Power Line Inspection Tasks with Multi-Aerial Robot Systems via Signal Temporal Logic Specifications

Giuseppe Silano[1], Tomas Baca[1], Robert Penicka[1], Davide Liuzza[2], and Martin Saska[1]

***Abstract*—A framework for computing feasible and constrained trajectories for a fleet of quad-rotors leveraging on Signal Temporal Logic (STL) specifications for power line inspection tasks is proposed in this paper. The planner allows the formulation of complex missions that avoid obstacles and maintain a safe distance between drones while performing the planned mission. An optimization problem is set to generate optimal strategies that satisfy these specifications and also take vehicle constraints into account. Further, an event-triggered replanner is proposed to reply to unforeseen events and external disturbances. An energy minimization term is also considered to implicitly save quad-rotors battery life while carrying out the mission. Numerical simulations in MATLAB and experimental results show the validity and the effectiveness of the proposed approach, and demonstrate its applicability in real-world scenarios.**

***Index Terms*—Task and Motion Planning, Multi-Robot Systems, Aerial Systems: Applications**

I. Introduction

OVER the last two decades, global energy demand has increased rapidly due to demographic and economic growth. This has created new challenges for electricity supply companies, which are constantly looking for new solutions to minimize the frequency of power outages. Power failures are particularly critical when the environment and public safety are at risk, e.g., for hospitals, sewage treatment plants and telecommunication systems. One of the major causes of a power outage is damage to transmission lines, usually due to high winds, storms, or inefficient maintenance activities [1].

Nowadays, the most common strategy for reducing energy interruptions is to schedule periodic inspections using manned helicopters equipped with multiple sensors. Data are captured over thousands of kilometers by experienced crews for subsequent processing. There are two major drawbacks to this approach: first, flights are dangerous for operators who have to fly close to power towers; second, the inspection is extremely time-consuming and expensive ($1,500 for a one-hour flight) and is prone to human error [2], [3].

Manuscript received: October 15, 2020; Revised December 19, 2020; Accepted February 20, 2021.

This paper was recommended for publication by Editor Nancy Amato upon evaluation of the Associate Editor and Reviewers' comments. This work was partially funded by the European Union's Horizon 2020 research and innovation programme AERIAL-CORE under grant agreement no. 871479, by CTU grant no. SGS20/174/OHK3/3T/13, and by the Czech Science Foundation (GAČR), within research projects no. 19-22555Y and 20-10280S.

[1]Giuseppe Silano, Tomas Baca, Robert Penicka, and Martin Saska are with the Czech Technical University in Prague, Czech Republic, email: {name.surname}@fel.cvut.cz.

[2]Davide Liuzza is with the ENEA Fusion and Nuclear Safety Department, Italy, email: davide.liuzza@enea.it.

Digital Object Identifier (DOI): see top of this page.

Multiple solutions have been investigated in the literature for automating this task. Unmanned Aerial Vehicles (UAVs) and Rolling On Wire (ROW) robots [3] have been proposed as valuable solutions to replace helicopters within the process. The most promising and the most flexible solution is to use UAVs that can perform various levels of inspection depending on the wing types and the task of interest [1].

However, the use of UAVs to achieve these tasks is particularly challenging, due to the strong electromagnetic interference produced by power lines, and the presence of obstacles along the line [2]. Accurate task planning is therefore needed to mitigate such issues and to accomplish the assigned mission safely. Temporal-Logic (TL) can be of help by providing a powerful mathematical tool for the automatic design of feedback control laws that meet complex temporal requirements. In particular, Signal Temporal Logic (STL) [4], [5] can be used to describe planning objectives that are more complex than point-to-point planning algorithms [6]. This approach leverages on the definition of quantitative semantics [7], [8] for TLs to interpret a formula w.r.t. a discretized abstraction of the robot motion modelled as a finite transition system. The result is an optimization problem with the goal of maximizing a real-valued metric (called *robustness*) that denotes how strongly a specification is satisfied or violated.

A. Related works

As detailed in [2], [9], there are three main challenges for UAVs inspecting power lines: (i) *visual servoing* to ensure power line tracking and autonomous navigation; (ii) *obstacle detection and avoidance* to prevent possible collisions with the towers and obstacles along the path; (iii) *robust control* to provide high stability and positioning, hence allowing for close-up inspections.

Much of the state-of-the-art focuses on the first two problems. Some works [2], [10] propose new methods for electric tower detection and image segmentation. Others deal with the detection of possible mechanical faults or damages to isolation material [3], [9]. In this case, a highly desired feature is a control strategy that enables trajectories to be obtained not only for a single vehicle, but possibly for a fleet cooperating at the same time, in the same area, while avoiding obstacles and possible crashes and respecting the given mission specifications and time bounds.

Other approaches focus on the endurance of the drone mission as a way to maximize the exploration within its battery life. Solutions have been proposed for performing cooperative aerial coverage path planning with a multi-UAV system [2], [11]. However, these problems do not usually take into consideration the dynamics of the drone or physical constraints

on vehicles. They, therefore, do not offer guarantees on the feasibility of the path and on the DOFs of the vehicle motion. Often, the proposed solution is an extension of the well-known Traveling Salesman Problem [11] which, being an NP-hard problem, easily becomes unsolvable within a reasonable time when the complexity increases exponentially by the number of vehicles and variables.

As regards the trajectory planning problem for a multi-robot system, especially for quad-rotors, several approaches have been investigated in the literature [12]–[14]. Most of the solutions rely on an abstract grid-based representation of the environment [15] or on abstract dynamics of the agents combining a discrete planner with a continuous trajectory generator [14]. Others propose centralized multi-agent path planning methods using relative safe flight corridors to find feasible trajectories for the agents [16]. Although these approaches can compute collision-free trajectories for a large number of agents in a short time, they do not offer guarantees that the aircraft will comply with the physical constraints or perform the task in a given time window. On the other hand, whether the model of quad-rotors is considered [13], the solutions rely on information sharing between agents, making them difficult to achieve in presence of electromagnetic interference, such as in power line inspection tasks. Moreover, even when the planning algorithms demonstrate computational efficiency [14], they do not provide any reference regarding velocity and acceleration, leaving the controller to generate these signals.

Many of solutions use Linear Temporal Logic (LTL) as the mission specification language to synthesize the optimization problem without considering explicit time bounds on the mission objectives [12]. Other solutions propose the use of STL specifications to describe mission requirements without the need to discretize the dynamics or the environment [17]. Unlike LTL, STL is equipped with qualitative and quantitative semantics, meaning that it is not only able to assess whether the system execution meets the desired requirements, but also provides a measure of how well the requirements are being met (i.e., a *robustness function*). Furthermore, STL semantics takes the absolute time information explicitly into account, therefore making it possible to plan when a given task has to be executed in the context of the whole mission.

B. Contributions

In this paper, we propose a framework for encoding inspection missions for a fleet of quad-rotors as STL specifications. Then, using the motion primitives defined in [18], we construct an optimization problem to generate optimal strategies that satisfy the specifications. The proposed approach generates feasible dynamic trajectories accounting for the velocity and acceleration constraints of the vehicles, avoiding obstacles and maintaining a safe distance between drones, while complying with the specifications for the mission. An event-triggered re-planning strategy is also proposed to account for disturbances and unforeseen events along the tracking. The optimization problem is reshaped to compute the feasible path to reconnect the drone to the previously computed optimal offline solution. In addition, a minimum energy problem is set up to implicitly prevent the quad-rotors from draining the battery while carrying out the mission specification successfully.

The advantages are twofold: (i) the full expressiveness of the STL formulas allows explicit time requirements to be taken into consideration, making the framework easy to reuse and customize for applications of interest; (ii) thanks to the motion primitives, the proposed approach can generate trajectories in accordance with pre-set velocity and acceleration constraints that can be well-tracked by lower-level controllers.

Numerical simulations achieved in MATLAB show the validity of the proposed approach. Various scenarios were considered for an evaluation of the trajectory generator performance. A comparison between the proposed strategy and an existing stat-of-the-art solution is given at this stage. In addition, Gazebo simulations and real-experiments were used to demonstrate the applicability of the method in a scenario closer to the real implementation.

II. Problem Description

The work presented here forms a part of the AERIAL-CORE European project. The *power tower inspection* task is considered. A multi-robot system carries out a detailed investigation of power equipment, looking for possible faults. The visual examination outputs videos or pictures of towers, cable installations, and their surroundings performing a preliminary remote evaluation. The aim is to identify components that need to be replaced.

We suppose that the UAVs operate in a known environment, represented by a map that also includes the position of obstacles and the power tower. Also, that the UAVs are equipped with the necessary sensors and software for their own precise localization and state estimation [19].

III. Preliminaries

Let us consider a continuous-time dynamical system $\mathcal{H}$ and its discrete time version $x_{k+1} = f(x_k, u_k)$, where $x_k, x_{k+1} \in X \subset \mathbb{R}^n$ are the current state and the next state of the system, respectively, $u \in U \subset \mathbb{R}^m$ is the control input and $f : X \times U \to X$ is differentiable in both of the arguments. The initial state is denoted by x_0 and takes values from some initial set $X_0 \subset \mathbb{R}^n$. Let $T_s \in \mathbb{R}_{\geq 0}$ and $T \in \mathbb{R}_{\geq 0}$ be the sampling period and the trajectory duration, respectively, so we can write the time interval as the vector $\mathbf{t} = (0, T_s, \ldots, NT_s)^\top \in \mathbb{R}^{N+1}$, where $NT_s = T$ and $\mathbf{t}_k$, $k \in \mathbb{N}_{\geq 0}$, denote the k-element of the vector $\mathbf{t}$. Therefore, given an initial state x_0 and a finite control input sequence $\mathbf{u} = (u_0, \ldots, u_{N-1})^\top \in \mathbb{R}^N$, a trajectory of the system is the unique sequence of states $\mathbf{x} = (x_0, \ldots, x_N)^\top \in \mathbb{R}^{N+1}$. Similarly to $\mathbf{t}_k$, with $\mathbf{u}_k$ and $\mathbf{x}_k$ we denote the k-element of vector $\mathbf{u}$ and $\mathbf{x}$, respectively.

A. Signal Temporal Logic

The trajectory generator is designed to satisfy a specification expressed in STL [4], [5]. STL is a logic that allows the succinct and unambiguous specification of a wide variety of desired system behaviors over time, such as "The quad-rotor reaches the goal within 10 time units while always avoiding obstacles". The semantics of STL are defined in [5] and is not reported here for the sake of brevity.

B. Robust Signal Temporal Logic

The presence of a dynamic environment, unforeseen events, and external disturbances can affect the closed loop behavior and the satisfaction of the STL formula φ. For this reason, it is convenient to have a maneuverability margin in an attempt to maximize the degree of satisfaction with the formula. This can be formally defined and computed using the *robust semantic* of temporal logic [4], [5], [8].

Definition 1 (Robustness): The robustness of an STL formula φ relative to the system trajectory $\mathbf{x}$ at time $\mathbf{t}_k$ is defined via the following recursive formulas

$$\begin{aligned}
\rho_{p_i}(\mathbf{x},\mathbf{t}_k) &= \mu_i(x_{\mathbf{t}_k}),\\
\rho_{\neg\varphi}(\mathbf{x},\mathbf{t}_k) &= -\rho_{\varphi}(\mathbf{x},\mathbf{t}_k),\\
\rho_{\varphi_1\wedge\varphi_2}(\mathbf{x},\mathbf{t}_k) &= \min\left(\rho_{\varphi_1}(\mathbf{x},\mathbf{t}_k),\rho_{\varphi_2}(\mathbf{x},\mathbf{t}_k)\right),\\
\rho_{\Box_I\varphi}(\mathbf{x},\mathbf{t}_k) &= \min_{\mathbf{t}'_k\in[\mathbf{t}_k+I]} \rho_{\varphi}(\mathbf{x},\mathbf{t}'_k),\\
\rho_{\Diamond_I\varphi}(\mathbf{x},\mathbf{t}_k) &= \max_{\mathbf{t}'_k\in[\mathbf{t}_k+I]} \rho_{\varphi}(\mathbf{x},\mathbf{t}'_k),\\
\rho_{\varphi_1\mathcal{U}\varphi_2}(\mathbf{x},\mathbf{t}_k) &= \max_{\mathbf{t}'_k\in[\mathbf{t}_k+I]} \Big(\min\left(\rho_{\varphi_2}(\mathbf{x},\mathbf{t}'_k)\right),\\
&\qquad \min_{\mathbf{t}''_k\in[\mathbf{t}_k,\mathbf{t}'_k]} \left(\rho_{\varphi_1}(\mathbf{x},\mathbf{t}''_k)\right)\Big),
\end{aligned}$$

where $\mathbf{t}_k+I$ is meant here as the Minkowski sum between the scalar $\mathbf{t}_k$ and the interval I. In the above formulas, $\mu_i(x_{\mathbf{t}_k})$ is a smooth function called *predicate* which results true if its value is grater or equal than zero, negative otherwise. On example for the robot case could be being inside a target region or being outside an obstacle region, with regions described by a certain number of predicates. All the other expressions define operators acting on other STL subformulas, thus implicitly describing the semantic in a recursive way. Further details can be found in [4], [5], [8]. For simplicity, we will write $\rho_\varphi(\mathbf{x})$ instead of $\rho_\varphi(\mathbf{x},0)$ when $\mathbf{t}_k=0$. Also, we will say that $\mathbf{x}$ violates the STL formula φ at time $\mathbf{t}_k$ if $\rho_\varphi(\mathbf{x},\mathbf{t}_k)\leq 0$ and that $\mathbf{x}$ satisfies φ if $\rho_\varphi(\mathbf{x},\mathbf{t}_k)>0$.

Thus, we can compute control inputs $\mathbf{u}$ by maximizing the robustness over the set of finite state and input sequences $\mathbf{x}$ and $\mathbf{u}$, respectively. The obtained sequence $\mathbf{u}^\star$ is valid if $\rho_\varphi(\mathbf{x}^\star,\mathbf{t}_k)$ is positive, where $\mathbf{x}^\star$ and $\mathbf{u}^\star$ obey the dynamical system $\mathcal{H}$. The larger $\rho_\varphi(\mathbf{x}^\star,\mathbf{t}_k)$ is, the more robust the behavior of the system is.

Definition 2 (LSE Robustness) [15]: Let us consider $c\geq 1$, the smooth approximation of the m-array max and min is

$$\max(\rho_{\varphi_1},\ldots,\rho_{\varphi_m}) \approx \frac{1}{c}\log\left(\sum_{i=1}^{m} e^{c\rho_{\varphi_i}}\right),$$

$$\min(\rho_{\varphi_1},\ldots,\rho_{\varphi_m}) \approx -\frac{1}{c}\log\left(\sum_{i=1}^{m} e^{-c\rho_{\varphi_i}}\right).$$

This log-sum-exponential (LSE) approximation is smooth, and an analytical form of its gradient exists. This robustness approximation approaches the true robustness values given according to Def. 1 as $c\to\infty$. The larger c is, the greater the accuracy of the approximation is.

IV. Problem Formulation

In this section, we show how to generate trajectories for a fleet of q quad-rotors starting from mission specifications φ.

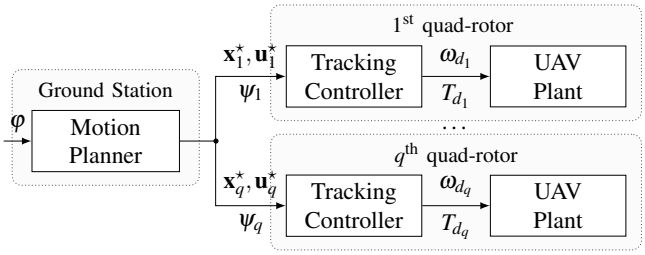

Figure 1: Control scheme. The motion planner generates the trajectories $(\mathbf{x}_i^\star,\mathbf{u}_i^\star)$ and the heading angles ψ_i, with $i=\{1,\ldots,q\}$, for the q quad-rotors by using the STL mission specifications φ. A tracking controller supplies the desired angular velocities ω_{d_i} and thrust T_{d_i} commands for the UAVs.

The motion planner is the result of an optimization problem that outputs a global feasible path for the vehicles accounting for their constraints. These paths are used as a reference by the trajectory tracking controller that performs the inspection. Figure 1 describes the overall system architecture.

A. Motion planner

The use of an STL robust semantic allows to synthesize the motion planner, i.e., finding a control sequence for the q quad-rotors that satisfies a given STL formula φ. Such a problem is casted for each quad-rotor as an optimization problem over the control $\mathbf{u}=(u_0,\ldots,u_{N-1})^\top$ and state $\mathbf{x}=(x_0,\ldots,x_{N-1})^\top$ sequences as follows

$$\begin{aligned}
\underset{\mathbf{u},\mathbf{x}}{\text{maximize}}\ & \rho_\varphi(\mathbf{x})\\
\text{s.t.}\ & \mathbf{x}_{k+1}=f(\mathbf{x}_k,\mathbf{u}_k),\forall k=\{0,1,\ldots,N-1\}
\end{aligned}, \tag{1}$$

where $\mathbf{x}_0=x_0$. Note that, in order to make this paper more readable, in (1) we provided the optimization problem for each quad-rotor, considering them decoupled. However, in the case of coupling among some of them, such as for a minimum distance to be always kept, problem (1) can be analogously written taking into account the state and control sequences of all the involved vehicles as decision variables, as well as their dynamics. The coupling constraint can be embedded in the STL formula used in the objective function.

As detailed in Def. 1, ρ_φ uses non-differentiable functions max and min. Therefore, the robustness of the STL formula φ is itself non-differentiable as a function of the trajectory $\mathbf{x}$ and the control inputs $\mathbf{u}$. While mixed-integer programming solvers [17], non-smooth optimizers, or stochastic heuristics [20] can be used to find a solution for this problem, the problem is NP-hard, and these approaches could fail with the increase of the number of variables. However, as shown in [15], a good approach for mitigating computational complexity is to adopt a smooth approximation $\tilde{\rho}_\varphi$ of the robust function ρ_φ. One of the possible choices is LSE robustness (Def. 2). In this case, the resulting optimization problem is still non-convex, but smooth optimization techniques, such as sequential quadratic programming, can be used to find a local maximum. In this paper, such an approach is adopted to compute the robustness value by using the smooth operator defined in [15].

To come up with a trajectory that satisfies the vehicle constraints, the motion primitives defined in [18] have been considered. The method allows for obtaining rapid generation and feasibility verification of motion primitives for quad-rotors. Let us define the state $\mathbf{x}$ and control $\mathbf{u}$ sequences as $\mathbf{x}_k = (\mathbf{p}_k^{(1)}, \mathbf{v}_k^{(1)}, \mathbf{p}_k^{(2)}, \mathbf{v}_k^{(2)}, \mathbf{p}_k^{(3)}, \mathbf{v}_k^{(3)})^\top$ and $\mathbf{u}_k = (\mathbf{a}_k^{(1)}, \mathbf{a}_k^{(2)}, \mathbf{a}_k^{(3)})^\top$, where $\mathbf{p}_k^{(j)}$, $\mathbf{v}_k^{(j)}$, and $\mathbf{a}_k^{(j)}$, with $j = \{1,2,3\}$, represent the vehicle's position, velocity, and acceleration at time instant k along the j-axis of the inertial frame, respectively. The optimization problem (1) can be reformulated approximating the translational dynamics of the quad-rotor separately along each j-axis with the splines $\mathbf{S}^{(j)}(\mathbf{p}_k^{(j)}, \mathbf{v}_k^{(j)}, \mathbf{a}_k^{(j)}) = (\mathbf{p}_{k+1}^{(j)}, \mathbf{v}_{k+1}^{(j)}, \mathbf{a}_{k+1}^{(j)})^\top$ defined as

$$\mathbf{S}^{(j)} = \begin{pmatrix} \frac{\alpha}{120}\mathbf{t}_k^5 + \frac{\beta}{24}\mathbf{t}_k^4 + \frac{\gamma}{6}\mathbf{t}_k^3 + \mathbf{a}_k^{(j)}\mathbf{t}_k^2 + \mathbf{v}_k^{(j)}\mathbf{t}_k + \mathbf{p}_k^{(j)} \\ \frac{\alpha}{24}\mathbf{t}_k^4 + \frac{\beta}{6}\mathbf{t}_k^3 + \frac{\gamma}{2}\mathbf{t}_k^2 + \mathbf{a}_k^{(j)}\mathbf{t}_k + \mathbf{v}_k^{(j)} \\ \frac{\alpha}{6}\mathbf{t}_k^3 + \frac{\beta}{2}\mathbf{t}_k^2 + \gamma\mathbf{t}_k + \mathbf{a}_k^{(j)} \end{pmatrix}, \quad (2)$$

where $\mathbf{p}_0^{(j)} = p_0^{(j)}$, $\mathbf{v}_0^{(j)} = v_0^{(j)}$, and $\mathbf{a}_0^{(j)} = a_0^{(j)}$, while parameters α, β, and γ that can be tuned to achieve a desired motion fixing a combination of position, velocity, and acceleration at the start and end points [18, Appx. A]. Such an approach ensures compliance with safety requirements and intrinsically embeds the gravity compensation [18, Sec. III]. Thus, the accelerations $\mathbf{a}^{(j)}$ are meant as the variations w.r.t. the vertical equilibrium position.

Thus, the problem (1) can be reformulated replacing $\rho_\varphi(\mathbf{x})$ with its smooth version $\tilde{\rho}_\varphi(\mathbf{x})$ considering for the mathematical formulation of the trajectory generator $\mathbf{S}^{(j)}$. Moreover, exploiting the decoupling of the drone dynamics into three orthogonal axes [18, Sec. III-C], the original optimization problem (1) can be split into three independent problems for each j-axis, as follows

$$\begin{aligned} \underset{\mathbf{p}^{(j)},\mathbf{v}^{(j)},\mathbf{a}^{(j)}}{\text{maximize}} \quad & \tilde{\rho}_\varphi(\mathbf{p}^{(j)}, \mathbf{v}^{(j)}) \\ \text{s.t.} \quad & |\mathbf{v}_k^{(j)}| \leq \mathbf{v}_{\max}^{(j)}, |\mathbf{a}_k^{(j)}| \leq \mathbf{a}_{\max}^{(j)}, \\ & \text{eq. (2)}, \forall k = \{0,1,\ldots,N-1\} \end{aligned} \quad , \quad (3)$$

where $\mathbf{v}_{\max}^{(j)}$ and $\mathbf{a}_{\max}^{(j)}$ are the desired maximum values of velocity and acceleration along the motion, respectively. The higher N is, the bigger the number of DOFs is. Consequently, the computational burden for solving the optimization problem increases. However, smaller values of N restrict the DOFs of the motion planner, thus potentially providing a trajectory that does not satisfy the STL specification. While the acceleration $\mathbf{a}^{(j)}$ is bounded in norm in the optimization problem (3), the bound on the jerk is implicitly accounted by the chosen motion primitives in [18].

B. Event-triggered replanner

As explained in the previous section, the adoption of motion primitives allows to obtain feasible solutions for the quad-rotors dynamics. It may be the case that, due to unexpected large disturbances at runtime, a significant mismatch between the planned trajectory and the quad-rotor state can be experienced. To cope with such an issue, here we introduce an online event-based replanner.

Specifically, in our case we consider to obtain data only at certain discrete time instances denoted by $\bar{\mathbf{t}}$. Let $T_e \in \mathbb{R}_{\geq 0}$ and $T_g \in \mathbb{R}_{\geq 0}$ be the event-triggering period (multiple of the sampling period T_s) and the "topic" waypoint period (a low-rate sequence of the state $\mathbf{x}$, with $T_g \gg T_s$), respectively, so we can write the discrete time instances $\bar{\mathbf{t}}$ and $\hat{\mathbf{t}}$ as the vectors $\bar{\mathbf{t}} = (0, T_e, \ldots, LT_e)^\top \in \mathbb{R}^{L+1}$ and $\hat{\mathbf{t}} = (0, T_g, \ldots, GT_g)^\top \in \mathbb{R}^{G+1}$, where $LT_e \subseteq T$ and $GT_g \subset T$. The term $\bar{\mathbf{t}}_l$, $l \in \mathbb{N}_{\geq 0}$, denotes the l-element of the vector $\bar{\mathbf{t}} \subseteq \mathbf{t}$, while $\hat{\mathbf{t}}_g$, $g \in \mathbb{N}_{\geq 0}$, denotes the g-element of the vector $\hat{\mathbf{t}} \subset \mathbf{t}$.

We also denote with $\tilde{\mathbf{p}}$ the runtime trajectory position of the drone. Notice that such trajectory could be different from the optimal one $\mathbf{p}^\star$ due to disturbances acting at runtime.

At each time instant, say $\bar{\mathbf{t}}_l \in \bar{\mathbf{t}}$ the condition $|\tilde{\mathbf{p}}_l - \mathbf{p}_l| > \eta$ is evaluated, with $\eta > 0$ a design parameter triggering threshold. If such condition results true, then a trigger is generated and the actual drone position is communicated to the ground station. The latter performs an optimal replanning operation over the time interval $\{\bar{\mathbf{t}}_l, \hat{\mathbf{t}}_{g+1}\}$, where $\hat{\mathbf{t}}_{g+1}$ is the time associated with the next topic position $\mathbf{p}_{g+1}$. In this way, the feasible path between the triggering position $\mathbf{p}_l$ and the next position $\mathbf{p}_{g+1}$ is computed.

C. Energy-aware planner

The synthesized motion planner problem in Sec. IV-A can be modified to ensure that the quad-rotors also save their battery charge while carrying out their mission successfully. The objective is to generate a trajectory for the q quad-rotors that also takes into account the energy requirement of the vehicles.

Let us define the decision variables $\boldsymbol{\varepsilon}_k = (\boldsymbol{\varepsilon}_k^{(1)}, \boldsymbol{\varepsilon}_k^{(2)}, \boldsymbol{\varepsilon}_k^{(3)})^\top$, where $\boldsymbol{\varepsilon}_k^{(j)}$, with $j = \{1,2,3\}$, represents the bound on the square norm of the quad-rotor acceleration (i.e., the control input) as a proxy of the energy at time instant k along the j-axis of the inertial frame. As discussed in [13], [14], the optimal trajectory that deals with energy minimization can be obtained by minimizing the positive semi-definitive quadratic form $\boldsymbol{\varepsilon}_k^\top \mathbf{Q} \boldsymbol{\varepsilon}_k$, where $\mathbf{Q} \in \mathbb{R}^{3N \times 3N}$ such that for all $\boldsymbol{\varepsilon}_k \in \mathbb{R}^{3N}$ we have that $\boldsymbol{\varepsilon}_k^\top \mathbf{Q} \boldsymbol{\varepsilon}_k \geq 0$. Thus, the optimization problem (3) can be reformulated by adding a new term to the cost function and bounding the system energy $\|\mathbf{a}^{(j)^\top} \mathbf{a}^{(j)}\|^2$. Namely we write:

$$\begin{aligned} \underset{\mathbf{p}^{(j)},\mathbf{v}^{(j)},\mathbf{a}^{(j)},\boldsymbol{\varepsilon}^{(j)}}{\text{maximize}} \quad & \tilde{\rho}_\varphi(\mathbf{p}^{(j)}, \mathbf{v}^{(j)}) - \boldsymbol{\varepsilon}^{(j)^\top} \mathbf{Q} \boldsymbol{\varepsilon}^{(j)} \\ \text{s.t.} \quad & |\mathbf{v}_k^{(j)}| \leq \mathbf{v}_{\max}^{(j)}, |\mathbf{a}_k^{(j)}| \leq \mathbf{a}_{\max}^{(j)}, \\ & \|\mathbf{a}_k^{(j)^\top} \mathbf{a}_k^{(j)}\|^2 \leq \boldsymbol{\varepsilon}_k^{(j)^\top} \boldsymbol{\varepsilon}_k^{(j)}, \boldsymbol{\varepsilon}_k^{(j)} \geq 0, \\ & \text{eq. (2)}, \forall k = \{0,1,\ldots,N-1\} \end{aligned} \quad . \quad (4)$$

The optimization problem both incorporates the satisfaction of the STL formula φ and the energy saving to prevent that the drones run out of battery while performing the mission, at the expense of a reduction of the robustness $\rho_\varphi(\mathbf{x})$.

D. Control architecture

The control architecture is reported in Fig. 1. Starting from mission and vehicle constraints, the *Motion Planner* solves

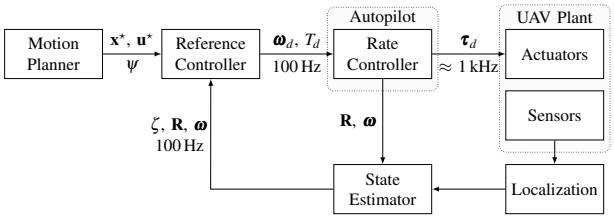

Figure 2: The control architecture. The *Motion Planner* supplies the trajectory $(\mathbf{x}^\star, \mathbf{u}^\star)$ and the heading angle ψ to the *Reference Controller*, which outputs the thrust T_d and angular velocities $\boldsymbol{\omega}_d$ for the embedded *Rate Controller*. A *State Estimator* provides the UAV translation and rotation (ζ, $\mathbf{R}$).

the optimization problem (3) supplying the trajectories $(\mathbf{x}^\star, \mathbf{u}^\star)$ and the heading angles ψ (provided as a constant reference for each target) for the q quad-rotors. The trajectory generation is run *one-shot*, i.e., once at time $\mathbf{t}_k = 0$, and the result is used as reference by the tracking controller.

In Fig. 2 the designed control architecture based on [19] is reported. This is divided into two parts: the high-level layer, i.e., *Reference Controller*, which generates the desired angular velocities $\boldsymbol{\omega}_d$ and thrust T_d command signals, by using the optimization outputs and the low-level layer, i.e., *Rate Controller*, which computes the propellers speed $\boldsymbol{\tau}_d$.

V. Numerical Results

To prove the validity and the effectiveness of the proposed approach, we carried out numerical simulations in MATLAB, extracting the needed STL specifications from the problem description (see Sec. II). At this stage, the vehicle dynamics and the trajectory tracking controller are not considered. The Gazebo robotics simulator was used in the second step to numerically verify the feasibility of the trajectories, exploiting the advantages of Software-in-the-loop simulations [21]. In particular, Gazebo simulations were used to reduce the probability of failures and to obtain a qualitative analysis of the system behavior. The framework was coded using the 2019b release of MATLAB, with the optimization problem formulated using CASADI library and NLP as solver. All simulations were performed on a laptop with an i7-8565U processor (1.80 GHz) and 32GB of RAM running on Ubuntu 18.04. Videos with the experiments and numerical simulations in MATLAB and Gazebo are available at http://mrs.felk.cvut.cz/ral-power-tower-inspection.

A. Power tower inspection

The task objective is to reach target regions (i.e., interesting areas to inspect) within the time interval $[0, {}^{2T}/_{3}]$ while staying within the workspace area ($[14\text{m} \times 18\text{m} \times 23\text{m}]$), avoiding possible collisions with the power tower and the obstacles along the path, and maintaining a safe distance ($\delta_{\min}$) between drones. The mission ends with the drones returning to the initial position within the time interval $[{}^{2T}/_{3}, T]$. When the drones reach the target regions, they start collecting images and videos by simulating a data acquisition process. To minimize the time required for inspection, the target regions are clustered to find a balance among the number of vehicles available for the inspection. However, advanced clustering algorithms may be used accounting for the drones positions and the distance between targets. For ease of experimentation, we considered only two drones and four target regions, but this does not imply a loss of generality of the approach. A numerical simulation was also carried out in MATLAB to show the feasibility of the problem as the number of drones and target regions increases (see Fig. 4). The task objective can be encoded with STL specifications as follows

$$
\begin{aligned}
&{}^{i}_{h}\varphi_{\text{dis}} = \square_{[0,T]}\left(\|{}^{i}\mathbf{p} - {}^{h}\mathbf{p}\| \geq \delta_{\min}\right),\\
&\varphi_{\text{lri}} = \bigwedge_{k,i,k\neq i}^{q}\left({}^{k}_{i}\varphi_{\text{dis}} \wedge {}^{k}\varphi_{\text{safe}} \wedge {}^{k}\varphi_{\text{ws}}\right) \bigwedge \Diamond_{[0,T]}\left(\left(\bigwedge_{k=1}^{q/w} {}^{k}\varphi_{\text{tr1}} \wedge {}^{k}\varphi_{\text{tr3}}\right.\right.\\
&\left.\left. \bigwedge_{k=q/w+1}^{q} {}^{k}\varphi_{\text{tr2}} \wedge {}^{k}\varphi_{\text{tr4}}\right)\mathcal{U}_{[2T/3,T]}\left(\bigwedge_{k=1}^{q} {}^{k}\varphi_{\text{home}}\right)\right),
\end{aligned}
\tag{5}
$$

where ${}^{i}_{h}\varphi_{\text{dis}}$ represents the safety distance requirement between the i-th drone and the h-th drone, φ_{ws}, φ_{safe}, and φ_{home} indicate the workspace, safety (i.e., avoiding collisions with the power tower and with obstacles along the path), and starting point specifications, respectively, w is the number of clusters, while φ_{tr1}, φ_{tr2}, φ_{tr3}, and φ_{tr4} are the target regions.

The scenario is depicted in Figs. 3 and 4 along with the obtained trajectories, obstacles and target regions (both the obstacles and the target regions are modeled as polyhedra). The 3D map was obtained from a three-dimensional terrestrial laser scan of the environment and contains an observation tower with a camera and some lights placed on top. These were chosen as regions of interest for the inspection. The tower is 20 m in height with a radius of 3 m. Table I reports the optimization problem and the parameter values for the drone considered in this paper, namely DJI F450. The optimization took 21 s to solve in the scenario with four target regions and two drones, and 43 s in the scenario with eight target regions and four drones.

	Sym.	Value-PT	Value-EA	Unit
LSE scaling factor	c	5	5	1
Drone safe distance	$\delta_{\min}$	3	3	m
Sampling period	T_s	0.05	0.05	s
Maximum velocity	$\mathbf{v}^{(j)}_{\max}$	3	3	m s^{-1}
Maximum acceleration	$\mathbf{a}^{(j)}_{\max}$	3	3	m s^{-2}
Trajectory duration	T	60	110	s

Table I: Optimization problem parameter values.

Gazebo simulations were performed to qualitatively and quantitatively analyze the time advantages deriving from the use of multiple drones to perform the inspection of a power tower w.r.t. using only one drone. The scenario reported in Fig. 4 was used as a testbed showing that the time required for the inspection took 60 s in the multi-UAV case and 255 s for the case of a single quad-rotor.

B. Energy-aware and event-triggered planner

As for the previous task, a pair of quad-rotors performs an inspection of a single power tower. The *power tower inspection* scenario was considered to also evaluate the performance of the *energy-aware* and *event-triggering* replanner. The

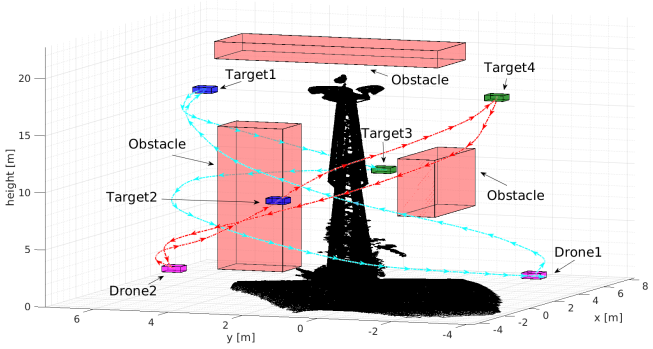

Figure 3: *Power tower inspection* scenario. Target regions are represented in blue and green and reflect the navigation order, respectively. Obstacles are depicted in red, while the starting points are in magenta.

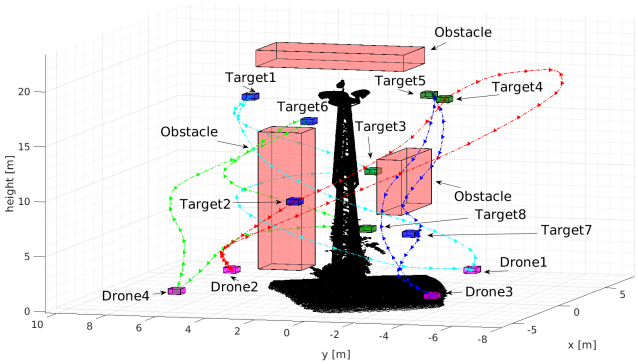

Figure 4: *Power tower inspection* scenario when considering four drones and eight target regions.

results of the numerical simulations carried out in MATLAB are reported in Figs. 5 and 6. As expected, the trajectories obtained considering the energy requirements (see Fig. 5) result closer to the obstacles than what happens when no energy requirement is enforced (see Fig. 3). This fact is motivated by the introduction of the energy saving cost term, at the expense of the overall robustness (see Figs. 10 and 11).

To validate the performance of the event-triggered replanner, we simulated the presence of two major unexpected disturbances deviating at runtime the quad-rotors from their original planned paths. Once the replanner detects major deviations from the planned trajectory (i.e., $|\tilde{\mathbf{p}}_l - \mathbf{p}_l| > \eta$), a partial replanning is triggered online to bring back the quad-rotors to next "topic" waypoint, as illustrated in Fig. 6. Then, the result is used as reference for the tracking controller. The optimization took less than 1 s for both disturbances.

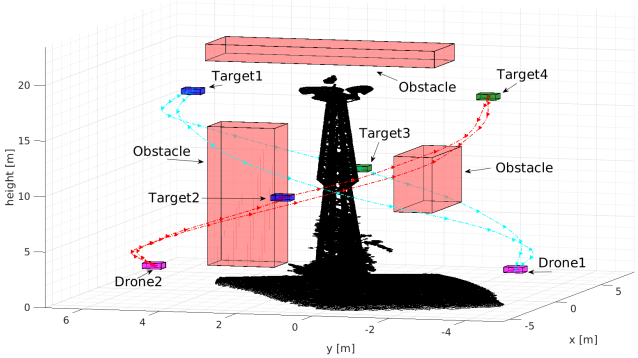

Figure 5: *Power tower inspection* scenario considering the energy-aware motion planner.

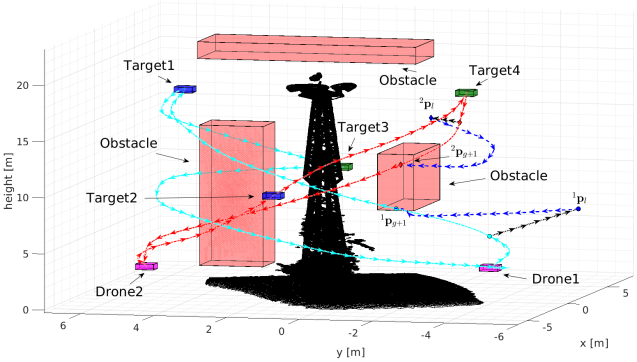

Figure 6: *Power tower inspection* scenario in case of unexpected large disturbances events at runtime. Arrows represent the drones' path along the mission. In black the deviation from the original, in blue the new path computed by the replanner.

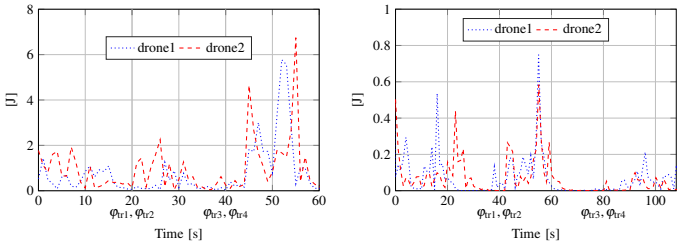

Figure 7: Energy consumption profiles with two quad-rotors performing the *power tower inspection*. From left to right: "drone1" and "drone2" data when considering the "basic" (3) and the energy-aware (4) motion planner, respectively.

Gazebo simulations were performed to evaluate the decrease in energy when using the trajectories obtained with the energy-aware motion planner, as shown in Fig. 11. As can be seen from the graph, the velocity and acceleration signals are still within the bounds and assume lower pick-values than Fig. 10. However, smaller values of the robustness are obtained. In Fig. 7 the energy consumption profiles are reported for the two problem formulations.

C. Comparison with kinodynamic RRT⋆

As described in Sec. I-A, various state-of-the-art solutions investigate the path planning problem in quad-rotors inspection scenarios. However, not all of them are suitable for the inspection of power line infrastructure. This section aims to compare the set up optimization problem (3) with the kinodynamic RRT⋆ proposed in [6]. Analogously to what done in our paper, the incremental sampling approach in [6] finds quad-rotor trajectories so as to remain in the workspace, avoid obstacles and incorporate bounds on the control inputs. Analogously to (2), the optimal trajectories are derived in terms of a solution

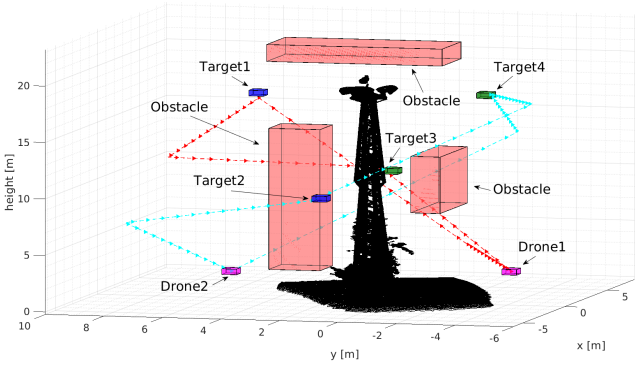

Figure 8: *Power tower inspection* scenario when considering the kinodynamic RRT*.

of a $2n^2$-degree polynomial. In this case the dynamics of the quad-rotor is linearized around the hovering state constraining the yaw (and its derivatives) to zero [6, eq. (31)]. The obtained trajectories are reported in Fig. 8.

It is worth noticing that the kinodynamic RRT* was not originally meant to work with multi-robot systems, making the specification of a minimum distance between drones difficult. To overcome this problem, we adapted [6] to the multi-robot case via iterating on the various quad-rotors considering the path obtained for the first $\{q-1\}$ quad-rotors as forbidden flight corridors for the q quad-rotor. Another important difference is that the notion of time is not explicitly taken into account. Besides this, mission specifications such as the minimum distance between drones, cannot be easily codified in the optimization problem. Furthermore, while the approach in [6] adopts a linearized model of the quad-rotor, in our case we do not impose such simplification. From a trajectory viewpoint, this implies for [6] spikes and corners difficult to follow in reality, possibly leading to sudden stresses on the actuators to respond to rather fast changes of direction. This results not only in possibly errors on trajectory tracking, with the possibility of violating mission specifications and safety requirements (e.g., the quad-rotor could collide with the power tower), but also in high energy consumption that could harm the mission. In addition to that, the demand on the onboard control system is higher, since it requires top performances. The algorithm took 8 s to find a solution for the problem, while the comparison between the drone and desired trajectory from Gazebo simulations is depicted in Fig. 12.

VI. Experimental Results

To evaluate and prove the applicability of the proposed approach in real-world autonomous inspection tasks, experiments with a DJI F450 quad-rotor were performed (see Fig. 9). Real flight tests verified not only the fulfillment of the STL specifications (i.e., φ_{tr1}, φ_{tr2}, φ_{tr3}, and φ_{tr4}), but also the compliance with trajectory generation requirements (i.e., maximum velocity $\mathbf{v}_{\max}^{(j)}$ and acceleration $\mathbf{a}_{\max}^{(j)}$, and safe distance $\delta_{\min}$). The STL motion planner (see Sec. IV) was implemented in MATLAB and the obtained trajectories were sent to onboard PCs before running the experiment.

VII. Conclusions

This paper has presented a framework for encoding power line inspection missions for a fleet of quad-rotors as STL specifications. In particular, an optimization problem was set to generate optimal strategies that satisfy such specifications accounting also for vehicle constraints. Further, an event-triggered replanner and an energy minimization problem have been proposed to reply to external disturbances and to enhance quad-rotors energy saving during the mission. The approach enables the use of complex and rich task specifications with automated trajectory generation. The solution potentially mitigates computational complexity explosion issues, thanks to the use of a smooth approximation of the robust semantic. The numerical simulations in MATLAB and Gazebo and also the experimental results proved the validity and the effectiveness of the proposed approach, demonstrating its applicability in real world scenarios. Future work includes investigating better solutions to cluster target regions accounting for drone positions and the distance between targets and extending the event-triggered replanner with an online search of the best partial reconnection trajectory or, if complete specification satisfaction is no longer possible, the minimum violation replanning.

References

[1] J. Major *et al.*, "Emerging and future inspection of overhead transmission lines," EPR Institute, Tech. Rep., 2011, no. 1021876.

[2] H. Baik and J. Valenzuela, "Unmanned Aircraft System Path Planning for Visually Inspecting Electric Transmission Towers," *Journal of Intelligent & Robotic Systems*, vol. 95, pp. 1097–1111, 2018.

[3] C. Martinez *et al.*, "The Power Line Inspection Software (PoLIS): A versatile system for automating power line inspection," *Engineering Applications of Artificial Intelligence*, vol. 71, pp. 293–314, 2018.

[4] A. Donzé and O. Maler, "Robust Satisfaction of Temporal Logic over Real-Valued Signals," in *Formal Modeling and Analysis of Timed Systems*, K. Chatterjee and T. A. Henzinger, Eds., 2010, pp. 92–106.

[5] O. Maler and D. Nickovic, "Monitoring Temporal Properties of Continuous Signals," in *Formal Techniques, Modelling and Analysis of Timed and Fault-Tolerant Systems*, 2004, pp. 152–166.

[6] D. J. Webb *et al.*, "Kinodynamic RRT*: Asymptotically optimal motion planning for robots with linear dynamics," in *IEEE International Conference on Robotics and Automation*, 2013, pp. 5054–5061.

[7] C. Belta, B. Yordanov, and E. A. Gol, *Formal Methods for Discrete-time Dynamical Systems*. Springer, 2017.

[8] G. E. Fainekos and G. J. Pappas, "Robustness of temporal logic specifications for continuous-time signals," *Theoretical Computer Science*, vol. 410, no. 42, pp. 4262–4291, 2009.

[9] A. Pagnano, M. Höpf, and R. Teti, "A Roadmap for Automated Power Line Inspection. Maintenance and Repair," *Procedia CIRP*, vol. 12, pp. 234–239, 2013.

[10] H. Chen, Z. He, B. Shi *et al.*, "Research on Recognition Method of Electrical Components Based on YOLO V3," *IEEE Access*, vol. 7, pp. 157 818–157 829, 2019.

[11] S. S. Mansouri, C. Kanellakis, E. Fresk *et al.*, "Cooperative coverage path planning for visual inspection," *Control Engineering Practice*, vol. 74, pp. 118–131, 2018.

[12] Y. Shoukry, P. Nuzzo, A. Balkan *et al.*, "Linear temporal logic motion planning for teams of underactuated robots using satisfiability modulo convex programming," in *IEEE Conference on Decision and Control*, 2017, pp. 1132–1137.

[13] C. E. Luis *et al.*, "Online Trajectory Generation With Distributed Model Predictive Control for Multi-Robot Motion Planning," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 604–611, 2020.

[14] W. Hönig, *et al.*, "Trajectory Planning for Quadrotor Swarms," *IEEE Transactions on Robotics*, vol. 34, no. 4, pp. 856–869, 2018.

[15] Y. V. Pant *et al.*, "Smooth operator: Control using the smooth robustness of temporal logic," in *IEEE Conference on Control Technology and Applications*, 2017, pp. 1235–1240.

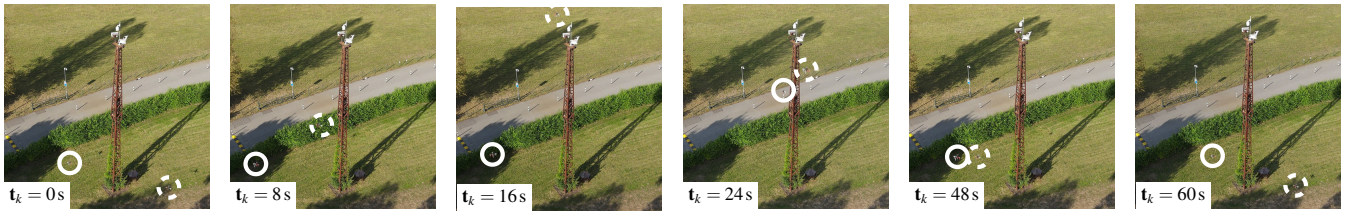

Figure 9: Snapshots of the *power tower inspection* scenario. The system evolution at different time instants $\mathbf{t}_k$ is reported. Solid and dashed circles are used to indicate "drone1" and "drone2", respectively.

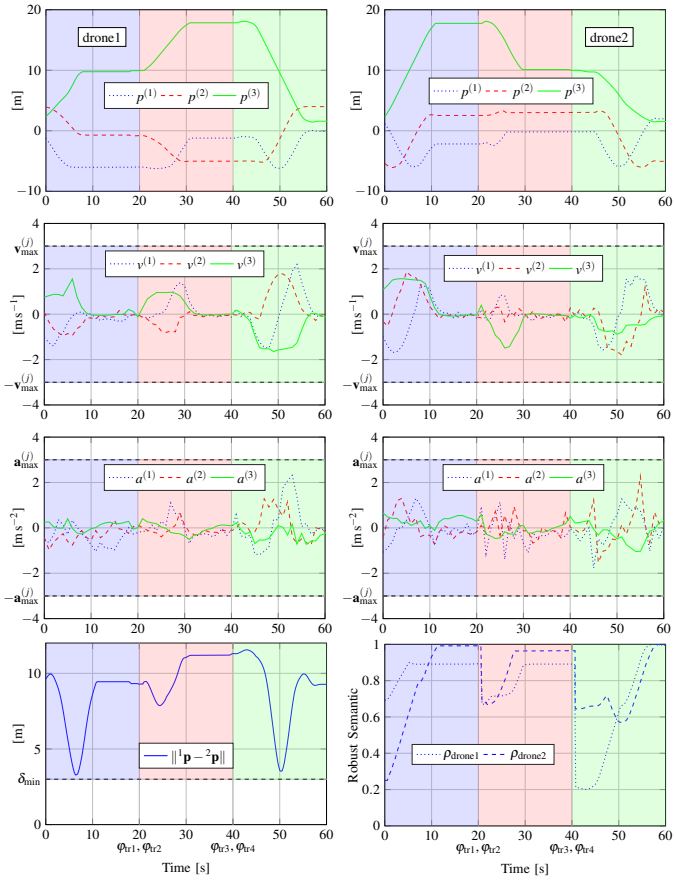

Figure 10: Position, linear velocity and acceleration, and mission requirements considering the "basic" motion planner. From left to right: "drone1" and "drone2" data. The STL specifications (φ_{tr1}, φ_{tr2}, φ_{tr3}, and φ_{tr4}) are also reported with different color regions.

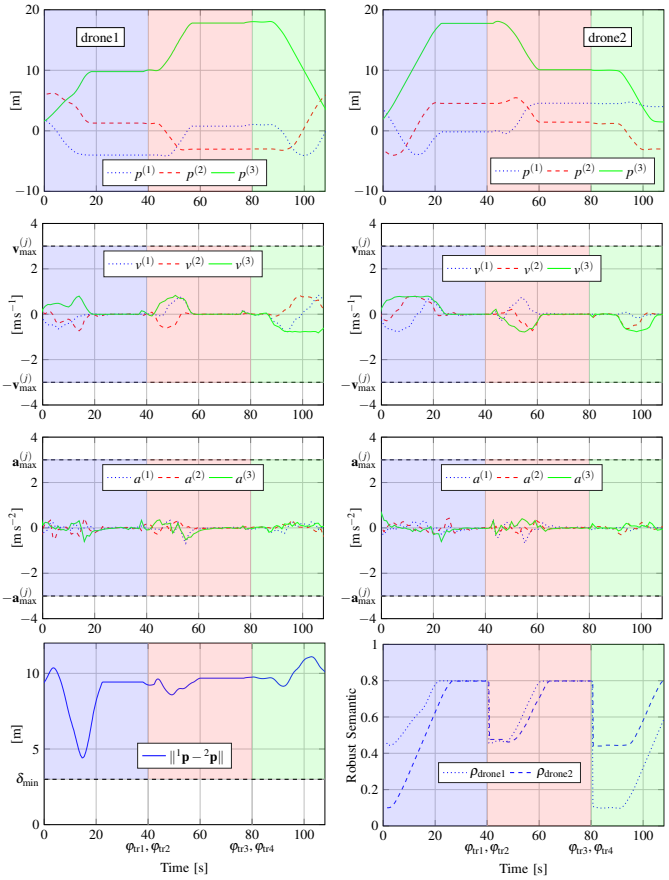

Figure 11: Position, linear velocity and acceleration and, mission requirements when considering the energy-aware motion planner performing the *power tower inspection*.

[16] J. Park *et al.*, "Fast Trajectory Planning for Multiple Quadrotors using Relative Safe Flight Corridor," in *IEEE International Conference on Intelligent Robots and Systems*, 2019, pp. 596–603.
[17] V. Raman, A. Donzé *et al.*, "Model predictive control with signal temporal logic specifications," in *IEEE Conference on Decision and Control*, 2014, pp. 81–87.
[18] M. W. Mueller, M. Hehn, and R. D'Andrea, "A Computationally Efficient Motion Primitive for Quadrocopter Trajectory Generation," *IEEE Transactions on Robotics*, vol. 31, no. 6, pp. 1294–1310, 2015.
[19] T. Baca, M. Petrlik, M. Vrba *et al.*, "The MRS UAV System: Pushing the Frontiers of Reproducible Research, Real-world Deployment, and Education with Autonomous Unmanned Aerial Vehicles," 2020. [Online]. Available: https://arxiv.org/pdf/2008.08050
[20] H. Abbas, G. Fainekos, S. Sankaranarayanan *et al.*, "Probabilistic Temporal Logic Falsification of Cyber-Physical Systems," *ACM Transaction on Embedded Computing Systems*, vol. 12, no. 2s, 2013.
[21] G. Silano *et al.*, "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.

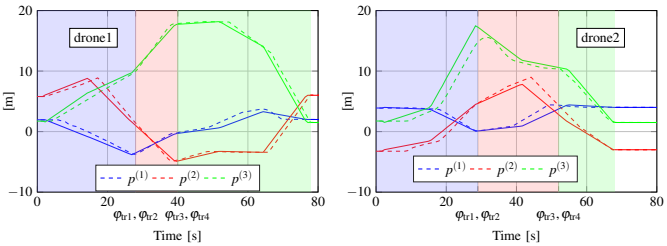

Figure 12: Drone positions when considering the kinodynamic RRT*. Solid lines represent the computed path, while dashed lines are the trajectories performed by the quad-rotors. Color regions help understand when a part of the mission is accomplished in terms of the corresponding STL case.