A Multi-Layer Software Architecture for Aerial Cognitive Multi-Robot Systems in Power Line Inspection Tasks

Giuseppe Silano[1], Jan Bednar[1], Tiago Nascimento[1,2], Jesus Capitan[3], Martin Saska[1], and Anibal Ollero[3]

***Abstract*— This paper presents a multi-layer software architecture to perform cooperative missions with a fleet of quadrotors providing support in electrical power line inspection operations. The proposed software framework guarantees the compliance with safety requirements between drones and human workers while ensuring that the mission is carried out successfully. Besides, cognitive capabilities are integrated in the multi-vehicle system in order to reply to unforeseen events and external disturbances. The feasibility and effectiveness of the proposed architecture are demonstrated by means of realistic simulations.**

***Index Terms*— Software architecture, multi-UAV system, cognitive robots, power line inspection**

I. Introduction

Over the last two decades, the global energy demand has increased rapidly due to demographic and economic growth, especially in emerging market areas. 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. Damaged transmission lines, usually due to high winds, storms, or inefficient inspection campaigns [1], is one of the major causes of power outages.

Nowadays, the most common strategy for reducing energy interruptions is to schedule periodic maintenance activities by carrying out repairs and replacements on active lines (see, Fig. 1). This is the most suitable method when system integrity, reliability, and operating revenues are essential, and when the removal of a circuit is not acceptable [1]. Manned helicopters and experienced crews take care of acquiring data over thousands of kilometers. Conductive suits, climbing harnesses, and arc control rods prevent operators from getting shocked while working on transmission lines. However, there are two major drawbacks of this approach: first, inspection and maintenance are dangerous for operators who work close to power towers and operate on electrified lines; second, these operations are extremely time-consuming and expensive ($1,500 for a one-hour flight) and prone to human error [2].

Fig. 1: An operator climbing back to the helicopter after performing maintenance on the conductors.

Therefore, there is a clear need for safe and practical techniques that enable more efficient maintenance and inspection procedures in electrical power lines, in order to reduce potential risks and costs for the distribution companies. Multiple solutions have been proposed in the literature for automating this task [3], but the most promising and flexible alternative is to use Unmanned Aerial Vehicles (UAVs), as they are capable of supporting inspection at different levels [1]. For instance, UAVs can inspect places of difficult access, but they can also monitoring human operations for safety purposes.

However, the use of UAVs for inspection tasks in power lines is particularly challenging due to issues like their limited battery capacity, the strong electromagnetic interference produced by power lines, and the presence of potential obstacles along the lines (e.g., branches, vegetation, marker balls) [2]. Enhanced systems with cognitive capabilities, e.g. based on novel perception sensors, such as event cameras [4], advanced data fusion techniques [5], or fast on-line planning [6], are of interest to address those complexities and to accomplish the assigned mission safely and successfully. More precisely, it is key for the system to integrate UAVs capable of processing the acquired knowledge of the surrounding environment and of planning and executing appropriate actions in reaction to unforeseen events and external disturbances. Furthermore, these actions should be performed in a cooperative manner by the fleet of UAVs while ensuring the compliance with safety requirements.

[1]Giuseppe Silano, Jan Bednar, Tiago Nascimento, and Martin Saska are with the Faculty of Electrical Engineering, Czech Technical University in Prague, Czech Republic (email: {silangiu, jan.bednar14, pereiti1, martin.saska}@fel.cvut.cz).

[2]Tiago Nascimento is with the Lab of Systems Engineering and Robotics (LASER), Department of Computer Systems, Universidade Federal da Paraiba, Brazil (email: tiagopn@ci.ufpb.br).

[3]Jesus Capitan and Anibal Ollero are with the GRVC Robotics Laboratory, University of Seville, Spain (email: {jcapitan, aollero}@us.es).

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 project no. 20-10280S.

platform	open source	modular	simulation	outside laboratory	cognitive architecture	multi-frame localization	rate output	software last update	reference
MRS UAV system	+	+	+	+	+	+	+	2020	[7]
Aerostack	+	+	+	+	-	-	-	2020	[8]
XTDrone	+	+	+	-	-	-	-	2020	[9]
RotorS	+	+	+	-	-	-	+	2020	[10]
ReCOPTER	+	-	-	-	-	-	-	2015	[11]
MAVwork	+	+	-	-	-	-	-	2013	[12]

TABLE I: Comparison of available open-source frameworks for UAV.

Versatile and reliable software architectures are essential to integrate these cognitive multi-UAV systems with many interconnected heterogeneous components (e.g., path planners, control and computer vision algorithms, task managers). Therefore, in this paper, we propose such an architecture to deal with multi-UAV missions in the context of power line inspection and human safety.

A. Related works

UAV frameworks have been proposed over the last 15 years as valuable tools for inspection and surveillance purposes [13], soil and field analysis and crop monitoring [8]. However, the design of a complete software stack for fully autonomous multi-UAV systems is still an open problem involving multiple interconnected aspects, such as the design of guidance and navigation, control systems [7], [14], and the development of a reliable communication network [15]. Several commercial and open-source projects have been proposed over the years to develop ready-to-use hardware and software architectures for multi-rotor vehicles [16]. Nevertheless, most of the shared code is not well documented, making it difficult to reuse components, to add new features, or to replicate the research results. These software projects do not offer guarantees on the reliability of the code or customization possibilities.

Table I summarizes the most popular software frameworks available in the literature. MAVwork [12] was one of the first frameworks for aerial vehicles in 2011. Two years later, ReCOPTER [11], [17] was released, which is an open-source framework for research and educational activities with multi-rotor vehicles. Although the software was released as supporting material for the paper, no further updates have been released since then, with the consequent issue for code reusability [16]. The first up-to-date framework to be mentioned is the RotorS [10] simulator. This work provides Gazebo-based simulations for numerous heterogeneous vehicles (e.g., the Ascending Firefly Hexarotor, the Parrot AR.Drone, the VoliroX). However, the control pipeline features are too basic, with little potential to be applied to real-world conditions. Another UAV framework is XTDrone [9], which offers a simulation testbed with many complex functionalities, including simulation of onboard sensors and complex localization systems. The control pipeline relies entirely on the PX4 embedded control software, which limits its application to other hardware platforms. In contrast, the Aerostack system [8], designed for the deployment of multi-rotor UAVs, is continuously being updated, and offers a straightforward transition from simulation to real experiments. However, this system is based on the DJI flight controller, whose control inputs are limited to orientation and thrust commands. Furthermore, the system lacks the feature of switching between multiple frames of reference.

B. Contributions

In this paper, we propose a multi-layer software architecture designed for the AERIAL-CORE European project[1], in which cognitive aerial platforms are being developed inspired by the application of autonomous power line inspection. The proposed software framework relies on the UAV platform in [7] and is built on top of the Robot Operating System (ROS) shell. The Nimbro network[2] is used to support communication between vehicles by implementing the *TCP Fast-Open protocol* and reducing bandwidth using the *libbz2* data compression algorithm.

In comparison to the frameworks described in Section I-A, the advantages are threefold: (i) besides the RotorS and Aerostack systems, no other existing platform provides a full-stack framework for multi-rotor UAVs that is actively maintained and supports simulation with a fleet of aerial vehicles; (ii) the integration of the Nimbro network ensures that the ROS messages are transmitted at the published rate, a crucial requirement in multi-UAV applications; and (iii) none of the above-mentioned frameworks present cognitive functionalities at the level required by the applications in the AERIAL-CORE project. For instance, these cognitive capabilities should enable the system to adapt to information learned online, e.g., UAVs failing or battery levels dropping faster than expected.

Gazebo simulations show the validity and effectiveness of the proposed platform, which is released as open-source[3]. These preliminary simulations showcase the capabilities and potential of the architecture. Moreover, they demonstrate an advanced level of integration of the whole multi-UAV system in the AERIAL-CORE project.

II. Problem Description

Two tasks of interest are considered: (i) *inspection*, where a fleet of multi-rotor UAVs carries out a detailed investigation of power equipment autonomously, helping the human workers to acquire views of the power tower that are not easily

[1] https://aerial-core.eu

[2] https://github.com/ctu-mrs/nimbro_network

[3] https://github.com/ctu-mrs/icuas_2021_sw_architecture_acws

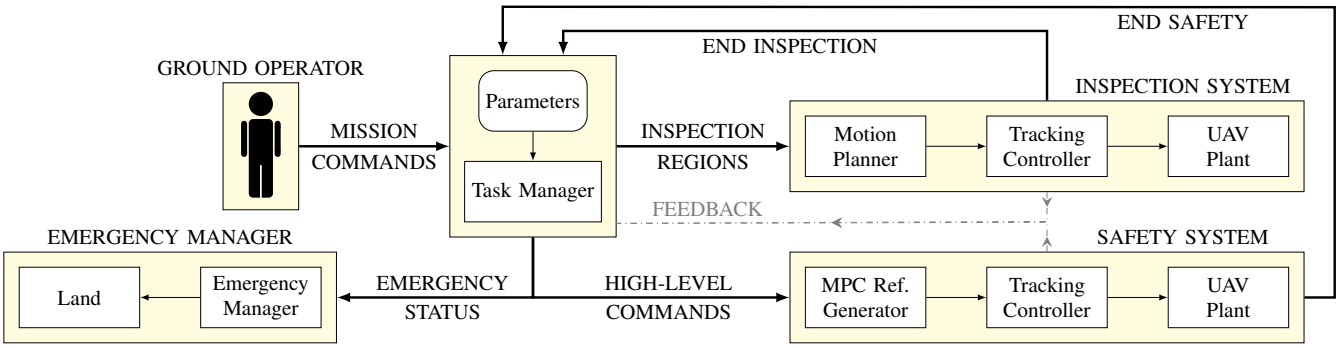

Fig. 2: Proposed software platform architecture. Arrows represent the data exchanged among blocks.

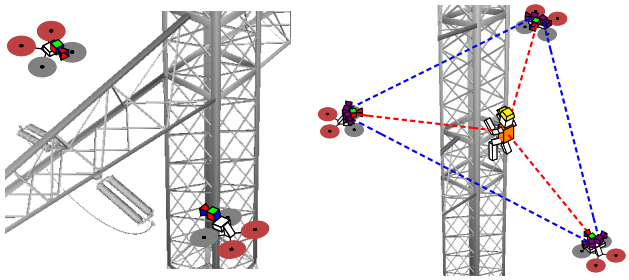

Fig. 3: From left to right: *inspection* and *safety* scenarios. Onboard cameras and sensors acquire data of the power equipment. A mutual localization system helps to maintain the formation avoiding contact with the tower.

accessible (see, Fig. 3a); and (ii) *safety*, where a formation of UAVs provides the supervising team with a view of the humans working on the power tower in order to monitor their status and to ensure their safety (see, Fig. 3b).

In both tasks, visual sensors are essential to perform the required work. In the UAV configuration, cameras are mounted in an *eye-in-hand* configuration, i.e., rigidly attached to the body frame. For the *inspection* task, cameras capture images of the power tower searching for damage to the mechanical structure and for failures of the electrical components. In the *safety* task, a visual servoing scheme is implemented to keep track of the movements and actions of the human workers throughout the entire operation. Camera images are also used to mutually localize the UAVs in the surrounding environment [5].

Finally, we assume that the UAVs operate in an environment represented by a previously acquired map, including the position of the power towers and other potential obstacles. We also assume that the UAVs are equipped with the necessary sensors and software for precise self-location and state estimation [7].

III. Software Framework

The software platform is organized into three layers of abstraction that provide high modularity and flexibility, while reflecting the problem description: the *task manager* (Section III-A), and the *inspection* (Section III-B) and the *safety* (Section III-C) systems. The framework controls the behavior of the UAVs and can be visualized as a chain of various software components working together to fulfill the mission specifications. The *task manager* and the *motion planner* block inside the *inspection* system are placed at a ground station. In contrast, all other blocks and layers of the proposed system run onboard the UAVs as a distributed system. The framework enables the UAVs through the *task manager system* to execute the *inspection* and *safety* tasks. Continuous feedback from the UAVs guarantees a certain degree of reliability against unexpected behaviors stopping the mission in case of problems. Figure 2 presents the proposed software architecture.

A. Task manager

The *task manager* is the core cognitive block of the system and it is in charge of implementing high-level behaviors, by computing multi-UAV cooperative plans and allocating tasks to different UAVs coping with their heterogeneity and their battery levels. Each UAV is assumed to have different onboard sensors or configurations in order to be more suitable for a specific task. This block implements planning algorithms on demand, so that the team of UAVs is able to support the human operators working on an electrical tower; but it is also capable of re-planning online to react to new learnt circumstances.

Therefore, the *task manager* has the cognitive capability to learn policies online using available information and to react to unexpected events not present in the initial plan (e.g., UAV failures, shorter flight time than expected, etc.). This is achieved through a high-level planner that computes initial plans given UAV constraints in terms of battery limits and heterogeneous capabilities. Then, the block monitors the plan execution continuously, integrating information perceived by the UAVs, in order to re-plan online if the initial plan is no longer feasible. Besides, *emergency* maneuvers are commanded for those UAVs failing, so that they land safely.

The ground operator is the first input of the system, providing a complete mission to be performed by the fleet of UAVs. Given that mission, the *task manager* computes a plan to implement the required actions with the team. These tasks consist of parametric high-level commands for different *inspection* and *safety* activities. As basic primitives,

the UAVs can be commanded to stay idle or to go to a known recharging station. Also, in case of a low battery level, an emergency landing is commanded. The other two high-level commands are: (i) to perform an *inspection* task, with the regions to be inspected encoded as a sequence of target regions; and (ii) to perform a *safety* task, with the identifier of the worker to be monitored and the geometry of the formation (i.e., the distance to the worker, the viewing angles, and the inter-UAV angles) as parameters. These high-level commands are sent out to the lower layers of the architecture, which are then in charge of implementing the corresponding behaviors by means of low-level controllers that deal with multi-UAV navigation and formation control.

The *task manager* also receives feedback information from the UAVs about their battery level, localization and target state estimation, in order to generate its learnt representation of the environment. This information is used to decide when a UAV has failed and needs to be landed urgently, and when the current plan is not feasible anymore and new task assignments are required, for instance to replace a UAV that may run out of battery before finishing its current task.

Regarding the underlying algorithm for high-level planning, our implementation consists of a Behavior Tree that encodes the constraints imposed by UAV battery levels and heterogeneous capabilities, trying to allocate tasks to UAVs with the objective of minimizing total travel time.

B. Inspection system

The *inspection* system is in charge of computing feasible and constrained trajectories for a fleet of q multi-rotors that have been assigned an *inspection* task together. This is done by leveraging on Signal Temporal Logic (STL) [18] specifications to perform the inspection of a power tower. Such a logic allows for planning and executing appropriate actions, starting from (possibly vague) high-level task specifications (e.g., the UAVs should reach the goal within 10 time units while always avoiding obstacles). In particular, STL can be used to describe planning objectives that are more complex than point-to-point planning algorithms (e.g., A*, RRT*).

An optimization problem (1) is formulated to generate feasible dynamic trajectories that satisfy these specifications and also take vehicle constraints into account, i.e., the maximum velocity and acceleration of the vehicles, by using the motion primitives defined in [19]. The planner [18] enables the formulation of complex missions that avoid obstacles and maintain a safe distance between UAVs while performing the inspection within a given time. When the UAVs reach the target regions, they start collecting images and videos and acquiring data from the onboard sensors. Both the inspection time and target regions as well as other high-level commands, such as stay idle or recharge, are defined by the *task manager* and encoded in the STL formula (φ) before the mission starts. Figure 4 describes the overall *inspection* control architecture.

Let $T_s \in \mathbb{R}_{\geq 0}$ and $T \in \mathbb{R}_{\geq 0}$ be the sampling period and the trajectory duration, respectively; we can write the time interval as the vector $\mathbf{t} = (0, T_s, \ldots, NT_s)^\top \in \mathbb{R}^{N+1}$, where

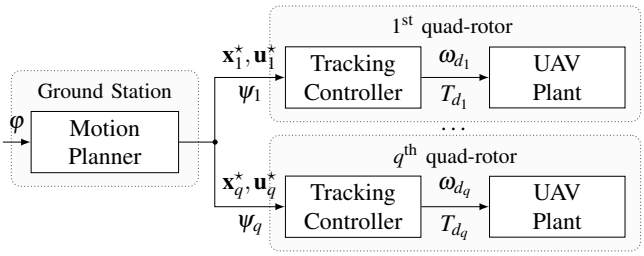

Fig. 4: *Inspection* control scheme. Starting from STL formula φ, the motion planner generates the trajectories $(\mathbf{x}_i^\star, \mathbf{u}_i^\star)$ and the heading angles ψ_i, with $i = \{1, \ldots, q\}$, for the q UAVs. A trajectory tracking controller supplies the desired angular velocities ω_{d_i} and thrust T_{d_i} commands for the UAVs.

$NT_s = T$ and $\mathbf{t}_k$, $k \in \mathbb{N}_{\geq 0}$, denote the k-element of the vector $\mathbf{t}$. Similarly, let us define the state $\mathbf{x}$ and control $\mathbf{u}$ sequences of the system 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 $\mathbf{p}_k^{(j)}$, $\mathbf{v}_k^{(j)}$, $\mathbf{a}_k^{(j)}$, with $j = \{1, 2, 3\}$, represent the vehicle's position, velocity, and acceleration at time instant k along the j-axis for the inertial frame. Finally, let us consider the STL formula φ and its smoothed robustness version $\tilde{\rho}_\varphi(\mathbf{x}, \mathbf{t}_k)$. The optimization problem can be formalized as follows:

$$\begin{aligned} \underset{\mathbf{p}^{(j)}, \mathbf{v}^{(j)}, \mathbf{a}^{(j)}}{\text{maximize}} \; & \tilde{\rho}_\varphi(\mathbf{p}^{(j)}, \mathbf{v}^{(j)}) \\ \text{s.t.} \quad & |\mathbf{v}_k^{(j)}| \leq \mathbf{v}_{\max}^{(j)}, |\mathbf{a}_k^{(j)}| \leq \mathbf{a}_{\max}^{(j)}, \\ & [18,\ \text{eq. } 2], \forall k = \{0, 1, \ldots, N-1\} \end{aligned} \quad , \tag{1}$$

where $\mathbf{v}_{\max}^{(j)}$ and $\mathbf{a}_{\max}^{(j)}$ are the desired maximum values of velocity and acceleration along the motion, respectively. The heading angles (ψ) are provided as a constant reference for each target region. Further details are available in [7], [18].

C. Safety system

The *safety* system aims to control a team of g UAVs that have been assigned a *safety* task to provide views of a human working on the power tower in order to monitor his status while ensuring compliance with safety requirements. A Model Predictive Control (MPC) framework based on [14], [20], [21] deals with generating an optimal trajectory ($\mathbf{p}^\star$, $\mathbf{v}^\star$, and $\mathbf{a}^\star$) for the vehicles avoiding collisions with the power tower and obstacles, considering the UAVs physical constraints (i.e., maximum velocity and acceleration), and keeping a viewing angle with respect to the worker in the camera.

A real-time implementation of the optimization problem (2) is considered to cope with a *safety* mission. Through the *task manager*, the ground operator can provide high-level commands to the aerial vehicles so that they change their viewing angles or the shape of their formation (i.e., their distance to the worker or their inter-UAV angles). The mutual localization system in [5] helps to maintain the UAV formation by complementing the GPS vehicle positions when electromagnetic interference generated by the power tower are not negligible.

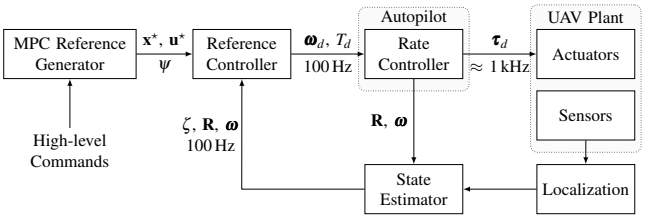

Fig. 5: *Safety* control architecture. The *MPC Reference Generator* supplies the trajectory $(\mathbf{x}^\star, \mathbf{u}^\star)$ and the heading angle ψ to the *Reference Controller*, which outputs the thrust T_d and angular velocities $\boldsymbol{\omega}_d$ for the autopilot that computes the propellers speed $\boldsymbol{\tau}_d$ for the *Actuators*. A *State Estimator* provides the UAV translation and rotation (ζ, $\mathbf{R}$, $\boldsymbol{\omega}$).

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 $f\colon X \times U \to X$ is differentiable in both the arguments. The initial state is denoted by x_0 and takes values from some initial set $X_0 \subset \mathbb{R}^n$. Let us also assume that z_k is the state vector of the perception system (e.g., the projections of the worker's 3D points on the image plane, the target position, etc.), and σ a set of parameters characterizing them. The perception and the vehicle states are coupled through the UAV dynamics, namely $z_{k+1} = f_p(x_k, u_k, \sigma)$. Hence, the discrete-time optimization problem over a receding horizon T_h, sampled in W shooting points, at a given instant t, can be expressed as:

$$\begin{aligned} \underset{\mathbf{x},\mathbf{u},\mathbf{z}}{\text{minimize}} \quad & J(\mathbf{x},\mathbf{u}) + J_p(\mathbf{z}) \\ \text{s.t.} \quad & r(\mathbf{x}_k,\mathbf{u}_k,\mathbf{z}_k) = 0, \\ & h(\mathbf{x}_k,\mathbf{u}_k,\mathbf{z}_k) \leq 0 \end{aligned} \quad , \tag{2}$$

where $J(\mathbf{x},\mathbf{u})$ and $J_p(\mathbf{z})$ are the action and perception objective functions, respectively, while $r(\mathbf{x}_k,\mathbf{u}_k,\mathbf{z}_k)$ and $h(\mathbf{x}_k,\mathbf{u}_k,\mathbf{z}_k)$ represent equality and inequality constraints that the solution should satisfy for perception, action, or both simultaneously, respectively. Roughly speaking, we encode the action objective $J(\mathbf{x},\mathbf{u})$ ensuring the minimum distance between the UAV and the human worker, while being compliant with safety requirements (i.e., maintaining a certain distance between UAVs and worker, bounding maximum velocity and acceleration of the vehicle). Such requirements are included as hard constraints in the $r(\mathbf{x}_k,\mathbf{u}_k,\mathbf{z}_k)$ and $h(\mathbf{x}_k,\mathbf{u}_k,\mathbf{z}_k)$ functions. First and second time derivative of tracking error are also considered to avoid discontinuities in the UAV behavior. On the other side, we integrate the visibility constraint in $J_p(\mathbf{z})$ such that the visibility cone originated by the camera view[4], especially its horizontal and vertical projections, includes at the best the human features. The optimization problem relies on the assumption that the system is deferentially flat. This allows for simplification of the optimization problem, transforming the nonlinear dynamics of the g UAVs in an equivalent linear description of the system. Figure 5 describes the overall system architecture, while more details are available in [20], [21].

[4]The *pinhole camera model* is taken into account for the vision sensor.

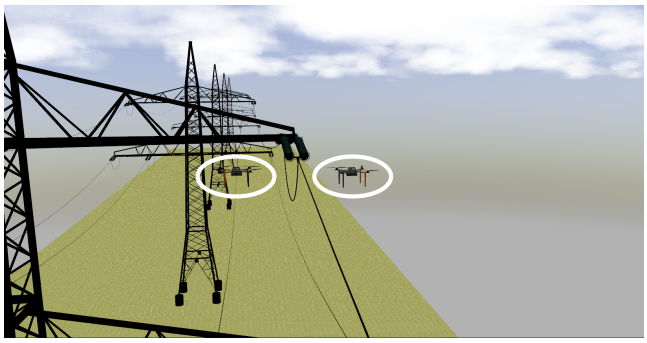

Fig. 6: Snapshot of the *inspection* scenario. Solid circles show the UAVs approaching an insulator.

IV. Simulation Results

In this section, we show some preliminary simulation results to demonstrate the feasibility and the effectiveness of the proposed software architecture. In particular, we simulated the system in a realistic scenario by using the Gazebo robotic simulator, exploiting the advantages of Software-in-the-loop simulations [22]. Our objective is also to show an advanced level of integration of the architecture. The framework was coded by using the Melodic Moreina release of ROS with the optimization problems formulated using the CASADI library[5] and NLP[6] and CVXGEN[7] as solvers. All simulations were performed on a laptop with an i7-8565U processor (1.80 GHz) and 32GB of RAM running on Ubuntu 18.04. Figures 6 and 7 depict snapshots of an *inspection* and a *safety* mission, respectively. Videos with the two simulations can be found at `http://mrs.felk.cvut.cz/software-architecture-acws`.

Both the *inspection* and *safety* scenarios consist of a series of power towers, each one with up to twelve insulators. The towers are 20 m high with a radius of 15 m. The presence of wires between the towers is also considered to simulate a scenario quite close to the real application. The STL models and the mesh files have also been made available[3]. For the sake of simplicity and ease of experimentation, we considered only one tower and six target regions for the *inspection* mission, but this does not imply a loss of generality of the architecture.

In the *inspection* scenario (see, Fig. 6), the *task manager* provides the inspection order of the insulators considering the battery limits and the heterogeneous capabilities of the vehicles before starting the mission. It also sets the desired maximum values of velocity and acceleration of the vehicles. Then, the optimization problem (1) is solved to provide the dynamic feasible trajectories for the UAVs. In the presented experimental results, a safety distance of 1 m is maintained

[5]`https://web.casadi.org`
[6]`http://cvxr.com`
[7]`https://cvxgen.com`

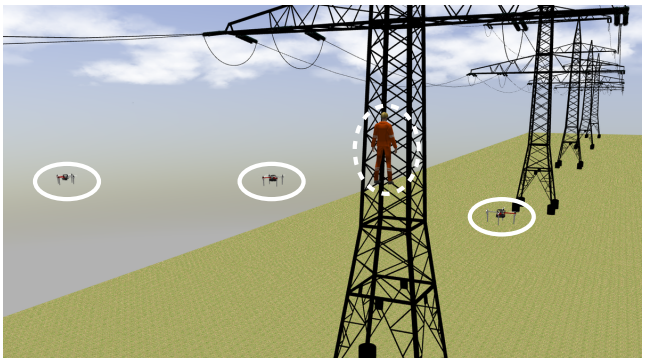

Fig. 7: Snapshot of the *safety* scenario. Solid and dashed circles indicate the UAVs and the operator, respectively.

between the UAVs, while the maximum velocity $\mathbf{v}_{\max}^{(j)}$ and acceleration $\mathbf{a}_{\max}^{(j)}$ are set to $3\,\mathrm{m\,s^{-1}}$ and $2.5\,\mathrm{m\,s^{-2}}$, respectively. The maximum velocity and acceleration values were chosen in accordance with the camera constraints to avoid producing blurred images. Besides, the safety distance guarantees some robustness with respect to the GPS accuracy in order to avoid drift that may impact the system location and lead the UAVs to potential collisions.

In the *safety* scenario (see, Fig. 7), three multi-rotors fly in a formation while tracking a human working on a tower. The optimization problem (2) running onboard the vehicles guarantees the compliance with safety requirements (i.e., avoiding collisions with the power tower and with obstacles along the path) while it ensures that the mission is carried out successfully. To evaluate and demonstrate the applicability of the proposed software framework, we simulated high-level commands from the ground operator that, through the *task manager*, change the viewpoints of the formation by moving the vehicles around the tower. Note that the *task manager* precludes the UAVs from running out of battery by reacting on time, sending them for emergency landing and reassigning their tasks to others. Further details on the onboard navigation system and sensors are available in [7].

V. Conclusions

This paper has presented a multi-layer software architecture for encoding and supporting cooperative power line inspection operations with a fleet of UAVs. Cognitive capabilities have been considered for the safe and successful accomplishment of the assigned missions. The architecture is designed around a set of software components that handle the current states of the system, assign high-level tasks, and monitor the progress of the fully autonomous mission, while ensuring compliance with safety requirements. Simulations in Gazebo have demonstrated the feasibility and the effectiveness of the proposed framework, aiming towards the fulfillment of real-world tests. Future work will include the integration of more challenging cognitive capabilities, such as human interaction and gesture recognition to learn humans' intentions, and lead to field experiments. Furthermore, we plan to investigate on planning algorithms that can also deal with uncertainties in task execution and workers' intentions.

References

[1] J. Park *et al.*, "Method of operating a GIS-based autopilot drone to inspect ultrahigh voltage power lines and its field tests," *Journal of Field Robotics*, vol. 37, no. 3, pp. 345–361, 2019.

[2] H. Baik *et al.*, "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] G. Gallego *et al.*, "Event-based Vision: A Survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 1–26, 2020.

[5] V. Walter *et al.*, "UVDAR System for Visual Relative Localization With Application to Leader–Follower Formations of Multirotor UAVs," *IEEE Robotics and Automation Letters*, vol. 4, no. 3, pp. 2637–2644, 2019.

[6] R. Penicka *et al.*, "Physical Orienteering Problem for Unmanned Aerial Vehicle Data Collection Planning in Environments With Obstacles," *IEEE Robotics and Automation Letters*, vol. 4, no. 3, pp. 3005–3012, 2019.

[7] T. Baca *et al.*, "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*, 2021, To Appear.

[8] J. L. Sanchez-Lopez *et al.*, "AEROSTACK: An architecture and open-source software framework for aerial robotics," in *International Conference on Unmanned Aircraft Systems*, 2016, pp. 332–341.

[9] K. Xiao *et al.*, "XTDrone: A Customizable Multi-Rotor UAVs Simulation Platform," 2020. [Online]. Available: https://arxiv.org/pdf/2003.09700

[10] F. Furrer *et al.*, "RotorS — A modular gazebo MAV simulator framework," in *Robot Operating System*. Springer, 2016, pp. 595–625.

[11] D. Abeywardena *et al.*, "Design and development of ReCOPTER: An open source ROS-based multi-rotor platform for research," in *ACRA*, 2015.

[12] I. Mellado-Bataller *et al.*, "MAVwork: a framework for unified interfacing between micro aerial vehicles and visual controllers," in *Frontiers of Intelligent Autonomous Systems*. Springer, 2013, pp. 165–179.

[13] A. Ollero *et al.*, "The AEROARMS Project: Aerial Robots with Advanced Manipulation Capabilities for Inspection and Maintenance," *IEEE Robotics Automation Magazine*, vol. 25, no. 4, pp. 12–23, 2018.

[14] M. Jacquet *et al.*, "Motor and Perception Constrained NMPC for Torque-Controlled Generic Aerial Vehicles," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 518–525, 2021.

[15] M. Ryll *et al.*, "A Novel Overactuated Quadrotor Unmanned Aerial Vehicle: Modeling, Control, and Experimental Validation," *IEEE Transactions on Control Systems Technology*, vol. 23, no. 2, pp. 540–556, 2015.

[16] E. Cervera, "Try to Start It! The Challenge of Reusing Code in Robotics Research," *IEEE Robotics and Automation Letters*, vol. 4, no. 1, pp. 49–56, 2019.

[17] J. L. Sanchez-Lopez *et al.*, "A Reliable Open-Source System Architecture for the Fast Designing and Prototyping of Autonomous Multi-UAV Systems: Simulation and Experimentation," *Journal of Intelligence & Robotics Systems*, vol. 84, pp. 779–797, 2016.

[18] G. Silano *et al.*, "Power Line Inspection Tasks with Multi-Aerial Robot Systems via Signal Temporal Logic Specifications," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 4169–4176, 2021.

[19] M. W. Mueller *et al.*, "A Computationally Efficient Motion Primitive for Quadrocopter Trajectory Generation," *IEEE Transactions on Robotics*, vol. 31, no. 6, pp. 1294–1310, 2015.

[20] M. Saska *et al.*, "Formation control of unmanned micro aerial vehicles for straitened environments," *Autonomous Robots*, vol. 44, pp. 991–1008, 2020.

[21] V. Krátký *et al.*, "Autonomous Reflectance Transformation Imaging by a Team of Unmanned Aerial Vehicles," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 2302–2309, 2020.

[22] 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.