, t''_k)) \right), \end{aligned}$$

where, as discussed for STL (see Section 3.2),  $t_k + I$  represents the Minkowski sum of scalar  $t_k$  and time interval  $I$ . These formulae consist of predicates,  $p_i$ , along with their corresponding weighted real-valued function  $\mu_i(\omega, \mathbf{x}, t_k)$ , which are true if their robustness value is greater than zero and false otherwise.

### 3.6. STL motion planner

By encoding the mission specifications detailed in Section 2 as an STL formula  $\varphi$  and replacing its robustness  $\rho_\varphi(\mathbf{x}, t_k)$  with

the smooth approximation  $\tilde{\rho}_\varphi(\mathbf{x}, t_k)$  (see Section 3.4), the problem of generating trajectories for the multi-rotors can be formulated as [29]:

$$\begin{aligned} &\underset{d \in \mathcal{D}}{\text{maximize}} \quad \tilde{\rho}_\varphi(\mathbf{p}^{(j)}, \mathbf{v}^{(j)}) \\ &\text{s.t.} \quad d_{\underline{v}}^{(j)} \leq d_{v_k}^{(j)} \leq d_{\overline{v}}^{(j)}, \\ &\quad d_{\underline{a}}^{(j)} \leq d_{a_k}^{(j)} \leq d_{\overline{a}}^{(j)}, \\ &\quad \tilde{\rho}_\varphi(d_{\mathbf{p}}^{(j)}, d_{\mathbf{v}}^{(j)}) \geq \zeta, \\ &\quad d_{\mathbf{S}}^{(j)}, \forall k = \{0, 1, \dots, N-1\} \end{aligned} \quad (1)$$

where  $d_{\underline{v}}^{(j)}$  and  $d_{\underline{a}}^{(j)}$  represent the lower limits of velocity and acceleration, respectively, while  $d_{\overline{v}}^{(j)}$  and  $d_{\overline{a}}^{(j)}$  denote their respective upper limits, for drone  $d$  along each  $j$ -axis of the world frame  $\mathcal{F}_W$ . Here,  $\mathcal{D}$  denotes the set of drones, and  $\mathbf{p}^{(j)}$  and  $\mathbf{v}^{(j)}$  concatenate the position and velocity of all drones. The minimum robustness threshold, denoted as  $\tilde{\rho}_\varphi(\mathbf{p}^{(j)}, \mathbf{v}^{(j)}) \geq \zeta$ , acts as a safety buffer for ensuring the satisfaction of the STL formula  $\varphi$  even in the presence of disturbances. As illustrated in [21], disturbances below magnitude  $\zeta$  do not result in formula violations. The specific value of  $\zeta$  can be determined such that  $|\rho_\varphi(\mathbf{x}) - \tilde{\rho}_\varphi(\mathbf{x})| \leq \zeta$ . Moreover, the shorthand notation  $d_{\mathbf{S}}^{(j)}$  refers to the motion primitives satisfying the dynamics of drone  $d$  along each  $j$ -axis, as explained in Section 3.1. For simplicity, we assume symmetric velocity and acceleration limits.

## 4. Problem Solution

In this section, we introduce a method to generate trajectories for a fleet of  $\delta$  multi-rotor UAVs belonging to set  $\mathcal{D}$ , i.e.,  $\delta = |\mathcal{D}|$ . The mission specifications have been described in Section 2 and are expressed as an STL formula  $\varphi$  (Section 4.1). The motion planner is a nonlinear, non-convex max-min optimization problem that accounts for mission specifications and physical constraints. However, finding optimal solutions within a reasonable time frame can be challenging because solvers easily get stuck in local optima depending on the initial guess [28]. To overcome this challenge, we propose a two-step hierarchical approach that simplifies the mission into an MILP planning problem (Section 4.2) before seeding the global STL optimizer (see Section 3.6). Our framework also includes an event-triggered replanner to cope with disturbances or unforeseen events (Section 4.3), as well as user preferences to handle potential issues arising from conflicting tasks (Section 4.4).

### 4.1. Specification mapping

This section aims to design the mission specifications for the problem described in Section 2 and derive the corresponding STL formula,  $\varphi$ . While a wind turbine inspection could theoretically be performed safely with just one UAV, practical considerations necessitate cooperative execution by multiple UAVs. Factors such as efficiency, coverage area, time-saving, enhanced safety, and redundancy in case of unexpected events can benefit from the involvement of multiple drones to

ensure mission success. Therefore, the wind turbine inspection is encoded as a cooperative execution by the UAVs to meet safety and task requirements.

The *safety requirements* must be met throughout the entire operation time  $T_N$ . Hence, the UAVs must adhere to three specifications: remaining within the workspace ( ${}^d\varphi_{ws}$ ), avoiding obstacles to prevent collisions ( ${}^d\varphi_{obs}$ ), and keeping a safe distance from other UAVs ( ${}^d\varphi_{dis}$ ). The *task requirements* include pylon and blade inspections. To inspect the pylon ( ${}^d\varphi_{tr}$ ), the UAV must visit all target areas once and remain there for at least an inspection time  $T_{ins}$ . In order to inspect the blade ( ${}^d\varphi_{bla}$ ), the UAV must approach the leading edge, maintain a certain distance from the blade surface while covering it at a limited speed, and reach the rotor shaft before turning to inspect the other side of the blade surface. The coverage task must last at least  $T_{bla}$ .

Notably, for the case scenario, we assume that each UAV can only cover some blades or targets, reflecting a scenario where some drones may be equipped with specialized instruments or sensors, resulting in their heterogeneous physical capabilities. This assumption necessitates coordination among multiple UAVs to ensure completion of all required inspections. Here, we do not specify which task (pylon or blade inspection) a single drone must accomplish. Instead, we allow the framework to choose the most convenient option, while also leaving open the possibility of accomplishing both tasks. Finally, each UAV must return to its initial position after completing its inspection operations ( ${}^d\varphi_{hm}$ ) and remain there thereafter.

All the above mission specifications can be represented in the STL formula:

$$\begin{aligned} \varphi = & \bigwedge_{d \in \mathcal{D}} \square_{[0, T_N]} ({}^d\varphi_{ws} \wedge {}^d\varphi_{obs} \wedge {}^d\varphi_{dis}) \wedge \\ & \bigwedge_{q=1}^{N_{tr}} \diamond_{[0, T_N - T_{ins}]} \bigvee_{d \in \mathcal{D}} \square_{[0, T_{ins}]} {}^d\varphi_{tr,q} \wedge \\ & \bigwedge_{q=1}^{N_{bla}} \diamond_{[0, T_N - T_{bla}]} \bigvee_{d \in \mathcal{D}} \square_{[0, T_{bla}]} {}^d\varphi_{bla,q} \wedge \\ & \bigwedge_{d \in \mathcal{D}} \diamond_{[1, T_N]} {}^d\varphi_{hm} \wedge \\ & \bigwedge_{d \in \mathcal{D}} \square_{[1, T_N - 1]} ({}^d\varphi_{hm} \implies \bigcirc_{[0, t_k + 1]} {}^d\varphi_{hm}). \end{aligned} \quad (2)$$

Here, the STL formula  $\varphi$  comprises six specifications ( ${}^d\varphi_{ws}$ ,  ${}^d\varphi_{obs}$ ,  ${}^d\varphi_{dis}$ ,  ${}^d\varphi_{tr}$ ,  ${}^d\varphi_{bla}$ , and  ${}^d\varphi_{hm}$ ) and three time intervals ( $T_N$ ,  $T_{ins}$ , and  $T_{bla}$ ). The following equations describe each of these specifications:

$${}^d\varphi_{ws} = \bigwedge_{j=1}^3 {}^d\mathbf{p}^{(j)} \in (\underline{p}_{ws}^{(j)}, \bar{p}_{ws}^{(j)}), \quad (3a)$$

$${}^d\varphi_{obs} = \bigwedge_{j=1}^3 \bigwedge_{q=1}^{N_{obs}} {}^d\mathbf{p}^{(j)} \notin (\underline{p}_{obs,q}^{(j)}, \bar{p}_{obs,q}^{(j)}), \quad (3b)$$

$${}^d\varphi_{hm} = \bigwedge_{j=1}^3 {}^d\mathbf{p}^{(j)} \in (\underline{p}_{hm}^{(j)}, \bar{p}_{hm}^{(j)}), \quad (3c)$$

$${}^d\varphi_{dis} = \bigwedge_{\{d,m\} \in \mathcal{D}, d \neq m} \|{}^d\mathbf{p} - {}^m\mathbf{p}\| \geq \Gamma_{dis}, \quad (3d)$$

$${}^d\varphi_{tr,q} = \bigwedge_{j=1}^3 {}^d\mathbf{p}^{(j)} \in (\underline{p}_{tr,q}^{(j)}, \bar{p}_{tr,q}^{(j)}), \quad (3e)$$

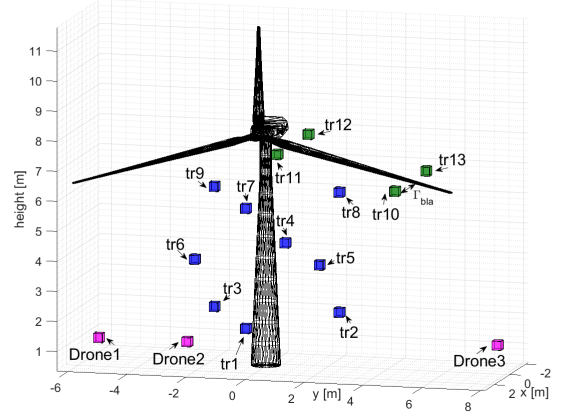


Figure 2: Wind turbine inspection scenario. Target areas and blade extreme points are represented in blue and green, respectively, while the drones' starting points are in magenta.

$$\begin{aligned} {}^d\varphi_{bla,q} = & \bigwedge_{j=1}^3 {}^d\mathbf{p}^{(j)} \in (\underline{p}_{bla,q}^{(j)}, \bar{p}_{bla,q}^{(j)}) \wedge \\ & \text{dist}_{bla,q}({}^d\mathbf{p}) \in (\Gamma_{bla} - \varepsilon, \Gamma_{bla} + \varepsilon), \end{aligned} \quad (3f)$$

where (3a) constrains the position of each UAV to remain within the designated workspace, with  $\underline{p}_{ws}^{(j)}$  and  $\bar{p}_{ws}^{(j)}$  denoting the working space limits. Obstacle avoidance and mission completion operation are defined in (3b) and (3c), respectively, where  $N_{obs}$  denotes the number of obstacles in the map. Rectangular regions with vertices denoted by  $\underline{p}_{obs,q}^{(j)}$ ,  $\underline{p}_{hm}^{(j)}$ ,  $\bar{p}_{obs,q}^{(j)}$ , and  $\bar{p}_{hm}^{(j)}$  define obstacle and drone's initial position areas. (3d) encodes the safety distance requirement, where  $\Gamma_{dis} \in \mathbb{R}_{>0}$  represents a threshold value for the drones mutual distance. Finally, inspecting the pylon and blade is done with (3e) and (3f), respectively, where  $N_{tr}$  and  $N_{bla}$  indicate the number of target areas and blade sides to cover, respectively. The vertices  $\underline{p}_{tr,q}^{(j)}$ ,  $\underline{p}_{bla,q}^{(j)}$ ,  $\bar{p}_{tr,q}^{(j)}$ , and  $\bar{p}_{bla,q}^{(j)}$  identify the target and covering areas, with the leading edge and rotor shaft serving as the extreme points for each side of the blade surface. The function  $\text{dist}_{bla,q}(\cdot)$  in (3f) calculates the Euclidean distance between the position of the UAV ( ${}^d\mathbf{p}$ ) and the corresponding point on blade  $q$ , determined by projecting the UAV's position orthogonally onto the surface of the blade. Its role in (3f) is to enforce a certain distance between the UAV and the blade surface, with  $\Gamma_{bla} \in \mathbb{R}_{>0}$  and  $\varepsilon \in \mathbb{R}_{>0}$  representing the minimum required distance and maneuverability margin, respectively. The latter enables the UAV to continue operating under challenging scenarios, such as collision risks, while meeting mission specifications. The *always* operators ( $\square$ ) are exploited to ensure the satisfaction of the minimum time requirements  $T_{bla}$  and  $T_{ins}$ . Figure 2 illustrates the scenario.

Utilizing the specifications outlined in (2), the optimization problem, as defined in Section 3.6, is formulated to determine feasible trajectories that maximize the smooth robustness  $\tilde{\rho}_\varphi(\mathbf{x})$  concerning the given mission specifications  $\varphi$ . To achieve this objective, it is essential to compute the robustness score for each predicate. The STL formula (2) consists of two types of predicates. The first type assesses whether the UAVs' position

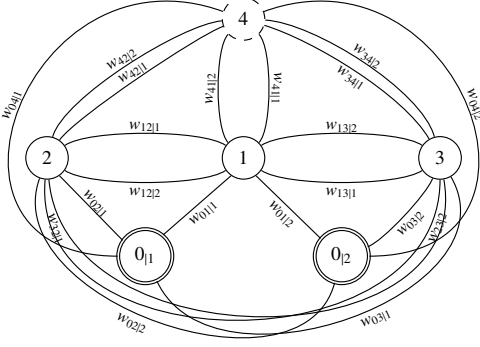


Figure 3: Instance of the graph  $G$  assuming three target areas (round solid nodes), one blade to cover (round dashed nodes), and two UAVs. Round double nodes are the depots. Notably, we assume that  $w_{ijld} = w_{jild}$ .

belongs or does not belong to a specific region, as illustrated in (3a), (3b), (3c), (3e), (3f). The second type evaluates the distance between UAVs, as indicated by the safety requirement in (3d). The robustness values are determined based on the Euclidean distance. For predicates belonging to the first group, a positive robustness signifies that the UAV lies within the designated region. The robustness increases as the minimum Euclidean distance to the boundaries of the region along the trajectory grows larger. However, in the case of (3b), the opposite holds true, where being within the obstacle region corresponds to a negative robustness. In the safety distance predicate (3d), the robustness is positive when the distance between UAVs exceeds the threshold  $\Gamma_{\text{dis}}$ . The robustness value increases as the minimum Euclidean distance between UAVs along the trajectory becomes larger.

#### 4.2. MILP encoding

An appropriate initial guess is crucial for obtaining optimal solutions for the STL motion planner within a reasonable time frame and avoiding the solver from getting trapped in local optima. To achieve this, the original wind turbine inspection problem is abstracted to create a simpler optimization problem with fewer constraints. The initial guess considers the mission requirements related to covering the blade surface and visiting all of the target areas ( ${}^d\varphi_{\text{bla}}$  and  ${}^d\varphi_{\text{tr}}$ ). However, the constraints related to obstacle avoidance, workspace, safety distance, and mission completion requirements ( ${}^d\varphi_{\text{obs}}$ ,  ${}^d\varphi_{\text{ws}}$ ,  ${}^d\varphi_{\text{dis}}$ , and  ${}^d\varphi_{\text{hm}}$ ) and mission time intervals ( $T_N$ ,  $T_{\text{ins}}$ , and  $T_{\text{bla}}$ ) are ignored to reduce complexity in the problem. To formulate the MILP planning problem, a graph-based representation is employed to connect the target areas, blade extreme points (leading edge and rotor shaft), and the vehicles' initial positions. To guarantee seamless coverage, the four blade extreme points (two on each side) are considered as a single node in the graph. The MILP assigns tasks to the vehicles and determines a navigation sequence for each UAV. This results in a type of VRP [36], which assigns tasks to vehicles and provides a navigation sequence for each UAV. Furthermore, we included the mission time in the optimization to distribute it equally among fleet members.

A directed weighted multigraph represented by the tuple  $G = (\mathcal{V}, \mathcal{E}, \mathcal{W}, \mathcal{D})$  is used to formulate the MILP, as shown

in Figure 3. The set of vertices  $\mathcal{V}$  is defined as  $\mathcal{T} \cup \mathcal{O}$ , where  $\mathcal{T}$  correspond to the target areas and the blade extreme points, and  $\mathcal{O} = \{o_1, \dots, o_\delta\}$  comprises the depots where each UAV is located at  $t_0$ . The cardinalities of the sets are denoted by  $|\mathcal{T}| = \tau$  and  $|\mathcal{O}| = \delta$ . The graph's edges and their weights are described by the sets  $\mathcal{E}$  and  $\mathcal{W}$ , respectively. The set  $\mathcal{D}$ , as said before, contains  $\delta$  available UAVs. The graph connectivity is such that all vertices in  $\mathcal{T}$  are fully connected to each other and to all vertices in the set  $\mathcal{O}$ , while the vertices in  $\mathcal{O}$  are unconnected. Specifically, for any  $\{i, j\} \in \mathcal{V}$  and  $d \in \mathcal{D}$ , the edge  $e_{ijld} \in \mathcal{E}$  connects vertices  $i$  and  $j$  using UAV  $d$ , while  $w_{ijld} \in \mathcal{W}$  denotes the corresponding weight. Notably, we assume that  $w_{ijld} = w_{jild}$  models the edge weights based on the UAVs' time of flight given their dynamic constraints, which are heterogeneous and denoted by  ${}^d\underline{v}^{(j)}$ ,  ${}^d\underline{a}^{(j)}$ ,  ${}^d\bar{v}^{(j)}$  and  ${}^d\bar{a}^{(j)}$ .

The UAV motion primitives  ${}^d\mathbf{S}^{(j)}$  (see Section 3.6) are employed to calculate the flight time for each UAV as it travels between target areas and blade extreme points, taking into account the physical limitations of the vehicles in terms of velocity ( ${}^d\underline{v}^{(j)}$  and  ${}^d\bar{v}^{(j)}$ ) and acceleration ( ${}^d\underline{a}^{(j)}$  and  ${}^d\bar{a}^{(j)}$ ). For brevity, further details for computing these times can be found elsewhere in [43]. The objective of the mission is to minimize the overall time.

To indicate the number of times an edge is selected in the MILP solution, we assign an integer variable  $z_{ijld}$ , where  $z_{ijld} \in \mathbb{N}$  for  $i \in \mathcal{O}$  and  $j \in \mathcal{T}$ , with  $d \in \mathcal{D}$ , to each edge  $e_{ijld} \in \mathcal{E}$ . We define an *in-neighborhood*, denoted by  $\mathcal{N}_i^{\text{in}}$  the set of nodes having an edge entering  $i$ , that is  $\mathcal{N}_i^{\text{in}} = \{j \in \mathcal{V} : (j, i) \in \mathcal{E}\}$ . Similarly, we define an *out-neighborhood* as the set of nodes having an entering edge which starts from  $i$ , that is  $\mathcal{N}_i^{\text{out}} = \{j \in \mathcal{V} : (i, j) \in \mathcal{E}\}$ . Using these variables, we propose to formulate the MILP planning problem as follows:

$$\text{minimize}_{z_{ijld}} \sum_{\{i,j\} \in \mathcal{V}, i \neq j, d \in \mathcal{D}} w_{ijld} z_{ijld} \quad (4a)$$

$$\text{s.t.} \sum_{i \in \mathcal{N}_j^{\text{in}}} z_{ijld} = \sum_{i \in \mathcal{N}_j^{\text{out}}} z_{ijld}, \quad \forall j \in \mathcal{T}, \quad \forall d \in \mathcal{D}, \quad (4b)$$

$$\sum_{d \in \mathcal{D}} \sum_{i \in \mathcal{N}_j^{\text{in}}} z_{ijld} \geq 1, \quad \forall j \in \mathcal{T}, \quad (4c)$$

$$\sum_{j \in \mathcal{N}_d^{\text{out}}} z_{o_d jld} = 1, \quad \forall d \in \mathcal{D}, \quad (4d)$$

$$\sum_{j \in \mathcal{N}_d^{\text{in}}} z_{j o_d ld} = 1, \quad \forall d \in \mathcal{D}. \quad (4e)$$

In the above formulae, (4a) is the objective function encoding the total flight time of the fleet of UAVs. Notice that (4a) implicitly minimizes the fact that a drone passes by the same edges multiple (useless) times. (4b) ensures that UAVs do not accumulate in an area they reach. (4c) ensures that all target areas and blade extreme points are visited at least once. (4d) and (4e) enforce the depot of each drone as its departure and arrival point. If disconnected subtours are provided in the MILP solution, we adopt the suboptimal decision of connecting them afterwards. Indeed, since the MILP solution is only used to seed the final STL optimizer, this choice is not a significant concern



and allows to save computational time with respect to (w.r.t.) enforcing subtours elimination constraints. Notice that (4) can be adapted to meet the mission requirements, including additional futures, such as capacity restrictions for delivery missions, and removing tight limitations, such as the need to start and end at the depot position. Additionally, heuristics and meta-heuristics techniques [15, 44] can be used to cope with the increasing number of constraints and variables.

After obtaining the optimal assignment from the MILP planning problem (4), the motion primitives described in [29] are used to generate dynamically feasible trajectories for each UAV in accordance with its optimal navigation sequence used to seed the STL optimizer.

### 4.3. Event-based replanner

Environmental uncertainties and disturbances can cause mismatches between planned and actual drone trajectories (e.g., wind gusts or technical fault). In such cases, it is beneficial for the planner to compute a new trajectory that minimizes deviation from the optimal offline solution. To address this challenge, we present an improved event-based replanner [29] that continuously adjusts the drone trajectory to make up for lost time until the original time requirements are met. The revised trajectories have less strict vehicle limitations ( $d_{\hat{v}}^{(j)}$ ,  $d_{\hat{a}}^{(j)}$ ,  $d_{\hat{v}}^{(j)}$ , and  $d_{\hat{a}}^{(j)}$ ) than the original plan ( $d_{\underline{v}}^{(j)}$ ,  $d_{\underline{a}}^{(j)}$ ,  $d_{\bar{v}}^{(j)}$ , and  $d_{\bar{a}}^{(j)}$ ), assuming that the original plan was scheduled with “conservative” constraints to avoid stresses on the actuators.

Let us introduce the revised velocity and acceleration constraints for each drone, characterized by their lower and upper limits, represented as  $d_{\hat{v}}^{(j)}$ ,  $d_{\hat{a}}^{(j)}$ ,  $d_{\hat{v}}^{(j)}$ , and  $d_{\hat{a}}^{(j)}$ . Here,  $d_{\hat{v}}^{(j)} < d_{\underline{v}}^{(j)}$ ,  $d_{\hat{a}}^{(j)} < d_{\underline{a}}^{(j)}$ ,  $d_{\hat{v}}^{(j)} > d_{\bar{v}}^{(j)}$ , and  $d_{\hat{a}}^{(j)} > d_{\bar{a}}^{(j)}$ . Let us also define the discrete time instants when events occur as  $\bar{\mathbf{t}} = (\bar{t}_0, \dots, \bar{t}_E)^\top \in \mathbb{R}^{E+1}$  and  $\bar{t}_k$  as a generic entry of  $\bar{\mathbf{t}}$ . Finally, let  ${}^d\tilde{\mathbf{p}}_k$  and  ${}^d\mathbf{p}_k^*$  represent the runtime and optimal positions of the  $d$ -th drone, respectively. Hence, we can define the event trigger condition for the planner as  $\|{}^d\tilde{\mathbf{p}}_k - {}^d\mathbf{p}_k^*\| > \eta$ , where  $\eta \in \mathbb{R}_{>0}$  is a threshold value set to achieve the best system behavior. Whenever this condition is met, a trigger is generated and the planner computes a new plan online using the motion planner in Sections 4.1 and 4.2 over the time interval  $[\bar{t}_k, \hat{t}_k]$ . Here,  $\hat{t}_k$  denotes the time instance associated with the next task assignment before the trigger was generated. Note that only the affected drone has its trajectory recomputed, while the other functioning drones follow their original plans.

It is noteworthy that the computational effort needed for replanning is considerably less compared to initial planning, as it deals with only one UAV and a reduced set of task regions (those yet to be visited). While exploring alternative strategies for replanning the routes of delayed drones may potentially yield better solutions in terms of robustness scores, it is crucial to highlight that our focus in this paper is on ensuring operational continuity, minimizing disruptions, and upholding safety in hazardous scenarios.

### 4.4. Attrition-aware planner

In collaborative missions, conflicting tasks can lead to an infeasible problem for the actual STL robustness score. For ex-

ample, a drone may be tasked with surveying areas that lack sufficient safety margins for the entire mission duration. To tackle this issue, incorporating user preferences into the planner formulation can be beneficial. The goal is to generate a feasible vehicle trajectory that prioritizes mission specifications, while ensuring their temporal satisfaction. A generalized robustness score approach for STL robust semantics [30, 41, 42] can be employed to formally capture requirements using weights. Rather than just focusing on boundary conditions, this allows for the assignment of time priorities when the specification is satisfied.

This addresses some limitations of the traditional robustness score, such as the inability to specify when a proposition should be preferentially satisfied, such as at the beginning or just before the end of the optimization. As an example, we consider the generalized robustness score for the logical operator *and* ( $\wedge$ ) (see Section 3.5):

$$\omega \rho_{\varphi_1 \wedge \varphi_2}(\omega, \mathbf{x}, t_k) = \min(\omega_1 \rho_{\varphi_1}(\mathbf{x}, t_k), \omega_2 \rho_{\varphi_2}(\mathbf{x}, t_k)),$$

where  $\omega \rho_{\varphi_1 \wedge \varphi_2}(\omega, \mathbf{x}, t_k)$  refers to the weighted version of the STL robustness formula  $\rho_{\varphi_1 \wedge \varphi_2}(\mathbf{x}, t_k)$ . The variable  $\omega = (\omega_1, \omega_2)^\top$ , with  $\{\omega_1, \omega_2\} \in \mathbb{R}_{>0}$ , represents the vector of weights applied to each predicate of the STL formula  $\varphi = \varphi_1 \wedge \varphi_2$ .

Thus, we can express the weighted generalization of the standard smooth robustness of the STL formula  $\varphi = \varphi_1 \wedge \varphi_2$ , denoted by  $\omega \tilde{\rho}_\varphi(\omega, \mathbf{x}, t_k)$ , as:

$$\omega \tilde{\rho}_{\varphi_1 \wedge \varphi_2}(\omega, \mathbf{x}, t_k) = \min_{i=\{1,2\}} \left\{ \left( \frac{1}{2} - \bar{\omega}_i \right) \text{sign}(\tilde{\rho}_{\varphi_i}) + \frac{1}{2} \right\} \tilde{\rho}_{\varphi_i},$$

where  $\bar{\omega}_i = \omega_i / (\mathbf{I}_2 \omega)$  are normalized weights, with  $\mathbf{I}_2 \in \mathbb{R}^{2 \times 2}$  being the identity matrix. The complete syntax and semantics of the generalized robustness score approach can be found in [30, 41, 42]. Due to space limitations, full details of the approach are not reported here.

Hence, we propose to formulate problem (1) by replacing the smooth approximation of the standard STL formula  $\tilde{\rho}_\varphi(\mathbf{x}, t_k)$  with its weighted version  $\omega \tilde{\rho}_\varphi(\omega, \mathbf{x}, t_k)$  as follows:

$$\begin{aligned} & \underset{d \in \mathcal{D}}{\text{maximize}} && \omega \tilde{\rho}_\varphi(\omega, \mathbf{p}, \mathbf{v}) \\ & \text{s.t.} && d_{\underline{v}}^{(j)} \leq d_{\mathbf{v}_k}^{(j)} \leq d_{\bar{v}}^{(j)}, \\ & && d_{\underline{a}}^{(j)} \leq d_{\mathbf{a}_k}^{(j)} \leq d_{\bar{a}}^{(j)}, \\ & && \omega \tilde{\rho}_\varphi(\omega, {}^d\mathbf{p}^{(j)}, {}^d\mathbf{v}^{(j)}) \geq \zeta, \\ & && {}^d\mathbf{S}^{(j)}, \forall k = \{0, 1, \dots, N-1\} \end{aligned} \quad (5)$$

In this way, we can ensure the satisfaction of collaborative missions while considering prioritized specifications to avoid issues that may arise from conflicting tasks during trajectory planning. For instance, if the original problem (1) is unable to find a solution due to conflicting tasks, adjusting the weights  $\omega$  of the specifications, such as prioritizing safety ( ${}^d\varphi_{\text{ws}}$ ,  ${}^d\varphi_{\text{obs}}$ ,  ${}^d\varphi_{\text{dis}}$ ) and mission completion requirements ( ${}^d\varphi_{\text{hm}}$ ) over task requirements ( ${}^d\varphi_{\text{tr}}$  and  ${}^d\varphi_{\text{bla}}$ ), could lead to safer trajectories. However, this comes at the expense of a reduction in the overall robustness score  $\rho_\varphi(\mathbf{x}, t_k)$ .

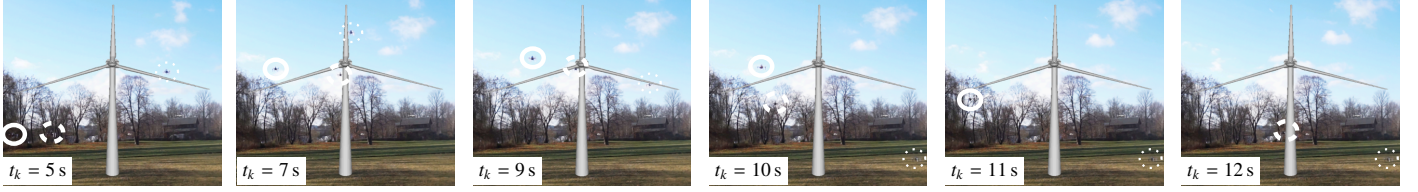


Figure 4: Experiment snapshots. Solid, dashed, and dotted circles indicate “drone1”, “drone2”, and “drone3”, respectively. The wind turbine was virtually projected within the UAVs’ camera frames to convey its proximity to the infrastructure.

## 5. Experimental Results

The proposed planning approach’s validity and effectiveness were assessed through MATLAB and Gazebo simulations, along with field experiments in a mock-up scenario. Initially, numerical simulations were performed in MATLAB without explicitly modeling actual vehicle dynamics and trajectory tracking controllers. This approach allowed us to evaluate the planning algorithm’s performance and gain valuable insights into its behavior. Subsequently, to further validate the generated trajectories and leverage the benefits of software-in-the-loop simulations [45, 46], we conducted additional simulations using the Gazebo robotics simulator. Ultimately, field experiments conducted in a mock-up scenario closely resembling real-world conditions demonstrated the practical applicability of the proposed method.

The experiments aimed to demonstrate several key aspects: (i) the alignment of planned trajectories with mission requirements, (ii) the necessity for the STL motion planner to fulfill mission specifications, and to show where the MILP alone is insufficient, (iii) the ability to replan missions in response to unexpected disturbances, (iv) a comparison of solutions with and without the attrition-aware planner, and (v) the feasibility of the method in real-world scenarios.

MATLAB was used to code the optimization method, with the MILP formulated using the CVX framework<sup>1</sup> and the STL motion planner using the CasADi library<sup>2</sup> and IPOPT<sup>3</sup> as a solver. The simulations were performed on a computer running Ubuntu 20.04 with an i7-8565U processor (1.80 GHz) and 32GB of RAM. Videos with the experiments and the numerical simulations in MATLAB and Gazebo are available at <http://mrs.felk.cvut.cz/milp-stl>. Figure 4 shows experiment snapshots with virtualized representations of the wind turbine.

### 5.1. Wind turbine inspection

We evaluated our planning approach using the wind turbine inspection scenario described in Section 2 and three drones. The scenario included nine target areas for the pylon inspection task and a single blade for the blade inspection task, as shown in Figure 5, in a mock-up area of 8 m × 20 m × 14 m. We chose to focus our inspection efforts on a single blade, as it provided sufficient scope to thoroughly test the effectiveness

and feasibility of our methodology. The optimization problem was run with the parameter values listed in Table 2. To ensure a reasonable computation time, short symbolic mission time intervals ( $T_{\text{ins}}$ ,  $T_{\text{bla}}$ , and  $T_N$ ) were used. During the inspection operation, the heading angles of the drones ( ${}^d\psi$ ) were adjusted and aligned with the displacement direction when moving towards the target areas (pylon inspection) or the blade (blade inspection). Furthermore, during blade inspection ( ${}^d\varphi_{\text{bla}}$ ), drones maintained their heading toward the blade surface between its two extremes, the leading edge and rotor shaft. Additionally, during pylon inspection ( ${}^d\varphi_{\text{tr}}$ ), the drone’s heading remained constant at a specific value for the inspection duration ( $T_{\text{ins}}$ ), chosen per each target based on the particular pylon area under surveillance.

The planned trajectories, along with the wind turbine, target areas, and blade extreme points, are depicted in Figure 5. The wind turbine is 12 m in height and 12 m in width, with a blade surface extension of 7 m starting from the rotor shaft to the leading edge. The small size of the wind turbine served two main purposes in our experiments. Firstly, it ensured that our experiments remained self-contained, conducted within a mock-up scenario to demonstrate the validity and effectiveness of our proposed planning approach. Secondly, it is worth noting that the experiments were conducted in an area where drone flight over 20 m in height was not permitted. The optimization problem required 145 s to solve and 4 s to find an initial guess solution. Real-world experiments confirm the alignment of planned trajectories with mission requirements, as evidenced by the results in Figure 12. The results indicate that the distance between vehicles remains above the specified threshold value  $\Gamma_{\text{dis}}$  at all times, and the velocity and acceleration of each vehicle remains within the allowed bounds of  $[{}^d\underline{v}^{(j)}, {}^d\bar{v}^{(j)}]$  and  $[{}^d\underline{a}^{(j)}, {}^d\bar{a}^{(j)}]$ , respectively. Additionally, throughout the blade inspection, the vehicle always maintain a certain distance ( $\Gamma_{\text{3|bla}} \in (\Gamma_{\text{bla}} - \varepsilon, \Gamma_{\text{bla}} + \varepsilon)$ ). Note that, for simplicity, we assume symmetric velocity and acceleration bounds for each drone, i.e.,  $|{}^d\underline{v}^{(j)}| = |{}^d\bar{v}^{(j)}|$  and  $|{}^d\underline{a}^{(j)}| = |{}^d\bar{a}^{(j)}|$ .

### 5.2. Comparative analysis with the initial guess solution

In this section, we assess the hierarchical planner’s effectiveness in solving the collaborative inspection problem. Notably, the STL optimization problem (1) is nonlinear and non-convex, requiring an initial guess for convergence to a feasible solution. As a result, we compare the MILP solution (4) with the solution obtained from the STL optimization, seeded with the MILP solution since we cannot compare them directly. The solver was indeed not able to converge for the pure STL

<sup>1</sup><http://cvxr.com/cvx/>

<sup>2</sup><https://web.casadi.org/>

<sup>3</sup><https://coin-or.github.io/Ipopt/>

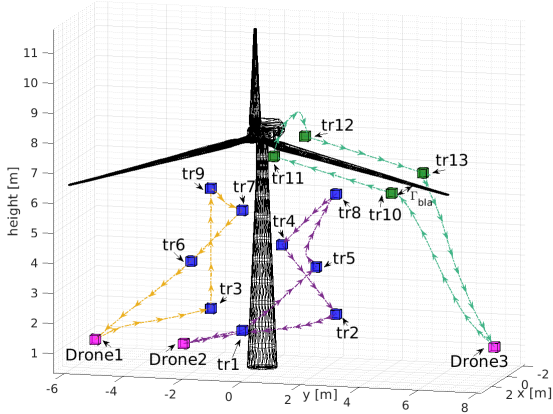


Figure 5: Wind turbine inspection scenario along with planner trajectories. Arrows depict the path followed by the drones throughout the mission.

Description	Sym.	Value	Description	Sym.	Value
Sampling period	$T_s$	0.05 [s]	Tunable parameter	$\lambda$	10 [-]
Safety distance	$\Gamma_{dis}$	1 [m]	Blade distance	$\Gamma_{bla}$	2.5 [m]
Max velocity	$\ d^i \dot{p}^{(j)}\ $	{1, 0.7, 1} [m/s]	Max accel.	$\ d^i \ddot{a}^{(j)}\ $	{1, 0.7, 1} [m/s <sup>2</sup> ]
Trigger. cond	$\eta$	1 [m]	Blade margin	$\varepsilon$	1 [m]
Rev. max velocity	$\ d^i \dot{p}^{(j)}\ $	2 [m/s]	Rev. max accel.	$\ d^i \ddot{a}^{(j)}\ $	5 [m/s <sup>2</sup> ]
Weight $\varphi_{tr}$	$\omega_{\varphi_{tr}}$	1 [-]	Weight $\varphi_{bla}$	$\omega_{\varphi_{bla}}$	1 [-]
Weight $\varphi_{ws}$	$\omega_{\varphi_{ws}}$	2 [-]	Weight $\varphi_{hm}$	$\omega_{\varphi_{hm}}$	1 [-]
Weight $\varphi_{obs}$	$\omega_{\varphi_{obs}}$	4 [-]	Pylon insp.	$T_{ins}$	1 [s]
Weight $\varphi_{dis}$	$\omega_{\varphi_{dis}}$	3 [-]	Mission time	$T_N$	13 [s]
STL safety margin	$\zeta$	0.2 [-]	Blade insp.	$T_{bla}$	1.5 [s]

Table 2: Optimization problem parameter values.

problem with generic initial condition, so highlighting the importance of having an MILP problem solution for warm starting the STL solution computation. We evaluate compliance with the mission requirements and how the hierarchical planner addresses nonlinear complexities such as obstacle avoidance, safety distance, and time requirements, which the MILP alone cannot handle. Figure 6 depicts the trajectories generated using motion primitives [29] and the waypoint sequences assigned by the MILP to each UAV.

Upon closer inspection, it is somewhat evident that the MILP formulation neglects vehicle accelerations  $\mathbf{a}^{(j)}$ , which are used for controlling the vehicles' motion. This may yield impractical trajectories or contain sharp turns and corners, potentially deviating from mission requirements, posing safety hazards, and resulting in high energy consumption. Furthermore, the MILP formulation also fails to consider the distance constraints ( $d^i \varphi_{dis}$  and  $d^i \varphi_{bla}$ ) due to the computational burden of accounting for vehicle dynamics. Thus, changing the  $\Gamma_{dis}$  (3d) and  $\Gamma_{bla}$  (3f) values may require a completely different set of optimal sequences (output of the MILP solver) for the final solution of the problem. Additionally, the MILP formulation does not address obstacle avoidance, which can be observed in Figure 6 with the trajectory crossing the wind turbine. In contrast, Figure 5 shows how the STL formulation fine-tunes the initial guess solution to meet the mutual safety distance constraint and other mission requirements.

Finally, the differences between the STL and MILP solutions not only affect the shape of trajectories, but also the sequence of targets to visit. To illustrate this point, Figure 7 presents a simple scenario involving two UAVs tasked with visiting a set of target areas within a specified time interval. Here,

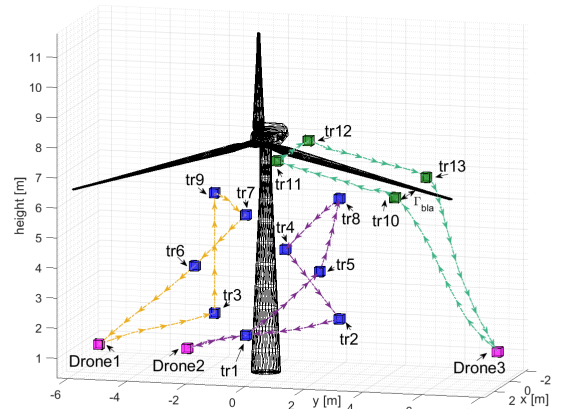


Figure 6: Trajectories obtained solely through the MILP formulation, without using the STL planner.

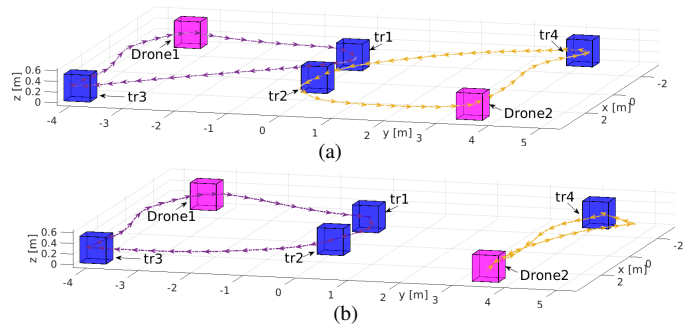


Figure 7: Simple scenario showing the independence of the final STL solution (7b) from the MILP initial guess (7a).

the MILP initial guess (Figure 7a) assigns the workload between the UAVs minimizing the total flight time, but ignores the vehicle dynamic and time constraints. Conversely, the STL planner (Figure 7b) reassigns targets to meet all mission specifications, showing the flexibility of the hierarchical planner.

### 5.3. Attrition-aware and event-triggered replanner

This section evaluates the performance of the *attrition-aware* and *event-triggered* replanner. The results of numerical simulations in MATLAB are depicted in Figures 8 and 10. User preferences were incorporated into the STL robust semantics to demonstrate the importance of safety requirements ( $d^i \varphi_{dis}$  and  $d^i \varphi_{obs}$ ) in the optimization problem, while also satisfying the same mission criteria. The trajectories obtained with user preferences resulted in safer paths, as shown in Figure 8. A generalized robustness score approach was used for the STL robust semantics, which affected the robustness of the safety requirements, as depicted in Figure 9. The weights used for the numerical simulations are reported in Table 2. The MILP planning problem (4) took 4 s to solve, while the STL optimization problem (5) took 107 s to solve.

To validate the event-triggered replanner's performance, we conducted simulations in the presence of unexpected disturbances that deviated the UAV from its planned path. The replanner was able to detect major deviations (i.e.,  $\|d^i \tilde{\mathbf{p}}_k - d^i \mathbf{p}_k^*\| > \eta$ ) and trigger a partial replanning process online, bringing

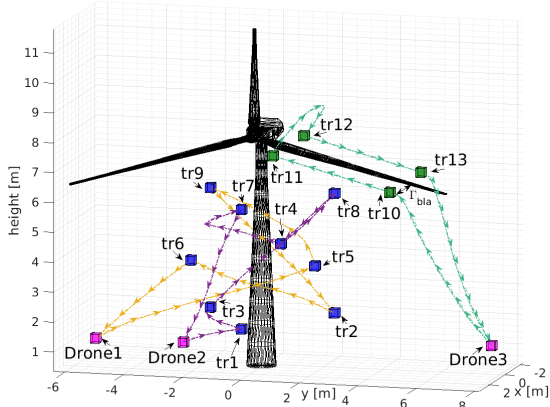


Figure 8: Wind turbine inspection scenario considering the attrition-aware motion planner.

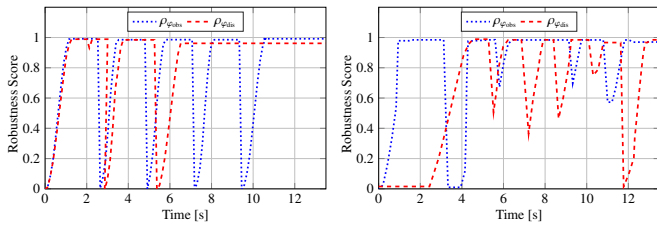


Figure 9: Robustness score profiles for  $\varphi_{\text{obs}}$  and  $\varphi_{\text{dis}}$  specifications. From left to right: data when considering the “basic” (1) and the attrition-aware (5) motion planner, respectively.

the UAV back to the next target area, as shown in Figure 10. The planner then checked whether mission requirements were fulfilled and continued replanning until the lost time was recovered. The optimization process took less than 1 s for each replanning.

#### 5.4. Field experiments

Experiments were performed with MRS F450 quadrotors [47, 48] in a mock-up scenario with three UAVs and a wind turbine (shown in Figure 5). Each UAV was equipped with an Intel NUC computer featuring an i7-8559U processor with 16GB of RAM, along with the Pixhawk flight controller. The software stack utilized the Noetic Ninjemys release

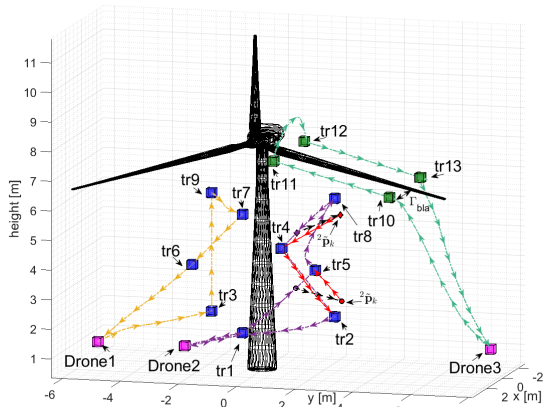


Figure 10: Event-triggered replanner. Black and red paths represent the deviation from the original path and the updated trajectory from the replanner, respectively.

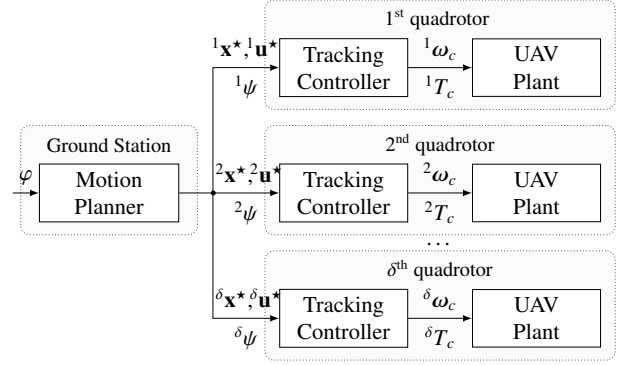


Figure 11: System Architecture. The ground station’s *STL Motion Planner* is responsible for generating trajectories ( ${}^1\mathbf{x}^*, {}^1\mathbf{u}^*, \dots, {}^\delta\mathbf{x}^*, {}^\delta\mathbf{u}^*$ ) and heading angles ( ${}^1\psi, \dots, {}^\delta\psi$ ) for the multi-rotor UAVs. These trajectories and heading angles serve as inputs to the *Tracking Controller*, which in turn calculates thrust ( ${}^1T_c, \dots, {}^\delta T_c$ ) and angular velocities ( ${}^1\omega_c, \dots, {}^\delta\omega_c$ ) for the *UAV Plant* [45].

of ROS running on Ubuntu 20.04. For further details, refer to [47, 48]. The wind turbine was simulated due to safety concerns. The UAVs followed trajectories generated in MATLAB and validated in Gazebo, accounting for the presence of the wind turbine. Videos of simulated and experimental results are available at <http://mrs.felk.cvut.cz/milp-stl>.

The system architecture, as illustrated in Figure 11, integrates the STL motion planner responsible for solving the optimization problem (1) to generate trajectories ( ${}^d\mathbf{x}^*, {}^d\mathbf{u}^*$ ) and heading angles ( ${}^d\psi$ ) for the fleet. This trajectory generation process occurs as a one-shot computation at time  $t_0$ , with the resulting trajectories serving as references for the UAV trajectory tracking controller [45].

During the flight tests, we confirmed the successful completion of the inspection mission specified by the STL formula (2). These flights also demonstrated adherence to physical constraints and safety features, including velocity ( ${}^d\mathbf{v}$  and  ${}^d\bar{\mathbf{v}}$ ) and acceleration ( ${}^d\mathbf{a}$  and  ${}^d\bar{\mathbf{a}}$ ) constraints, minimum safety distance ( $\Gamma_{\text{dis}}$ ), and blade distance ( $\Gamma_{\text{bla}}$ ) detailed in Table 2. Figure 4 provides snapshots from the experiments, illustrating the UAVs’ proximity to wind turbine infrastructure such as pylons and blades. To visualize this, we utilized a ROS package to project 3D mesh files – originally utilized in MATLAB and Gazebo simulations – onto the camera frames of the UAVs.

#### 5.5. Discussion

In this discussion section, we aim to compare our earlier work in [29] with the current study, highlighting significant improvements and advantages.

Firstly, the current solution yields feasible trajectories with diverse time bounds and vehicle constraints, while the approach proposed in [29] assumes quadrotors have the same physical constraints in terms of maximum velocity and acceleration ( $(\|{}^d\mathbf{v}^{(j)}\| = \|{}^d\bar{\mathbf{v}}^{(j)}\|, \|{}^d\mathbf{a}^{(j)}\| = \|{}^d\bar{\mathbf{a}}^{(j)}\|, \forall d \in \mathcal{D})$ ). Furthermore, and more importantly, the inspection mission in [29] is limited to visiting target areas of interest (for taking photos of power line insulators and tower mechanical structures), similar to the pylon inspection specification. However, in the pylon inspection, target areas are assigned to the drones by the MILP (4), so

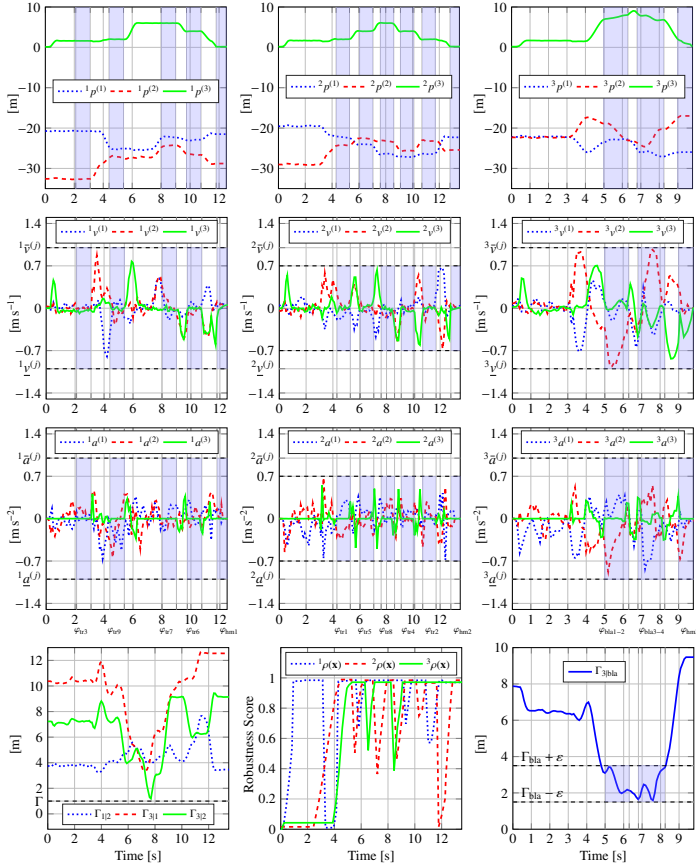


Figure 12: Position, velocity, acceleration, mutual safety distance ( $\Gamma_{m[m]}$ ), and distance from the blade  $\Gamma_{3[bla]}$  surface. Blue-colored time windows indicate the pylon and blade inspection operations.

one drone could theoretically cover all targets while the second could remain on the ground. In the algorithm proposed in [29], the targets are manually assigned to the drones. Finally, the optimization problem in (1) sets a minimum robustness threshold value ( $\tilde{\rho}_\varphi(d^{\mathcal{J}}, d^{\mathcal{V}^{\mathcal{J}}}) \geq \zeta$ ) that acts as a safety buffer, ensuring the satisfaction of the STL formula  $\varphi$  even in the presence of disturbances, which is not included in the previous work.

Secondly, the collaborative mission requirements, along with the heterogeneous constraints, make the problem challenging to solve. The closer the initial guess is to the optimal solution, the higher the chances that the nonlinear optimization problem will be feasible and converge to the optimal solution. This is the second contribution of the paper, i.e., proposing a two-step hierarchical solution for computing the initial solution.

Lastly, as discussed in the comparative analysis between the MILP and the STL solution seeded with the MILP initial guess in Section 5.2, the pure STL approach (as structured in the authors' previous work [29]) does not converge with a generic initial condition. This underscores the importance of having an MILP problem to warm-start the STL solution computation. This is another significant contribution of the manuscript.

## 6. Conclusions

This paper has presented a motion planning framework for collaborative inspection missions using a fleet of multi-rotor UAVs under heterogeneous constraints, with a focus on wind turbine inspection. The approach uses STL specifications to generate feasible trajectories meeting mission requirements, including safety and mission time requirements. An MILP approach provides a feasible initial guess solution for the STL planner, which helps solution convergence. An event-triggered replanning and attrition-aware planning handle failure and conflicting tasks. Validation has been performed through MATLAB and Gazebo simulations and field experiments.

Our investigation has revealed that depending solely on the simplified MILP solver falls short for our application. Nonetheless, it does serve as a valuable foundation for the complete STL planner. Embracing a hierarchical approach empowers us to handle a broader range of mission specifications and requirements compared to existing methods, albeit with a rise in computational complexity. In future work, we plan to investigate risk-aware techniques to model sensor failure and communication dropouts in the planning problem. Additionally, delving into conflicting temporal logic specifications and other temporal logic languages will expand the framework's utility to dynamically changing environments.

## Acknowledgments

The authors would like to thank Daniel Smrcka, Jan Bednar, Jiri Horyna, Tomas Baca, and the MRS group in Prague for their help with the field experiments. This publication is part of the R+D+i project TED2021-131716B-C22, funded by MCIN/AEI/10.13039/501100011033 and by the European Union NextGenerationEU/PRTR. This work was also partially supported by the European Union's Horizon 2020 research and innovation project AERIAL-CORE under grant agreement no. 871479, by the ECSEL Joint Undertaking (JU) research and innovation programme COMP4DRONES under grant agreement no. 826610, by the Czech Science Foundation (GAČR) grant no. 23-07517S, by CTU grant no. SGS23/177/OHK3/3T/13, and by the EU under the project Robotics and advanced industrial production (reg. no. CZ.02.01.01/00/22 008/0004590).

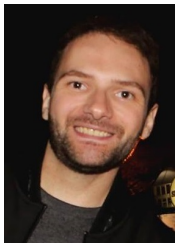
## References

- [1] A. Ollero, A. Suarez, J. M. Marredo, G. Cioffi, R. Pinięcka, G. Vasiljevic, V. D. Hoang, M. Marolla, J. Xing, M. Saska, S. Bogdan, E. Ebeid, F. Ruggiero, J. R. Martínez-de Dios, D. Scaramuzza, V. Lippiello, A. Viguria, Application of Intelligent Aerial Robots to the Inspection and Maintenance of Electrical Power Lines, in: K. Loupos (Ed.), Robotics and Automation Solutions for Inspection and Maintenance in Critical Infrastructures, now publishers inc., 2024, Ch. 8, pp. 178–201. doi: 10.1561/9781638282839.ch8.
- [2] A. Ollero, G. Heredia, A. Franchi, G. Antonelli, K. Kondak, A. Sanfeliu, A. Viguria, J. R. Martínez-de Dios, F. Pierri, J. Cortes, A. Santamaria-Navarro, M. A. Trujillo Soto, R. Balachandran, J. Andrade-Cetto, A. Rodriguez, The AEROARMS Project: Aerial Robots with Advanced Manipulation Capabilities for Inspection and Maintenance, IEEE Robotics & Automation Magazine 25 (4) (2018) 12–23. doi:10.1109/MRA.2018.2852789.

- [3] M. Car, L. Markovic, A. Ivanovic, M. Orsag, S. Bogdan, Autonomous Wind-Turbine Blade Inspection Using LiDAR-Equipped Unmanned Aerial Vehicle, *IEEE Access* 8 (2020) 131380–131387. doi:10.1109/ACCESS.2020.3009738.
- [4] A. Caballero, G. Silano, A Signal Temporal Logic Motion Planner for Bird Diverter Installation Tasks With Multi-Robot Aerial Systems, *IEEE Access* 11 (2023) 81361–81377. doi:10.1109/ACCESS.2023.3300240.
- [5] H. Shakhtrah, A. H. Sawalmeh, A. Al-Fuqaha, Z. Dou, E. Almaita, I. Khalil, N. S. Othman, A. Khreishah, M. Guizani, Unmanned Aerial Vehicles (UAVs): A Survey on Civil Applications and Key Research Challenges, *IEEE Access* 7 (2019) 48572–48634. doi:10.1109/ACCESS.2019.2909530.
- [6] A. Madridan, A. Al-Kaff, D. Martín, A. de la Escalera, Trajectory planning for multi-robot systems: Methods and applications, *Expert Systems with Applications* 173 (2021) 1–14. doi:10.1016/j.eswa.2021.114660.
- [7] R. Merkert, J. Bushell, Managing the drone revolution: A systematic literature review into the current use of airborne drones and future strategic directions for their effective control, *Journal of Air Transport Management* 89 (2020) 1–10. doi:10.1016/j.jairtraman.2020.101929.
- [8] G. Silano, A. Afifi, M. Saska, A. Franchi, A Signal Temporal Logic Planner for Ergonomic Human–Robot Collaboration, in: *International Conference on Unmanned Aircraft Systems*, 2023, pp. 328–335. doi:10.1109/ICUAS57906.2023.10156559.
- [9] C. Kanellakis, E. Fresk, S. S. Mansouri, D. Kominiaik, G. Nikolakopoulos, Towards Visual Inspection of Wind Turbines: A Case of Visual Data Acquisition Using Autonomous Aerial Robots, *IEEE Access* 8 (2020) 181650–181661. doi:10.1109/ACCESS.2020.3028195.
- [10] H.-A. Langaker, C. L. S. Hkon Kjerkreit, R. J. Moore, Øystein H Holhjem, I. Jensen, A. Morrison, A. A. Transeth, O. Kvien, G. Berg, T. A. Olsen, A. Hatlestad, T. Negrđ, R. Broch, J. E. Johnsen, An autonomous drone-based system for inspection of electrical substations, *International Journal of Advanced Robotic Systems* 18 (2) (2021) 1–15. doi:10.1177/17298814211002973.
- [11] G. Pola, M. D. Di Benedetto, Control of Cyber-Physical-Systems with logic specifications: A formal methods approach, *Ann. Rev. in Contr.* 47 (2019) 178–192. doi:10.1016/j.arcontrol.2019.03.010.
- [12] O. Maler, D. Nickovic, Monitoring Temporal Properties of Continuous Signals, in: *FTMTFTS*, 2004, pp. 152–166. doi:10.1007/978-3-540-30206-3\_12.
- [13] A. Donzé, O. Maler, Robust Satisfaction of Temporal Logic over Real-Valued Signals, in: *FORMATS*, 2010, pp. 92–106. doi:10.1007/978-3-642-15297-9\_9.
- [14] S. S. Mansouri, C. Kanellakis, E. Fresk, D. Kominiaik, G. Nikolakopoulos, Cooperative coverage path planning for visual inspection, *Control Engineering Practice* 74 (2018) 118–131. doi:10.1016/j.conengprac.2018.03.002.
- [15] C. S. Tan, R. Mohd-Mokhtar, M. R. Arshad, A Comprehensive Review of Coverage Path Planning in Robotics Using Classical and Heuristic Algorithms, *IEEE Access* 9 (2021) 119310–119342. doi:10.1109/ACCESS.2021.3108177.
- [16] F. Nekovář, J. Faigl, M. Saska, Multi-Tour Set Traveling Salesman Problem in Planning Power Transmission Line Inspection, *IEEE Robotics and Automation Letters* 6 (4) (2021) 6196–6203. doi:10.1109/LRA.2021.3091695.
- [17] J. Park, J. Kim, I. Jang, H. J. Kim, Efficient Multi-Agent Trajectory Planning with Feasibility Guarantee using Relative Bernstein Polynomial, in: *IEEE International Conference on Robotics and Automation*, 2020, pp. 434–440. doi:10.1109/ICRA40945.2020.9197162.
- [18] N. D. Tehrani, A. Krzywosw, I. Cherepinsky, S. Carlson, Multi-Objective Task Allocation for Multi-Agent Systems using Hierarchical Cost Function, in: *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2022, pp. 12045–12050. doi:10.1109/IROS47612.2022.9981071.
- [19] H. Zhu, J. Alonso-Mora, Chance-Constrained Collision Avoidance for MAVs in Dynamic Environments, *IEEE Robotics and Automation Letters* 4 (2) (2019) 776–783. doi:10.1109/LRA.2019.2893494.
- [20] W. Hönig, J. A. Preiss, T. K. S. Kumar, G. S. Sukhatme, N. Ayanian, Trajectory Planning for Quadrotor Swarms, *IEEE Transactions on Robotics* 34 (4) (2018) 856–869. doi:10.1109/TRO.2018.2853613.
- [21] Y. V. Pant, H. Abbas, R. Mangharam, Smooth operator: Control using the smooth robustness of temporal logic, in: *IEEE Conference on Control Technology and Applications*, 2017, pp. 1235–1240. doi:10.1109/CCTA.2017.8062628.
- [22] C. E. Luis, M. Vukosavljev, A. P. Schoellig, Online Trajectory Generation With Distributed Model Predictive Control for Multi-Robot Motion Planning, *IEEE Robotics and Automation Letters* 5 (2) (2020) 604–611. doi:10.1109/LRA.2020.2964159.
- [23] S. I. Krich, M. Montanari, V. Amendolare, P. Berestesky, Wind Turbine Interference Mitigation Using a Waveform Diversity Radar, *IEEE Transactions on Aerospace and Electronic Systems* 53 (2) (2017) 805–815. doi:10.1109/TAES.2017.2665143.
- [24] L. Lindemann, D. V. Dimarogonas, Barrier Function Based Collaborative Control of Multiple Robots Under Signal Temporal Logic Tasks, *IEEE Transactions on Control of Network Systems* 7 (4) (2020) 1916–1928. doi:10.1109/TCNS.2020.3014602.
- [25] Y. Chen, X. C. Ding, A. Stefanescu, C. Belta, Formal Approach to the Deployment of Distributed Robotic Teams, *IEEE Transactions on Robotics* 28 (1) (2012) 158–171. doi:10.1109/TRO.2011.2163434.
- [26] K. Leahy, Z. Serlin, C.-I. Vasile, A. Schoer, A. M. Jones, R. Tron, C. Belta, Scalable and Robust Algorithms for Task-Based Coordination From High-Level Specifications (ScRATChES), *IEEE Transactions on Robotics* 38 (4) (2022) 2516–2535. doi:10.1109/TRO.2021.3130794.
- [27] A. T. Buyukkocak, D. Aksaray, Y. Yazicioğlu, Control Barrier Functions with Actuation Constraints under Signal Temporal Logic Specifications, in: *European Control Conference*, 2022, pp. 162–168. doi:10.23919/ECC55457.2022.9838028.
- [28] D. P. Bertsekas, *Dynamic Programming and Optimal Control*, Athena Scientific, 2012, fourth Edition.
- [29] G. Silano, T. Baca, R. Penicka, D. Liuzza, M. Saska, Power Line Inspection Tasks With Multi-Aerial Robot Systems Via Signal Temporal Logic Specifications, *IEEE Robotics and Automation Letters* 6 (2) (2021) 4169–4176. doi:10.1109/LRA.2021.3068114.
- [30] N. Mehdipour, C.-I. Vasile, C. Belta, Specifying User Preferences Using Weighted Signal Temporal Logic, *IEEE Control Systems Letters* 5 (6) (2021) 2006–2011. doi:10.1109/LCSYS.2020.3047362.
- [31] E. Plaku, S. Karaman, Motion planning with temporal-logic specifications: Progress and challenges, *AI Communications* 29 (1) (2016) 151–162. doi:10.3233/AIC-150682.
- [32] D. Sun, J. Chen, S. Mitra, C. Fan, Multi-Agent Motion Planning From Signal Temporal Logic Specifications, *IEEE Robotics and Automation Letters* 7 (2) (2022) 3451–3458. doi:10.1109/LRA.2022.3146951.
- [33] G. Yang, C. Belta, R. Tron, Continuous-time Signal Temporal Logic Planning with Control Barrier Functions, in: *American Control Conference*, 2020, pp. 4612–4618. doi:10.23919/ACC45564.2020.9147387.
- [34] Y. V. Pant, H. Abbas, R. A. Quaye, R. Mangharam, Fly-by-logic: Control of multi-drone fleets with temporal logic objectives, in: *ACM/IEEE 9th International Conference on Cyber-Physical Systems*, 2018, pp. 186–197. doi:10.1109/ICCP.2018.00026.
- [35] C.-I. Vasile, V. Raman, S. Karaman, Sampling-based synthesis of maximally-satisfying controllers for temporal logic specifications, in: *IEEE International Conference on Intelligent Robots and Systems*, 2017, pp. 3840–3847. doi:10.1109/IROS.2017.8206235.
- [36] S. M. LaValle, *Sampling-Based Motion Planning*, Cambridge University Press, 2006.
- [37] C. Belta, S. Sadraddini, Formal Methods for Control Synthesis: An Optimization Perspective, *Annual Review of Control, Robotics, and Autonomous Systems* 2 (2) (2019) 115–140. doi:10.1146/annurev-control-053018-023717.
- [38] C. Belta, B. Yordanov, E. A. Gol, *Formal Methods for Discrete-Time Dynamical Systems*, Vol. 89, Springer, 2017.
- [39] G. E. Fainekos, G. J. Pappas, Robustness of temporal logic specifications for continuous-time signals, *Theoretical Computer Science* 410 (42) (2009) 4262–4291. doi:10.1016/j.tcs.2009.06.021.
- [40] Y. Gilpin, V. Kurtz, H. Lin, A Smooth Robustness Measure of Signal Temporal Logic for Symbolic Control, *IEEE Control Systems Letters* 5 (1) (2021) 241–246. doi:10.1109/LCSYS.2020.3001875.
- [41] G. A. Cardona, D. Kamale, C.-I. Vasile, Mixed Integer Linear Program-

ming Approach for Control Synthesis with Weighted Signal Temporal Logic, in: 26th ACM International Conference on Hybrid Systems: Computation and Control, 2023, pp. 1–12. doi:10.1145/3575870.3587120.

- [42] G. A. Cardona, C.-I. Vasile, Preferences on Partial Satisfaction using Weighted Signal Temporal Logic Specifications, in: European Control Conference, 2023, pp. 1–6. doi:10.23919/ECC57647.2023.10178201.
- [43] M. W. Mueller, M. Hehn, R. D’Andrea, A Computationally Efficient Motion Primitive for Quadcopter Trajectory Generation, IEEE Transactions on Robotics 31 (6) (2015) 1294–1310. doi:10.1109/TRO.2015.2479878.
- [44] F. Hillier, G. J. Lieberman, Introduction to Operations Research: Eight edition, McGraw-Hill Education, 2004.
- [45] T. Baca, M. Petrlik, M. Vrba, V. Spurny, R. Penicka, D. Hert, M. Saska, The MRS UAV System: Pushing the Frontiers of Reproducible Research, Real-world Deployment, and Education with Autonomous Unmanned Aerial Vehicles, Journal of Intelligent & Robotic Systems 102 (26) (2021) 1–28. doi:10.1007/s10846-021-01383-5.
- [46] G. Silano, P. Oppido, L. Iannelli, Software-in-the-loop simulation for improving flight control system design: a quadrotor case study, in: IEEE International Conference on Systems, Man and Cybernetics, 2019, pp. 466–471. doi:10.1109/SMC.2019.8914154.
- [47] D. Hert, T. Baca, P. Petracek, V. Kratky, V. Spurny, M. Petrlik, V. Matous, D. Zaitlik, P. Stoudek, V. Walter, P. Stepan, J. Horyna, V. Ptrizl, G. Silano, D. Bonilla Licea, P. Stibinger, R. Penicka, T. Nascimento, M. Saska, MRS Modular UAV Hardware Platforms for Supporting Research in Real-World Outdoor and Indoor Environments, in: International Conference on Unmanned Aircraft Systems, 2022, pp. 1264–1273. doi:10.1109/ICUAS54217.2022.9836083.
- [48] D. Hert, T. Baca, P. Petracek, V. Kratky, R. Penicka, V. Spurny, M. Petrlik, V. Matous, D. Zaitlik, P. Stoudek, V. Walter, P. Stepan, J. Horyna, V. Ptrizl, M. Sramek, A. Ahmad, G. Silano, D. Bonilla Licea, P. Stibinger, T. Nascimento, M. Saska, MRS Drone: A Modular Platform for Real-World Deployment of Aerial Multi-Robot Systems, Journal of Intelligent & Robotic Systems 108 (64) (2023) 1–34. doi:10.1007/s10846-023-01879-2.



**Giuseppe Silano** (Member, IEEE) is a tenured researcher at Ricerca sul Sistema Energetico and an associated researcher at Czech Technical University in Prague (CTU-P). He earned his B.Sc., M.Sc., and Ph.D. from the University of Sannio, Italy, in 2012, 2016, and 2020, respectively. During his Ph.D., he visited LAAS-CNRS and participated in the MBZIRC 2020 robotic competition. He

was a post-doctoral researcher at CTU-P with the MRS group (2020-2022) and a visiting researcher at the RAM group, University of Twente, in 2022. His research focuses on UAV motion planning, model predictive control, formal methods for robotics, communication-aware robotics, and human-robot collaboration. He has authored over 25 publications and held leadership roles in European research projects. Since 2022, he serves as an associate editor for key conferences.

**Alvaro Caballero** earned his B.Sc. in aerospace engineering, M.Sc. in aeronautical engineering, and Ph.D. in aerial robotics from the University of Seville, Spain, in 2014, 2016, and 2022, respectively. Currently, he holds a post-doctoral position at the GRVC



Robotics Lab, University of Seville. Since 2014, he has contributed to various projects, including FP7 EC-SAFEMOBIL, MBZIRC 2017, and H2020 AEROARMS, HYFLIERS, AERIAL-CORE, and OMICRON. He also collaborates with industry leaders like NAVANTIA or ENEL. His main research interests focus on motion planning for aerial manipulation in inspection and maintenance operations.

**Davide Liuzza** obtained his Ph.D. in automation engineering from the University of Naples Federico II, Italy, in 2013. He conducted research visits to institutions in the UK and Sweden during his doctoral studies. Afterward, he held postdoctoral positions at KTH and the University of Sannio. Subsequently, he was a visiting researcher at Chalmers University of Technology and a staff researcher at ENEA. Currently, he serves as an Assistant Professor at the University of Sannio. His research focuses on networked control systems, multiagent system coordination, nonlinear systems’ stability, and energy system control. He also explores topics like human-robot coordination and real-time control of nuclear fusion systems.



**Luigi Iannelli** (Senior Member, IEEE) earned his M.Sc. in computer engineering from the University of Sannio, Italy, in 1999, and his Ph.D. in information engineering from the University of Napoli Federico II, Italy, in 2003. He became an associate professor of automatic control at the University of Sannio in 2016 after serving as an assistant professor. With research visits to institutions in Sweden and the Netherlands, his work centers on analyzing and controlling switched systems, stability of piecewise-linear systems, smart grid control, and applying control theory to power electronics and UAVs. He co-edited “Dynamics and Control of Switched Electronic Systems” (Springer, 2012).



**Stjepan Bogdan** (Senior Member, IEEE) earned his B.Sc., M.Sc., and Ph.D from the University of Zagreb, Croatia, in 1990, 1993, and 1999, respectively. He was a Fulbright Researcher at the Automation and Robotics Research Institute, Arlington, USA, under Prof. Frank Lewis. Currently, he is a Full Professor at the Laboratory for Robotics and Intelligent Control Systems (LARICS), University of Zagreb. Bogdan has coauthored four books and numerous articles on topics such as autonomous systems, aerial robotics, multi-agent systems, intelligent control systems, bio-inspired systems, and discrete event systems.



**Martin Saska** (Member, IEEE) holds an M.Sc. from Czech Technical University (2005) and a Ph.D. from the University of Wuerzburg, Germany. He was a Visiting Scholar at the University of Illinois at Urbana-Champaign (2008) and the University of Pennsyl-

vania (2012, 2014, and 2016). Since 2009, he has been with the Czech Technical University, first as a research fellow and then as an associate professor, leading the Multi-Robot Systems Lab and co-founding the Center for Robotics and Autonomous Systems. Saska has authored/coauthored more than 90 peer-reviewed conference papers and 60 journal publications.