Distributed Fixed-Wing Drone Network for Firefighting Emergency Response

Daniele Turrini, *Student ID: 249485*
Giacomo Bianchi, *Student ID: 248734*

***Abstract*—Rapid and sustained response to wildfires is becoming increasingly critical as fire incidents rise annually. While ground-based fire trucks effectively suppress fires with road access, aerial assets are required elsewhere but suffer from slow turnaround and refill delays. This paper proposes a distributed control algorithm for a fleet of fixed-wing unmanned aerial vehicles (UAVs) that alternately refill at designated supply depots and discharge water over active fire zones. Each UAV, connected via a fully meshed communication network, continuously shares and updates fire-front positions. Task allocation is achieved through a dynamic Voronoi–Lloyd partitioning scheme, which redistributes UAV coverage in real time to adapt to moving fire fronts. Simulation results demonstrate that our approach scales with varying UAV counts and fire dynamics, delivering significant gains in response speed and discharge consistency over centralized methods. The proposed system enables persistent, adaptive aerial firefighting operations, reducing refill-induced downtime and enhancing overall wildfire suppression efficacy.**

I. Introduction

Wildfires have been increasing in frequency and intensity worldwide, posing severe risks to ecosystems, property, and human life. Traditional ground-based firefighting assets such as fire trucks and hand crews excel where road networks exist, but inaccessible terrain and rapidly spreading fire fronts often necessitate aerial intervention. The existing aerial firefighting systems are typically slow and require refill protocols that introduce critical delays. Indeed, since the drop of water is concentrated in small areas, the frequency of those drops becomes crucial. Today all the significant actions to extinguish the fires are done by humans, using water pumps if the truck can reach the site, or controlling air tankers if the environment doesn't allow ground measures. For this purpose fixed-wing and rotary UAVs offer promising agility and coverage to this problem.
Recently some drones have started to be applied in firefighting applications but mainly for the visualization and understanding of the fire position and extension; indeed, the fast deployment of a drone can lead to a better response plan, and also real-time monitoring. Our purpose is to create a distributed algorithm, able to control a swarm of fixed-wing UAVs to make them put out one or more fires in a certain defined area. The advantages of this application would be many:

- *Safety*: since the UAVs don't need any physical person inside, they can reach dangerous heights over the fire for a more precise drop without risk any human life;
- *Low inactive time between 2 drops*: The main advantage of this system is the possibility to have a fair number of drones working together. This reduce the waiting time between two consecutive drops.
- *Easy collaboration*: when more then one firefighter plane work together, the constant communications and collaboration could be difficult and require a lot of mental effort under stress.
- *Precision*: with all the data measured by sensors and the communications network over all the UAVs, each drone have a good estimate of the fire position that improve the precision of the drop.

A. Problem Formulation

This work addresses the problem of distributed control for a fleet of UAVs operating as aerial firefighting agents in a three-dimensional environment. The UAVs are fully connected via a communication network and are capable of autonomously executing periodic tasks regardless of the number of agents, fire sources, or refill places. The proposed system is designed to be resilient to UAV failures, allowing for real-time adaptation without interrupting mission continuity. A key challenge considered is the dynamic prioritization of fires. In this framework, fire severity is modeled by its spatial extension, which directly influences task allocation: larger fires are assigned a proportionally greater number of UAVs to ensure a faster and more effective response. The objective of this project is to integrate a robust distributed control strategy using a Voronoi–Lloyd partitioning scheme for spatial coverage, an Extended Kalman Filter (EKF) for optimal state estimation of fire fronts based on noisy sensor data, and a Linear Consensus algorithm for decentralized estimation and dissemination of fire locations and sizes. The combination of these techniques enables scalable, adaptive, and coordinated aerial firefighting missions in complex, evolving environments.

B. Paper Organization

The rest of the report is organized as follows. In Section II we have the description of which model have been used in our simulation and all the sensors considered. Then the solutions found and their descriptions are in Section III. In Section IV, it is resumed how the simulation works. Then Section V shows all the Simulation Results, and then Section VI concludes the report.

II. Adopted Models

A. Communication System

For the Voronoi partitioning algorithm to operate correctly, each UAV must obtain the estimated positions of all other agents. Accordingly, we initially assume a fully connected communication topology (Fig. 1), in which every UAV broadcasts information to all the others.

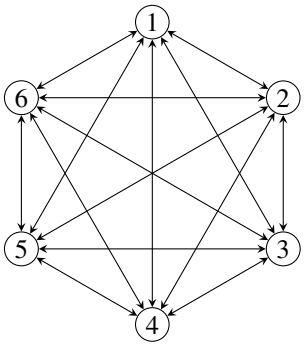


Figure 1. Example of fully connected graph with 6 nodes and bidirectional links.

However, maintaining full connectivity at all times is impractical due to packet loss and intermittent link failures. To evaluate the robustness of our distributed control scheme, we therefore introduce stochastic communication outages by randomly disabling one or more inter-UAV links. This allows us to quantify the impact of intermittent connectivity on overall system performance.

B. System Model

Ideally, accurate modeling of an aerial vehicle in three-dimensional space requires a state vector of at least 12 variables (or 13 when using quaternions for orientation). These typically include:

- 3 for the position $[x, y, z]$,
- 3 (or 4) for the orientation $[\phi, \theta, \psi]$,
- 3 for the linear velocity $[v_x, v_y, v_z]$,
- 3 for the angular velocity $[\omega_x, \omega_y, \omega_z]$.

However, such high-fidelity dynamic modeling introduces considerable complexity in both control design and real-time implementation, which is not necessary for the scope of this study. Therefore, a simplified dynamic model is adopted to balance realism with computational efficiency.

The proposed state vector is reduced to the following four parameters:

$$\mathbf{s} = \begin{bmatrix} x \\ y \\ z \\ \theta \end{bmatrix}. \tag{1}$$

This reduced model captures the UAV's 3D position and its yaw orientation (θ), while omitting linear and angular velocity states. Instead, velocities are treated as direct control inputs, defined as:

$$\mathbf{u} = \begin{bmatrix} v_{lin} \\ v_z \\ \omega_z \end{bmatrix}, \tag{2}$$

where $v_{lin} = \sqrt{\dot{x}^2 + \dot{y}^2}$ which cannot be uncoupled in this model. This formulation resembles an extended unicycle model, augmented to handle vertical motion independently.

The adopted UAV model follows a unicycle-like kinematic structure, which inherently exhibits nonlinear dynamics. The discrete-time equation of motion can be expressed as:

$$s_{k+1} = f(s_k, u_k, \nu_k), \tag{3}$$

where s_k, and s_{k+1} are respectively the state at the current and next instant. u_k is the control applied at the current instant and ν_k is an additive Gaussian noise that embeds both the uncertainty on the model and the control, $\nu \sim N(0, Q)$ with Q the covariance matrix obtained with a standard deviation of 1 [m]. Knowing this a well-defined version system's dynamics can be written as:

$$\begin{bmatrix} x_{k+1} \\ y_{k+1} \\ z_{k+1} \\ \theta_{k+1} \end{bmatrix} = \begin{bmatrix} x_k \\ y_k \\ z_k \\ \theta_k \end{bmatrix} + \begin{bmatrix} cos\theta_k & 0 & 0 \\ sin\theta_k & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \Delta t \left(\begin{bmatrix} v_{lin} \\ v_z \\ \omega_z \end{bmatrix} + \nu_k \right). \tag{4}$$

The Δt is the time step and is a simulation parameter and $\nu_k = (\nu_1, \nu_2, \nu_3)^T$ if we want to consider how the noise affects each control.

Given that the UAVs in question are fixed-wing aircraft, it is important to account for the aerodynamic constraints inherent to this class of vehicles. In particular, fixed-wing UAVs require a minimum forward (linear) velocity to maintain lift and sustain flight. As such, a lower bound is imposed on the linear velocity v_{lin} to ensure realistic operation and to avoid stall conditions. So ensure the physical feasibility of the model for fixed-wing UAVs, some constraints are imposed on the control inputs:

- $v_{lin} \in [50, 100]$ [m/s]. We imposed an upper limit of 100 [m/s], to compute the lower limit we used the *stall-speed equation*, an aerodynamic formula defined as:

$$V_s = \sqrt{\frac{2W}{\rho S C_{L,max}}}, \tag{5}$$

  where W is the aircraft weight [N] , ρ is the air density ($\approx$1.225 kg/m³ at sea level), S is the wing planform area (m²), $C_{L,max}$ is the maximum lift coefficient (typically 1.2–1.5 for small-medium wings with flaps). Assuming an approximate total mass of 3000 kg, where 1000 kg corresponds to the unloaded UAV and 2000 kg represents the maximum water payload, and a wing planform around 16 m^2, the resulting V_s become 44.7 [m/s];
- $v_z \in [-100, 100]$ [m/s];
- $w_z \in [-10, 10]$ [rad/s].

C. Sensors

Each UAV is equipped with three primary sensors to provide the state estimates required by the state estimation algorithm:

- **Global Positioning System (GPS):** Provides absolute position measurements $p = [x, y]^T$ with a Gaussian noise:

$$\mathrm{p}_{\text{meas}} = \mathrm{p} + \epsilon_{\text{GPS}}, \quad \epsilon_{\text{GPS}} \sim \mathcal{N}(0,\, \sigma^2_{\text{GPS}} I_), \quad \sigma_{\text{GPS}} = 3\,\text{m}.$$

- **Ultrasonic Altimeter:** Measures altitude above ground level in the range $0 \le z \le 40$ m, with a Gaussian noise:

$$z_{\text{meas}} = z + \epsilon_{\text{ult}}, \quad \epsilon_{\text{ult}} \sim \mathcal{N}(0,\, \sigma^2_{\text{ult}}), \quad \sigma_{\text{ult}} = 1.5\,\text{m}.$$

  (Note: more sophisticated payload-refill sensing is beyond the scope of this work.)
- **Gyroscope:** Measures yaw rate ω_z with additive Gaussian noise:

$$\omega_{\text{meas}} = \omega + \epsilon_{\text{gyro}}, \quad \epsilon_{\text{gyr}} \sim \mathcal{N}(0,\, \sigma^2_{\text{gyr}}), \quad \sigma_{\text{gyr}} = 0.3\,\text{rad/s}.$$

To accommodate sensor characteristics and processing constraints, each measurement channel operates at a distinct sampling frequency:

$$\begin{aligned} f_{\text{GPS}} &= 10 \text{ Hz}, \\ f_{\text{ultrasonic}} &= 25 \text{ Hz}, \\ f_{\text{gyro}} &= 100 \text{ Hz}. \end{aligned}$$

These heterogeneous update rates are handled within the estimation framework by timestamped measurements and Kalman filter updates, ensuring that each UAV maintains an accurate and timely estimate of its own state for subsequent Voronoi–Lloyd partitioning and consensus-based communication.

To enable each UAV to detect and quantify active fires, every vehicle is equipped with:

- **High-resolution RGB camera** for visual flame and smoke detection, and
- **Thermal infrared (IR) sensor** for radiometric fire-front mapping.

Both sensors are operated at a minimum standoff distance of 70 $[m]$ which is the sensors' optimal focus and dynamic range. Visual and thermal data are processed on board to extract, for each detected fire its position and extension. Those local fire estimates are then exchanged across the fleet via the fully connected (or intermittently connected) communication network. Then a linear consensus protocol fuses the individual measurements into a global estimate of fire positions and extents. This consensus-based fusion ensures that all UAVs converge to a consistent situational picture, which is then used to drive the Voronoi–Lloyd partitioning and subsequent coordinated firefighting maneuvers.

III. Solution

In this section we will analyze in details the three main points of the proposed approach. Those points try to solve the main problems encountered in this project:

- *Distributed control algorithm*,
- *UAV State estimation*,
- *Distributed fire estimation*.

A. Voronoi–Lloyd-Based UAV Partitioning and Control

The coordination logic for the UAV fleet is driven by a weighted Voronoi–Lloyd control algorithm, which dynamically partitions the area among UAVs to ensure efficient task distribution and adaptive response to environmental conditions. The algorithm serves two main purposes: (i) to assign spatial regions of responsibility to each UAV based on proximity and mission-specific priorities, and (ii) to steer each UAV toward the weighted centroid of its assigned region through a proportional control mechanism.

The algorithm begins by discretizing the mission space into a two-dimensional grid with a specified resolution. Let $\mathcal{P} \subset \mathbb{R}^2$ denote the space domain, and let each point $q \in \mathcal{P}$ represent a grid cell in the x, y plane. Assuming to have n UAVs, each one with position $p_i = [x_i, y_i]^T \in \mathbb{R}^2$, for $i = 1, \ldots, n$. For each grid point $q \in \mathcal{P}$, the Euclidean distance to all UAV positions is computed, and the grid cell is assigned to the nearest UAV. This process results in a discrete Voronoi tessellation, where each UAV is responsible for a unique partition of the mission space.

The Voronoi cell associated with the i-th UAV is defined as:

$$\mathcal{V}_i = \{q \in \mathcal{P} \mid \|q - p_i\| \le \|q - p_j\|, \quad \forall j \ne i\}, \tag{6}$$

where $\|\cdot\|$ denotes the standard Euclidean norm. Each cell $\mathcal{V}_i$ thus contains all the points in $\mathcal{P}$ which are closest to UAV i as to any others.

To enforce collision avoidance between the UAVs, we approximate each vehicle by a circle of radius δ_i. For any two agents i and j, define the combined safety distance:

$$\Delta_{ij} = \delta_j + \delta_i$$

Let $p_i, p_j \in \mathcal{P}$ denote their planar positions. We then modify the usual Voronoi cell for agent i as follows:

$$\tilde{\mathcal{V}}_i = \begin{cases} \{q \in \mathcal{P} \mid \|q - p_i\| \le \|q - p_j\|\}, & \text{if } \Delta_{ij} \le \frac{\|p_i - p_j\|}{2} \\ \{q \in \mathcal{P} \mid \|q - p_i\| \le \|q - \tilde{p}_j\|\}, & \text{otherwise} \end{cases}, \tag{7}$$

where the *virtual position* $\tilde{p}_j$ is obtained by shifting p_j toward p_i just enough to enforce the safety band. It is computed as follows:

$$\tilde{p}_j = p_j + 2\left(\Delta_{ij} - \frac{\|p_i - p_j\|}{2}\right)\frac{p_i - p_j}{\|p_i - p_j\|}.$$

This construction pushes the Voronoi bisector away from p_i by exactly Δ_{ij}, ensuring that the weighted centroid of $\tilde{\mathcal{V}}_i$ always lies outside the forbidden overlap region and thus prevents collisions when each UAV moves toward its centroid, as Fig. 1 shows.

In this project, it is possible to use arbitrarily the formulation of the Voronoi tessellation with (Eq. 7) or without the safety factor (Eq. 6).

To evaluate the importance of each cell within a region, a task-specific density function is applied defined as $\phi(q, \sigma, t)$ (knowing that q represent a vector of 2D coordinates $[x, y]^T$). This function serves as a weight map tailored to different mission objectives. For fire suppression, the density reflects

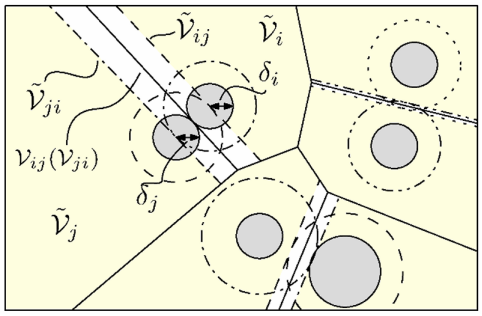


Fig. 1: Example of modified Voronoi tessellation with collision avoidance extension.

the estimated fire intensity. For water refilling, it indicates the proximity to the nearest water source. During landing operations, it supports spatial balancing to ensure a safe and stable landing process. Figure 2 illustrates density distributions for fires, water objectives and landing position, where Gaussian functions are used with variable standard deviations to control the spatial extent of influence; so the standard deviation of the Gaussian controls the extent while the mean controls the position in the environment x, y plane.

Once the density map is applied, the algorithm calculates the mass of each UAV's Voronoi cell by summing the weights of the grid points within the cell as described in the following equation:

$$M_{\mathcal{V}_i} = \int_{\mathcal{V}_i} \phi(q, \sigma, t)\, dq. \tag{8}$$

The weighted centroid, representing the center of mass of the region, is then computed as follows

$$C_{\mathcal{V}_i} = \frac{1}{M_{\mathcal{V}_i}} \int_{\mathcal{V}_i} q\, \phi(q, \sigma, t)\, dq. \tag{9}$$

A velocity command is generated to guide each UAV toward this centroid. In the horizontal X-Y plane, the UAV applies a proportional control law:

$$v_{lin} = K_p \cdot (p_i - C_{\mathcal{V}_i}), \quad \omega_z = K_a \cdot \theta_{error}, \tag{10}$$

where v_{lin} is the linear speed vector, ω_z is the yaw-rate command, K_p and K_a are positive control gains (defined in our simulation respectively as 50 and 10), $\mathbf{p}_i$ denotes the current position of the i-th UAV, and $\mathbf{C}_{\mathcal{V}_i}$ its corresponding centroid. The scalar θ_{error} quantifies the angular deviation between the vehicle's heading and the direction toward the centroid. It is computed as

$$\theta_{error} = \operatorname{atan2}\left(C_{\mathcal{V}_i,y} - s_{i,y},\ C_{\mathcal{V}_i,x} - s_{i,x}\right) - s_{i,\theta}, \tag{11}$$

where s_i is the state vector of the i-th UAV.

Vertically, the UAV adjusts its altitude using a proportional controller to align with a predefined flight surface, adapting to terrain changes and maintaining a mission-appropriate flight height. Indeed the vertical control is defined as:

$$v_z = K_z \cdot (E(s_x, s_y) - s_z + h), \tag{12}$$

where K_z is the proportional gain (defined as 100 to prevent ground colliding), s_x, s_y, s_z are respectively the coordinates of the UAV, $E(s_x, s_y)$ return the hight of the environment ground and h is the flight height that we want to keep from the ground.

In the multi-UAV framework, each vehicle executes one of three mission roles, which differ only in the choice of density function $\phi_i(x, y, \sigma, t)$ used within a Voronoi-based coverage controller. For each objective the density function implement gaussian function in correspondence to the target, each of this has its characteristics that are constantly updated. By redefining this spatial weighting function, the same control law automatically adapts to varying mission objectives:

- **Fire Suppression:** The UAV prioritizes regions with higher fire intensity by using a fire-specific density map. In this case, the density function for the i-th UAV is given by $\phi_i(x, y, \sigma, t) = D_i(x, y, \sigma, t)$, where

$$D_i(x, y, \sigma, t) = e^{-\left(\frac{(x - x_{\text{fire1}}(t))^2}{2\sigma_{\text{fire1}}^2} + \frac{(y - y_{\text{fire1}}(t))^2}{2\sigma_{\text{fire1}}^2}\right)} + e^{-\left(\frac{(x - x_{\text{fire2}}(t))^2}{2\sigma_{\text{fire2}}^2} + \frac{(y - y_{\text{fire2}}(t))^2}{2\sigma_{\text{fire2}}^2}\right)}, \tag{13}$$

  and $\sigma = [\sigma_{\text{fire1}}, \sigma_{\text{fire2}}]$ defines the spatial influence of each fire source.
- **Water Refill Coordination:** The UAV is attracted toward areas with water sources to enable refilling operations. The density function becomes $\phi_i(x, y, \sigma, t) = W_i(x, y, \sigma)$, defined as

$$W_i(x, y, \sigma) = e^{-\left(\frac{(x - x_{\text{water}})^2}{2\sigma_{\text{water}}^2} + \frac{(y - y_{\text{water}})^2}{2\sigma_{\text{water}}^2}\right)}, \tag{14}$$

  where $\sigma = \sigma_{\text{water}}$ (set constant equal to 40) determines the spatial extent of the water zone's influence.
- **Landing:** Once the fires are extinguished, UAVs return to their initial takeoff positions. The density function in this phase is $\phi_i(x, y, \sigma, t) = L_i(x, y)$, where

$$L_i(x, y) = e^{-\left(\frac{(x - x_{\text{start}_i})^2}{2\sigma_{\text{landing}}^2} + \frac{(y - y_{\text{start}_i})^2}{2\sigma_{\text{landing}}^2}\right)}, \tag{15}$$

  with x_{start_i} and y_{start_i} as the initial s_x and s_y coordinates of the i-th UAV before takeoff and $\sigma = \sigma_{\text{landing}}$ (set constant equal to 10).

By simply updating $\phi_i(\cdot)$, each UAV's motion planner, based on Voronoi tessellation, automatically switches between suppression, refilling, and landing behaviors without any modification to the underlying control architecture. This abstraction promotes modularity, simplifies role switching, and facilitates formal stability and convergence analysis under a unified mathematical framework.

As previously mentioned, all UAVs are capable of communicating with one another. In this way, each UAV is aware of the estimated positions of all others and can independently implement the Lloyd-Voronoi algorithm to compute its control actions.

B. Extended Kalman Filter for UAV State Estimation

The whole control algorithm assume to know every UAV position, so a precise estimate of the UAV state is essential for a proper working. For the state estimation of each UAV

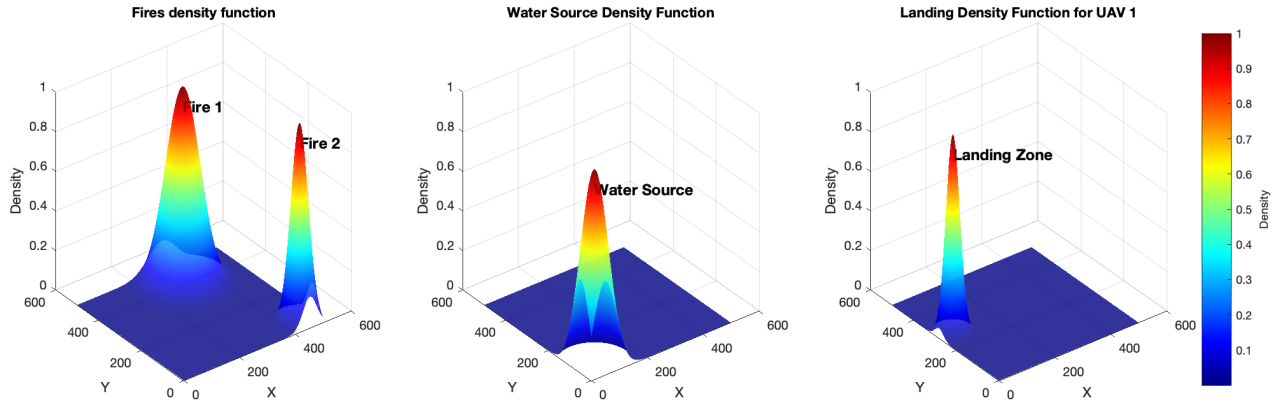


Fig. 2: Density functions: fire intensity distribution (left), water source proximity (center) and density function for the landing phase (right). Each distribution is modeled using Gaussian kernels, whose standard deviation controls the area of influence. These weights directly affect the centroid computation and UAV motion.

considering noisy and intermittent measurements, an Extended Kalman Filter (EKF) is implemented. The EKF is well-suited for nonlinear dynamic systems and allows for sequential fusion of heterogeneous sensor data such as GPS, ultrasonic altimeters, and gyroscopes.

The EKF operates in two phases: prediction and update.

a) Prediction Step: Given the current state estimate and control input, the EKF predicts the next state using the unicycle nonlinear dynamic model:

$$\hat{s}_{k+1}^{-} = f(\hat{s}_k, u_k), \tag{16}$$

where f is the state transition function, $\hat{s}_k$ is the estimation of the state at the time k, and u_k is the control input computed by the Voronoi algorithm. The corresponding state covariance matrix is propagated using the linearized model:

$$P_{k+1}^{-} = A_k P_k A_k^T + G_k Q G_k^T. \tag{17}$$

Here, A_k and G_k are the Jacobian matrices of the dynamics and process noise respectively, defined as:

$$A_k = \left.\frac{\partial f(s,u,\nu)}{\partial s}\right|_{\substack{s=\hat{s}_k \\ u=u_k \\ \nu=0}},$$

$$G_k = \left.\frac{\partial f(s,u,\nu)}{\partial \nu}\right|_{\substack{s=\hat{s}_k \\ u=u_k \\ \nu=0}},$$

Q instead is the process noise covariance matrix, that considers the uncertainties both in the model and the control.

The matrix A_k and G_k in this case project are considered as:

$$A_k = \begin{bmatrix} 1 & 0 & 0 & -u_1 \sin(\theta)\Delta t \\ 0 & 1 & 0 & u_1 \cos(\theta)\Delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix},$$

$$G_k = \begin{bmatrix} \cos(\theta)\Delta t & 0 & 0 \\ \sin(\theta)\Delta t & 0 & 0 \\ 0 & \Delta t & 0 \\ 0 & 0 & \Delta t \end{bmatrix}.$$

The process noise covariance matrix Q models the uncertainty in linear and angular velocities:

$$Q = \begin{bmatrix} \sigma_{u_x}^2 & 0 & 0 \\ 0 & \sigma_{u_y}^2 & 0 \\ 0 & 0 & \sigma_{\omega}^2 \end{bmatrix},$$

where σ_{u_x}, σ_{u_y}, and σ_{ω} are the standard deviations of the linear velocities in the x-y plane and the angular velocity, respectively, all of which are defined to be equal to 1.

b) Update Step: The Extended Kalman Filter (EKF) integrates sensor measurements operating at different sampling frequencies to more accurately reflect real-world conditions in the simulation. To further enhance realism, the simulation incorporates communication imperfections by assigning each sensor a probability of successfully delivering a valid measurement at any given time step. These probabilities are:

- GPS: $p_{\text{GPS}} = 0.9$,
- Ultrasonic altimeter: $p_{\text{ultrasonic}} = 0.8$,
- Gyroscope: $p_{\text{gyroscope}} = 0.7$.

This probabilistic model captures intermittent measurement dropouts, enabling a more realistic emulation of unreliable communication links and sensor availability.

At each update step, the algorithm checks whether a measurement from a given sensor is available (based on its update frequency and probability). If so, the resulting measurement can be written as:

$$m_k = h(s_k, \theta_{k-1}, \Delta t, \epsilon_k), \tag{18}$$

where we considered the measurement function h as nonlinear since our gyroscope doesn't measure directly the yaw angle but the angular velocity. So we use the previous angle stored θ_{k-1} and the time step Δt to compute the angular rate. It is defined as :

$$m_k = \begin{bmatrix} x_k + \epsilon_{GPS} \\ y_k + \epsilon_{GPS} \\ z_k + \epsilon_{ultrasonic} \\ \frac{\theta_k - \theta_{k-1}}{\Delta t} + \epsilon_{gyro} \end{bmatrix}. \tag{19}$$

Then the observation matrix H is the Jacobian of the measurement function and is defined as:

$$H = \left.\frac{\partial h(s, \theta_{k-1}, \Delta t, \epsilon)}{\partial s}\right|_{\substack{s=\hat{s}_k \\ \epsilon=0}} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & \frac{1}{\Delta t} \end{bmatrix}.$$

Since the measurement can occur at different time instant, we have to take a sub-matrices of H according to which measurement has been done:

- $H_{GPS} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$,
- $H_{ultrasonic} = 1$,
- $H_{gyro} = \frac{1}{\Delta t}$.

Also the uncertainty vector has to be modified accordingly with the measurement knowing that it is defined as:

$$\epsilon = \begin{bmatrix} \epsilon_{GPS} \\ \epsilon_{GPS} \\ \epsilon_{ultrasonic} \\ \epsilon_{gyr} \end{bmatrix}. \tag{20}$$

The next step is to compute the innovation y_{k+1} defined as:

$$y_{k+1} = m_{k+1} - \hat{m}_{k+1} = m_{k+1} - h_{k+1}(\hat{s}_{k+1}, \theta_k, \Delta t, 0). \tag{21}$$

Then we compute the covariance matrix of the innovation:

$$S_{k+1} = H_{k+1} P_{k+1}^{-} H_{k+1}^{T} + R, \tag{22}$$

where R is defined as

$$R = \begin{bmatrix} \sigma^2_{\text{GPS}} & 0 & 0 & 0 \\ 0 & \sigma^2_{\text{GPS}} & 0 & 0 \\ 0 & 0 & \sigma^2_{\text{ult}} & 0 \\ 0 & 0 & 0 & \sigma^2_{\text{gyr}} \end{bmatrix},$$

Then the covariance matrix predicted P_{k+1}^{-} and the covariance matrix of the innovation S_{k+1} are used to compute the Kalman gain (K_{k+1}) formulation:

$$K_{k+1} = P_{k+1}^{-} H_{k+1}^{T} (S_{k+1})^{-1}. \tag{23}$$

The Kalman gain is used to compute the estimation of the state update by integrating the measurement m_{k+1} of the state as follows:

$$\hat{s}_{k+1} = \hat{s}_{k+1}^{-} + K_{k+1} y_{k+1} \tag{24}$$

and finally the covariance update is computed:

$$P_{k+1} = (I - K_{k+1} H_{k+1}) P_{k+1}^{-}. \tag{25}$$

This modular update structure allows each UAV to incorporate only the measurements currently available, enhancing robustness in dynamic environments with intermittent or delayed sensor data.

C. Consensus Algorithm

The consensus algorithm employed in this project is designed to intelligently fuse measurements of fire location and extent, assigning greater weight to more recent observations. Indeed this algorithm allow us to aggregate local estimations from each UAV into a shared global estimate.

To perform this consensus algorithm we used:

- **Consensus transition Matrix** (Q_c): Each UAV maintains a weight matrix that reflects the importance of information received from other UAVs. This matrix is dynamically updated every time is performed a new measurement .
- **Last Measurements Time**: Each UAV tracks the time elapsed since the last measurement taken by itself and by other UAVs. This value is used to compute the weights in the consensus matrix.

The consensus algorithm is executed iteratively during the simulation. The main steps are as follows:

1) Local Measurements: Each UAV measures the fire's position and spread if it is within sensor range and is currently tasked with heading toward the fire. These measurements include uncertainty (random noise) to simulate real-world conditions. After a measurement, the UAV reset the Last Measurement time to 1 and send the measured quantity to all the others UAVs. In this way all the other drones know both the measured quantity and time when it arrives.

2) Weight Calculation: Each UAV computes the weights for the consensus matrix Q_c based on the Last Measurement values. To make all the estimates converge to same values we have to ensure that Q_c is at least a stochastic matrix. To do that every time there is a new measurement the Q_c matrix is update as illustrated in the following example.

Consider three UAVs, each of which maintains a vector of the timestamps at which it last received each fire measurement:

$$Lm_1 = \begin{bmatrix} 1 \\ 4 \\ 8 \end{bmatrix} \quad Lm_2 = \begin{bmatrix} 2 \\ 5 \\ 8 \end{bmatrix} \quad Lm_3 = \begin{bmatrix} 1 \\ 3 \\ 9 \end{bmatrix}. \tag{26}$$

Owing to communication delays, these timestamp vectors may differ slightly between vehicles. We first compute the element-wise inverse of each vector to obtain:

$$Lm_1^{-1} = \begin{bmatrix} 1 \\ \frac{1}{4} \\ \frac{1}{8} \end{bmatrix} \quad Lm_2^{-1} = \begin{bmatrix} \frac{1}{2} \\ \frac{1}{5} \\ \frac{1}{8} \end{bmatrix} \quad Lm_3^{-1} = \begin{bmatrix} 1 \\ \frac{1}{3} \\ \frac{1}{9} \end{bmatrix} \tag{27}$$

and then sum the entries of each inverse vector:

$$\Sigma Lm_1 = 1.375 \quad \Sigma Lm_2 = 0.825 \quad \Sigma Lm_3 = 1.441\,. \tag{28}$$

Finally, we define the consensus weighting matrix $Q_c \in \mathbb{R}^{3\times 3}$ by normalizing each row of the inverse-timestamp arrays:

$$Q_c = \begin{bmatrix} \frac{1}{1.375} & \frac{\frac{1}{4}}{1.375} & \frac{\frac{1}{8}}{1.375} \\ \frac{\frac{1}{2}}{0.825} & \frac{\frac{1}{5}}{0.825} & \frac{\frac{1}{8}}{0.825} \\ \frac{1}{1.441} & \frac{\frac{1}{3}}{1.441} & \frac{\frac{1}{9}}{1.441} \end{bmatrix}. \tag{29}$$

By construction, Q_c is row-stochastic and assigns larger weights to more recent measurements, thus biasing the consensus update toward the freshest data.

3) Estimate Updates: Local estimates of fire position and spread are updated using the consensus matrix in the following way, considering ξ as a generic estimated quantity:

$$\xi_{\text{k+1}} = Q_c \cdot \xi_{\text{k}}.$$

This represents a weighted average of the UAVs' estimates, where the weights are derived from Q_c, weighed in relation to how many instants have passed since the last measurement was made.

4) Extinction Thresholds: If the estimated extension of a fire drops below a predefined threshold, it is set to zero, indicating that the fire has been extinguished.

5) Integration with Control: The consensus algorithm is closely integrated with UAV motion control; indeed, the updated estimates from the consensus process are used to compute the Voronoi algorithm.

IV. Implementation Details

A. Initialization and Environment Definition

Since the whole control system is distributed, the number of UAVs operating in the scenario can be arbitrarily chosen and spatially arranged according to specific needs. In this particular case, a swarm of 5 UAVs was considered.

Initially, all UAVs are positioned with an x coordinate of 50 meters and a z coordinate (altitude) set at ground level. The heading angle θ is initialized to $-\pi/2$ radians. To prevent overlap along the y-axis, UAVs are spaced 30 meters apart, starting from a base y coordinate of 250 meters.

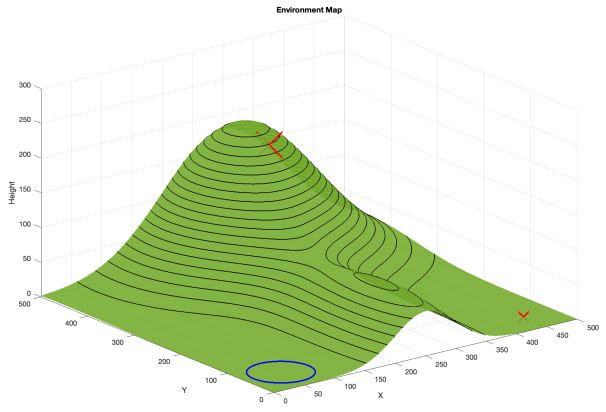


Fig. 3: Simulation environment: the blue circle denotes the lake (water refill), and the red $\times$ symbols denote the two moving fires.

The simulation framework is capable of adapting to diverse geographic scenarios by tailoring control strategies to the environment's topology. An example scenario is depicted in Fig. 3, where:

- A lake centered at $(50, 50, 0)\,\text{m}$ with a diameter of $90\,\text{m}$ serves as the UAVs' water replenishment site.
- Two moving fires are initially located at
  - Fire 1: $(300, 400, 180)\,\text{m}$ with an initial radius (σ_{fire1}) of $50\,\text{m}$,
  - Fire 2: $(450, 50, 0)\,\text{m}$ with an initial radius (σ_{fire2}) of $20\,\text{m}$.

Each fire's center evolves linearly over the simulation horizon $t = 1, \ldots, T_{\text{sim}}$ according to

$$\begin{cases} x_{F1}(t) = x_{F1,\text{start}} - 5\,(t-1) \\ y_{F1}(t) = y_{F1,\text{start}} - 2\,(t-1), \end{cases}$$

$$\begin{cases} x_{F2}(t) = x_{F2,\text{start}} - 2.5\,(t-1) \\ y_{F2}(t) = y_{F2,\text{start}} + 1\,(t-1). \end{cases}$$

.

B. Communication loss

To realistically simulate UAVs communication, a verification mechanism is implemented where each drone not only broadcasts its estimated state but also sends a communication `check` flag. This mechanism allows each UAV to determine whether it has successfully received data from its peers at each time step.

Communication reliability is modeled probabilistically, with a predefined success probability, we consider a probability of incorrect communication between drones of 0.05. At every iteration, each UAV evaluates whether it has received new data from the others. If communication from a specific UAV fails, either due to random packet loss or because the UAV has crashed, the corresponding `check` flag is set to zero.

When communication from a UAV is lost, the system does not discard the drone's contribution. Instead, it reuses the last known state estimate of that UAV, along with its previously received fire position and intensity estimates. This fallback mechanism ensures continuity and robustness of the overall coordination strategy, even under partial communication failures or drone malfunctions.

C. UAV Crash Handling

To account for possible UAV malfunctions during mission execution, the system includes a mechanism to detect and handle UAV crashes. In this context, a UAV crash is defined as a permanent loss of communication, typically due to hardware failure or electrical system shutdown, which causes the UAV to fall and it becomes non-operational.

However, to avoid false positives caused by transient communication issues or packet loss, the system does not immediately classify a UAV as crashed upon a single communication failure. Instead, it monitors each UAV how many consecutive time steps a UAV has failed to transmit valid data. If this count exceeds a predefined threshold of 10, the system flags the UAV as crashed.

Once a crash is detected, the failed UAV is effectively removed from all relevant control, estimation, and coordination variables. This includes:

- Reducing the total number of active UAVs.
- Deleting the crashed UAV's state vectors and sensor measurements.

- Updating the consensus matrices used for shared estimations.

This delayed crash-detection logic ensures that the system remains resilient to temporary network outages while still accurately modeling UAV losses when genuine failures occur. Overall, this approach contributes to a fault-tolerant coordination framework that can sustain mission continuity despite partial fleet degradation.

D. Takeoff, Refill, and Landing Procedures

To simulate a realistic UAV firefighting mission, the system models not only the firefighting phase but also the complete mission life-cycle, including takeoff, refill operations, and landing.

a) Takeoff Management: The mission begins with a sequential takeoff procedure. UAVs are initially kept stationary on the ground and are allowed to take off one by one according to a predefined frequency, ensuring a controlled and staggered ascent. This is governed by a counter and a frequency parameter that introduces a delay between successive takeoffs, mimicking real-world operations where UAVs may require safe distances during launch.

b) Refill Phase: When a UAV reaches a water source (typically a lake), a refill phase is triggered. During this phase, the UAV's motion is constrained: it flies at minimum linear velocity and maintains a constant altitude. This restriction simulates the time and stability required for refilling the water tank while hovering over the lake.

c) Dynamic Objective Assignment: After completing the refill phase, UAVs reassess their mission objectives based on the current fire conditions. If both fire extension estimates (`sigma_est_fire1` and `sigma_est_fire2`) are equal to zero—indicating that all fires have been extinguished—the UAVs update their objective to `objective 3`, which corresponds to returning to their initial positions for landing. Otherwise, they resume fire suppression activities, setting their mission to `objective 1` (active firefighting). The `objective 2` state corresponds to the UAVs heading to the water source for refilling.

d) Landing Procedure: When all UAVs agree that both fires have been extinguished, and their individual objectives are set to `3`, they begin returning to their original takeoff positions. As each UAV approaches its initial coordinates, the system initiates a controlled landing sequence. If the UAV is within a certain distance of its starting point, it begins to descend by adjusting its vertical control input. Once the UAV is in its initial position, all its motion commands are nullified, effectively simulating a safe landing on the ground.

V. Simulation Results

This section presents the data and analysis obtained through simulations.

A. Localization with EKF

Fig. 4 shows the state estimation errors obtained with the Extended Kalman Filter, using the measurement update frequencies and standard deviations described in Section II-C.

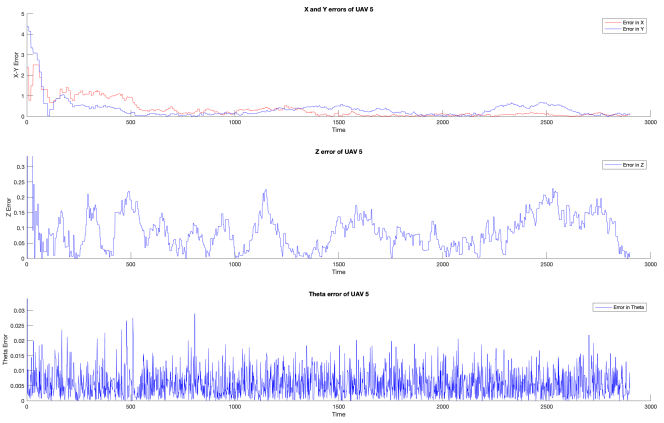


Fig. 4: EKF state estimation error. The first plot shows both the X and Y estimation error, the second shows the Z estimation error and the third shows the θ one

a) X-Y coordinates: The top plot displays the errors in the x and y coordinates: here, the errors are of similar magnitude for both axes. However, an unwanted phenomenon can arise: over certain time intervals, the error in one coordinate can become noticeably larger than in the other. Specifically, this si due to the process-noise propagation matrix G_k, the terms $\cos(\theta)$ and $\sin(\theta)$ distribute the forward-velocity variance between the x and y axes. If the UAV travels predominantly along the horizontal axis (i.e., $\theta \approx 0$), most of the forward-velocity noise projects onto the x coordinate, with very little affecting the y coordinate, and vice versa for vertical movements. Over long simulations with balanced horizontal and vertical maneuvers, this anisotropic effect tends to average out and become negligible. Anyway for all the UAVs after one third of the simulation, the estimation error is bounded between $1\ [m]$. Considering that we are using a measurement rate for the GPS of $10\ [Hz]$, and its standard deviation is $\sigma_{GPS} = 3[m]$, a reduction of one third of the estimation uncertainty is a good result. Fig. 5 compares the true flight path of a representative UAV with its onboard estimate and the raw GPS readings over the same time interval.

b) Z coordinate: The middle plot in Fig. 4 illustrates the estimation error for the z coordinate. Rather than converging to zero, the error remains bounded within approximately ± 0.25,m. This residual uncertainty is roughly one-sixth of the ultrasonic sensor's measurement standard deviation, $\sigma_{\text{ultrasonic}} = 1.5\ [m]$. The tighter bound on the z-error is largely due to the higher sampling rate of the ultrasonic sensor compared to the GPS.

c) θ Orientation: The last plot of Fig. 4 shows the estimation error of the yaw angle θ. Although the error trace exhibits high-frequency noise, since it is derived from angular velocity measurements, it remains tightly bounded within approximately $\pm 0.03\ [rad]$. This accuracy is achieved through the gyroscope's high sampling rate ($100\ [Hz]$) and its measurement noise standard deviation ($\sigma_{\text{gyro}} = 0.3$, rad/s). In this project, we did not model gyroscopic drift, assuming it is fully compensated by our filtering.

Fig. 6 illustrates the behavior of the covariance matrix trace

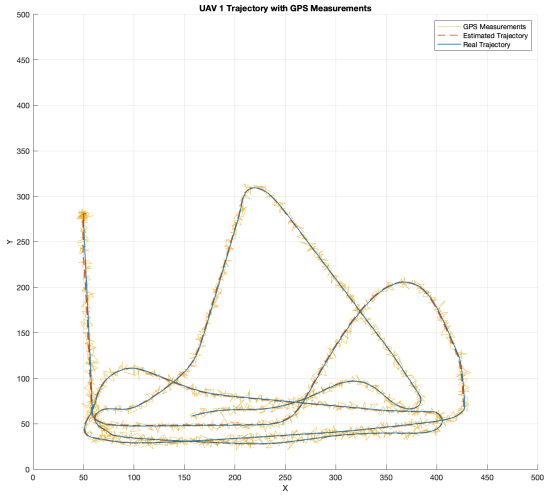


Fig. 5: Real, estimated , and GPS-measured trajectories of a generic UAV during its mission.

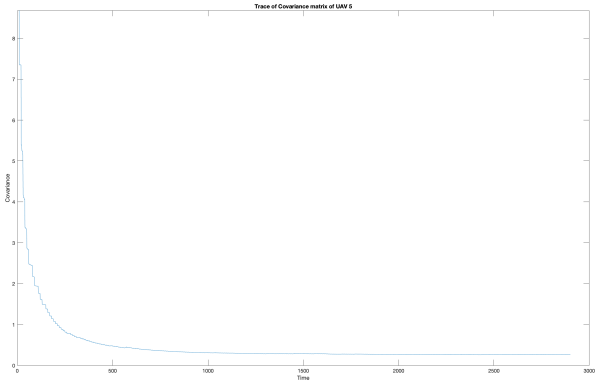


Fig. 6: Behavior of the Covariance matrix's Trace $Tr(P)$.

(Tr(P)) for the same UAV that executed which estimation errors are analyzed in Fig. 4. The trace exhibits a similar pattern across different UAVs following an exponential trend, as depicted in the first plot. A closer inspection, shown in Fig. 7, reveals periodic increases corresponding to the prediction steps, followed by sharp decreases due to the update steps of the estimation process. The reduction in Tr(P) during the update step depends on the specific measurements obtained at that instant, and therefore can vary accordingly.

B. Consensus Algorithm

The convergence of the state estimates under the consensus algorithm is illustrated in Fig. 8, which shows the estimation process for fire 1 during the simulation. The concept is the same for the second fire, shown in Fig. 9.

In this simulation, the true fire moves linearly over time to emulate a moving blaze. A measurement is taken only when a UAV comes within the sensor range (IR and camera) of the fire, which in this simulation is set to 70 $[m]$. Each new measurement is then broadcast to al other drones; the resulting updates to the consensus weight matrix Q_c ensure that the most recent observation carries greater influence in the fused

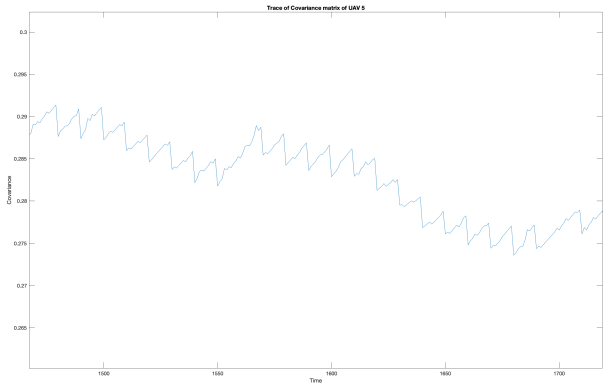


Fig. 7: Zoom of the Covariance matrix's Trace $Tr(P)$ behavior. This figure shows the different prediction steps, when $Tr(P)$ increase, and the Update ones, when $Tr(P)$ decrease.

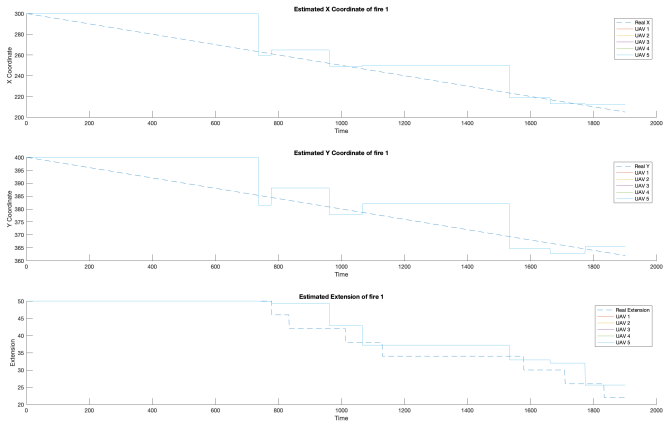


Fig. 8: Evolution of the estimated position and size of Fire 1.

estimate. For clarity, we model the effect of water drops on each fire as a reduction in its area by a fixed factor of four (see Fig. 8 and Fig 9), although in reality, the relationship between water dro and fire shrinkage is more complex.

In Fig. 8 the individual estimate trajectories overlap so quickly that their distinctions are hard to discern; they converge within one or two iterations (see the zoom in Fig. 10). This behavior arises because the long interval between measurements causes the latest reading to dominate the consensus weights, minimizing the influence of older data.

C. Voronoi Computation

Three figures are presented, illustrating the three main phases of the simulation. In the figures, the estimated position of each UAV (green triangle) and the actual position (black triangle) are shown. Additionally, the Voronoi tessellation is displayed, where each area corresponds to a specific UAV. The centroids of the Voronoi cells are also indicated using green crosses. The positions of the water source (blue circumference), as well as the positions and spread of the fires, are represented. Red crosses indicate the actual fire positions, while purple crosses indicate the estimated fire positions. The size of each cross corresponds to the actual or estimated spread (intensity) of the fire.

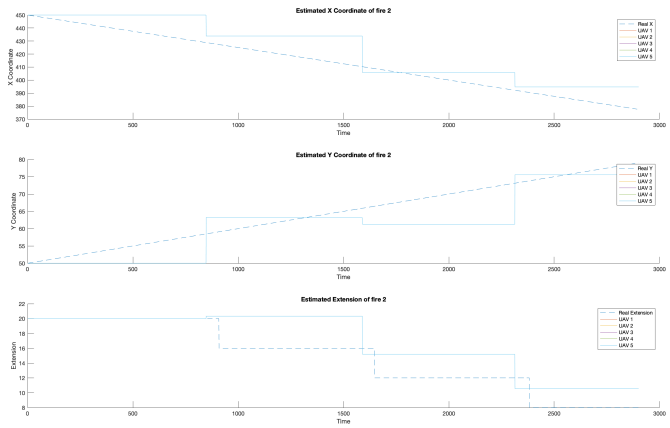


Fig. 9: Evolution of the estimated position and size of Fire 2.

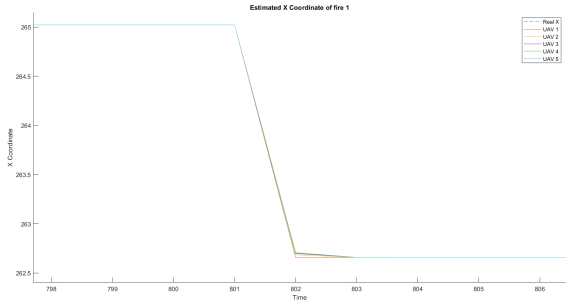


Fig. 10: Zoomed view of the convergence of the x-coordinate estimates for Fire 1.

Figure 11a shows the takeoff phase. Figure 11b depicts the firefighting and refilling phase, during which the UAVs alternate between dropping water on the fire and refilling their tanks. Finally, Figure 11c shows the landing phase, once all the fires have been extinguished.

In Fig. 12, the Voronoi tessellation is shown considering the safety factor and the modified Voronoi algorithm designed to better prevent collisions between drones, as expressed in Eq. 7. Each UAV has a safety area represented by a dashed green circle. Additionally, each UAV is associated with multiple colored points, each representing the virtual position $\tilde{p}_j$ of a neighboring drone j relative to drone i, as computed during the tessellation process. These virtual positions are color-coded to distinguish between different neighboring UAVs. It can be observed that when two UAVs are close to each other, the corresponding virtual points tend to move even closer, effectively pushing the UAVs in opposite directions through the control law.

D. Simulation Evaluation

To assess the performance of our approach, we focus on a key metric that distinguishes autonomous UAVs from human-piloted aircraft: the time interval between successive water drops. As outlined in Section I, shorter intervals should increase firefighting efficiency and reduce the total extinguishing time. Accordingly, we recorded the inter-drop intervals for simulations using different numbers of UAVs. For each configuration, Table I reports the mean interval between consecutive drops under the assumption that all UAVs remain fully operational throughout the mission.

n° UAV	Mean Drop Time Difference	n° Drops
3	235	9
4	163	12
5	143	15
6	129	18
7	139	17
8	135	17
9	138	17
10	132	17
11	122	19
12	110	21
13	96	24
14	95	24

TABLE I: decay factor in water drop intervals with respect to the increasing number of UAVs

As shown in Fig. 13, the mean waiting time between consecutive drops remains roughly constant for fleet sizes between 7 and 10 UAVs, but then decreases steadily as the number increases from 11 to 13. This behavior stems from two competing effects on the Lloyd–Voronoi control algorithm.

On one hand, adding more UAVs should, by reducing idle time, improve coverage efficiency. In practice, however, a "deadlock" can occur when multiple drones converge on the same fire simultaneously. In such cases, each drone's Voronoi-cell centroid freezes in place until an adjacent UAV arrives and perturbs the tessellation. During this stall, the affected drones circle their static centroids, wasting both time and fuel.

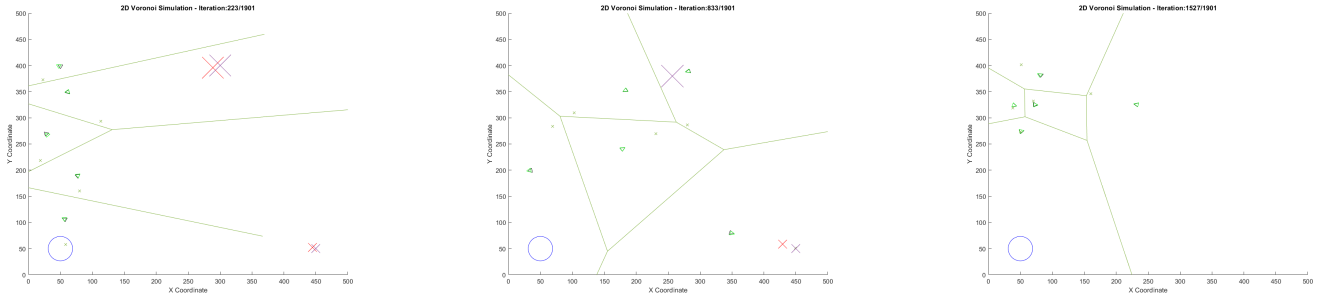
When the fleet size is between 7 and 10, these deadlock events tend to last longer: the probability that another UAV will arrive to break the tie is relatively low. Once the fleet grows beyond 10, however, the chances that another drone will quickly enter the contested cell, and thus resolve the deadlock, rise sharply, shortening the average waiting time. Beyond 14 UAVs, though, aerial congestion becomes so severe that collision risk and control-signal interference begin to degrade system safety and overall performance.

E. Fire Priority Management

Figure 14 illustrates the number of drops on each fire as a function of the number of UAVs deployed. In this simulation, the fire extensions are defined as $\sigma_{f1} = 50\ [m]$ and $\sigma_{f2} = 25\ [m]$. Based on these values, Fire 1 accounts for 66.6% of the total fire area, while Fire 2 covers the remaining 33.3%. Therefore, we initially expected the percentage of drops to roughly reflect this proportion.

However, the observed results diverge slightly from this expectation. When averaging the drop distributions across all simulations, Fire 1 received $\mu_{dropsf1} = 72.6\%$ of the drops, whereas Fire 2 received $\mu_{dropsf2} = 27.4\%$.

Moreover, Figure 14 reveals a noteworthy trend: as the number of UAVs increases, particularly in the range of 7 to 10, where deadlock conditions become more critical, the number of drops on Fire 2 is larger then the average. This suggests that resource contention and coordination challenges among UAVs



(a) Takeoff phase (b) Firefighting and refilling phase (c) Landing phase

Fig. 11: UAV Voronoi tessellations across three mission phases: (a) takeoff, (b) firefighting and refilling, (c) landing.

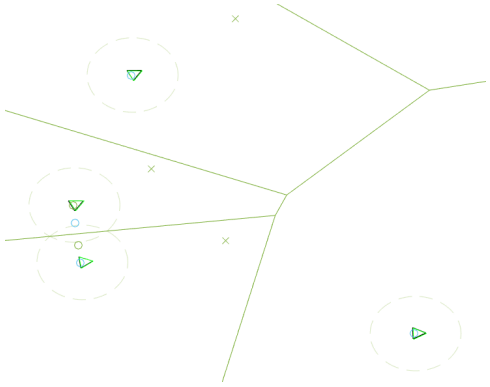


Fig. 12: Voronoi tessellation with safety factor to avoid collisions between UAVs.

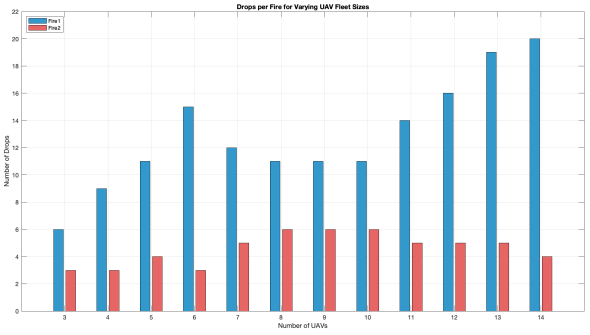


Fig. 14: Histogram showing the number of drops on each fire as a function of the number of UAVs used in the simulation all with the same number of iterations.

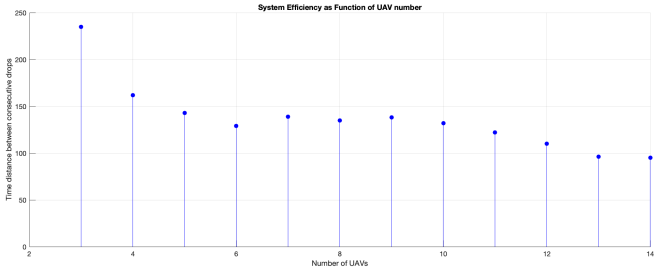


Fig. 13: Visualization of the drops waiting time decrease over the number of UAVs

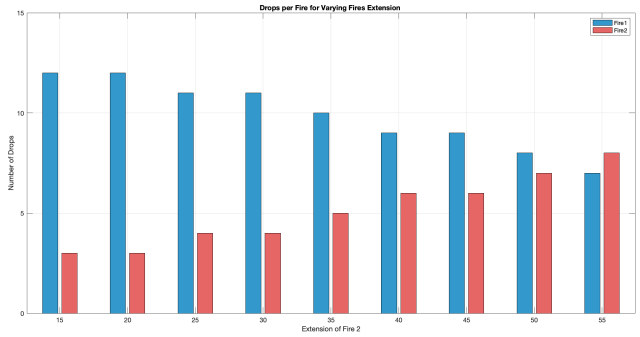


Fig. 15: Histogram showing the number of drops on each fire as a function of the extension of the second fire, while the first is set to a constant size $\sigma_{F1} = 50\ [m]$.

may influence the allocation of firefighting efforts between the two fires.

Figure 15 illustrates how the number of water drops on each fire changes with the relative size of the fires. This histogram summarizes the results of multiple simulations in which the number of UAVs was fixed at 5, the simulation duration was set to 30, and only scenarios without deadlocks were considered. Fire 1 maintained a constant diameter size of $50\ [m]$, while the size of Fire 2 varied from $15\ [m]$ to $50\ [m]$. The results shows how the number of drops change linearly with the size of the fires. The trend confirms the effectiveness of the UAV allocation strategy in responding proportionally to the size of the fire.

VI. Conclusions

In this work, we presented a distributed control system for a fleet of fixed-wing UAVs designed to autonomously detect, track, and suppress wildfires. By integrating Voronoi–Lloyd-based spatial partitioning, an Extended Kalman Filter for robust state estimation, and a linear consensus algorithm for distributed fire localization, the proposed system enables scalable and adaptive coordination among multiple UAVs in dynamic environments.

Simulation results demonstrate the effectiveness of the approach in reducing the interval between consecutive water

drops and ensuring complete fire suppression through persistent monitoring and refilling operations. The modular design of the control logic allows UAVs to seamlessly switch roles between fire suppression, refueling, and returning to base, depending on the mission stage and sensor feedback.

The system provides a solid foundation for real-time, autonomous aerial firefighting.

A. Possible Improvements

Although this work demonstrates a viable distributed fixed-wing UAV network for wildfire mitigation. Several key enhancements could improve realism, performance, and scalability, such as:

- *UAV Modeling*. The current implementation simplifies the UAV as a unicycle in the x, y plane with altitude z control-decoupled from horizontal dynamics. For more realistic applications, the model should incorporate aerodynamic lift as a function of velocity and altitude, and reflect bank-angle–dependent tilting during turns.
- *Control Strategy*. Our guidance logic currently issues high-level commands without considering motor current constraints. A more detailed strategy could compute the exact current profiles required by each motor, enabling real-time control allocation and energy-efficient mission planning.
- *Deadlock Avoidance and Coordination*. We observed deadlock phenomena when deploying more than six UAVs (Section V-D). To address this, future work should explore decentralized conflict-resolution protocols (e.g., priority negotiation or consensus switching) and robust obstacle-avoidance schemes (e.g., velocity obstacles or barrier certificates) to ensure collision-free coordination under high agent densities.

Overall, these improvements will bring the system closer to practical deployment, enhancing the reliability and effectiveness of UAV swarms in real-world wildfire emergency response.