HAL
open science

Real-time wildfire monitoring with a fleet of UAVs

Rafael Bailon-Ruiz, Arthur Bit-Monnot, Simon Lacroix

To cite this version:

Rafael Bailon-Ruiz, Arthur Bit-Monnot, Simon Lacroix. Real-time wildfire monitoring with a fleet of UAVs. Robotics and Autonomous Systems, 2022, 152, pp.104071. 10.1016/j.robot.2022.104071 . hal-03600830

HAL Id: hal-03600830

https://laas.hal.science/hal-03600830v1

Submitted on 7 Mar 2022

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Real-time wildfire monitoring with a fleet of UAVs

Rafael Bailon-Ruiz[a], Arthur Bit-Monnot[a,b], Simon Lacroix[a]

[a]*LAAS-CNRS, Université de Toulouse, CNRS, Toulouse, France*
[b]*LAAS-CNRS, Université de Toulouse, INSA, Toulouse, France*

Abstract

This paper introduces a wildfire monitoring system based on a fleet of Unmanned Aerial Vehicles (UAVs) to provide firefighters with precise and up-to-date information about a propagating wildfire, so that they can devise efficient suppression actions. We present an approach to plan trajectories for a fleet of fixed-wing UAVs to observe a wildfire evolving over time by tailoring the Variable Neighborhood Search metaheuristic to the problem characteristics. Realistic models of the terrain, of the fire propagation process, and of the UAVs are exploited, together with a model of the wind, to predict wildfire spread and plan accordingly the UAVs motions. Algorithms and models are integrated within a software architecture allowing tests with real and simulated UAVs flying over synthetic wildfires. Results of a mixed-reality test campaign show the ability of the proposed system to effectively map wildfire propagation.

Keywords: UAV, Remote sensing, Wildfire monitoring, Multi-robot planning

1. Introduction

Real time monitoring of wildfires is essential to assess the situation and plan effective countermeasures. Current wildfire observation solutions provide information that is either imprecise, incomplete or delayed, *e.g.* from lookout towers or satellites. Manned helicopters and airplanes can provide precise and up-to-date information, but at high costs and risks. Unmanned Aerial Vehicles (UAVs) are a way to overcome the limitations of traditional observation techniques. They can be expeditiously deployed at low cost with no risk for the firefighters, and provide extensive and precise information.

1.1. Related work

The wildfire community is increasingly adopting UAVs: beside avoiding putting pilots at risk, the observation they provide can improve the cost-effectiveness and efficiency of wildfire fighting efforts [1, 2]. Early examples of wildfire observation platforms exploited High Altitude Long Endurance UAVs as a complement to existing satellite monitoring systems [3]. Such UAVs can fly for hours and carry heavy powerful payloads to observe the terrain, but their operational complexity reduces their application extent.

Low Altitude Short Endurance UAVs in single-vehicle configurations are easier to deploy [4]. Equipped with visible or infrared cameras, they can provide in real time accurate information on the fire size, location and perimeter. The precise images can also be used to characterize fire geometry [5, 6], a relevant information for the firefighters to assess the fire severity. The produced fire maps can be used to improve the parameters of a wildfire propagation simulator [7], yielding the possibility to establish wildfire prognosis, which can benefit to decision support tools (Figure 1).

Deploying *a fleet of UAVs* can ensure the achievement of long-lasting wildfire monitoring tasks that provide nearly complete, accurate and up-to-date information over a wildfire span, as well advocated in [5, 8] and proposed in [9]. Building systems upon multiple UAVs implies extra design challenges, as vehicles must collaborate to exploit the full potential of the fleet. The complexity of these interactions has to be managed in real time and requires the automation of the fleet operation. For instance, deciding how to deploy the UAVs to maximize the amount of gathered information requires to assess the situation of both the fleet and fire, which can hardly be handled by an operator. Autonomous control algorithms are necessary to operate the fleet, coupled with data processing abilities providing the sufficient level of situational awareness to uphold the fleet operations.

1.2. Approach

The work presented here introduces the Situation Assessment and Observation Planning (SAOP) system whose purpose is to monitor wildfires with fleets of UAVs in order to provide firefighters with real-time information on the fire perimeters and their evolution over time.

SAOP operates along a Perception – Decision – Action scheme. The UAVs observations are combined into a fire map, on the basis of which which a realistic wildfire simulator produces a fire spread forecast. The fire map and forecast are sent to the operators, but more importantly from our concern, it is used to define an observation plan, that defines the optimal paths for the fleet to observe the fire. The resulting trajectories are sent to the UAVs for

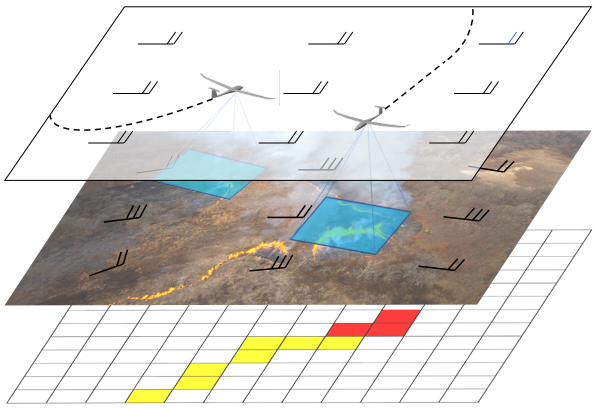


Figure 1: Fleets of UAVs can be used to monitor a wildfire by continuously mapping the location of its perimeter. The proposed approach incorporates observations made by the UAVs (bottom layer, in red) and fire propagation predictions (in yellow).

execution, and the newly gathered information is used to update the fire map.

SAOP is a centralized approach: while local fire maps can be built on-board the UAVs, their fusion, the fire propagation forecast and the establishment of observation plans is run on a ground station that gathers all the necessary informations from the UAVs and sends them the planned observation trajectories.

1.3. Outline

This paper presents the two main ingredients required by the autonomous operation of a fleet of UAVs to monitor wildfires: *situation assessment* and *observation planning*, and depicts the way they are integrated within a modular software architecture.

Section 2 describes the approach to wildfire situation assessment, thanks to the use of a wildfire simulator and an ad-hoc map fusion algorithm that combines predictions and observations. Section 3 is the core of the paper. It provides a formal definition for the Wildfire Observation Problem (WOP) with multiple UAVs and depicts a planning algorithm built upon the Variable Neighborhood Search metaheuristic to devise observation trajectories. Section 4 depicts the software architecture that integrates the situation assessment and observation planning abilities in a mixed-reality simulation framework that is exploited to test the proposed system in a hybrid real-synthetic environment. Two field campaigns that illustrated the proposed approach and architecture with a real UAV and a synthetic wildfire are presented.

2. Wildfire situation assessment

Wildfire situation assessment relies on a continuous loop of the following three processes (Figure 2):

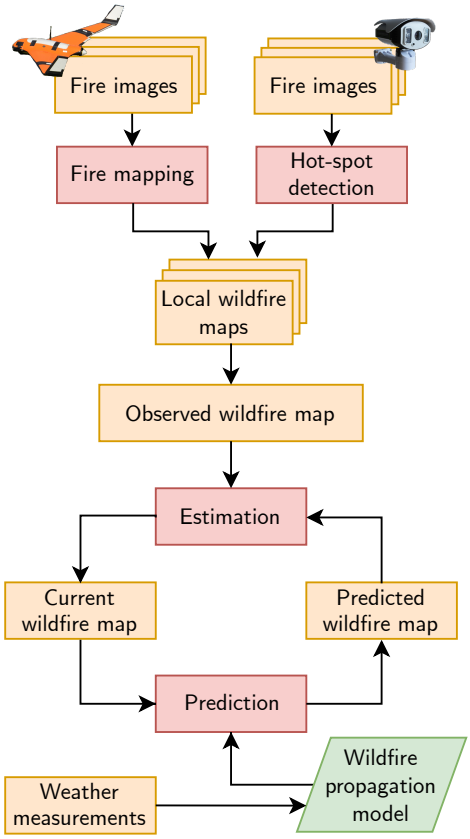


Figure 2: Diagram of the proposed situation assessment process.

1. *Prediction* of future wildfire spread based on the current estimate and the forecast provided by the propagation model (section 2.1).
2. *Observations* by UAVs, in the form of *local* wildfire maps fused into the *observed* wildfire map (section 2.2).
3. *Estimation* of the current state by fusing actual observations with previous forecasts (section 2.3).

The wildfire map is the key piece of information upon which the proposed situation assessment process is defined. It is a raster map structure that encodes the evolution of the location and time of the fire front, through the ignition time of every location. Three instances of maps define the way to communicate information from one step to another:

- The *observed* wildfire map holds the fusion of the *local* wildfire maps derived from UAV observations. It depicts the knowledge about the wildfire spread at the time of its observation.
- The *current* wildfire map depicts the complete current fire front and past propagation. This map is the result of the wildfire estimation algorithm applied to the observed and predicted wildfire maps.
- The *predicted* wildfire map describes the expected future evolution of the wildfire. It is built from the

current wildfire map to which is applied the wildfire propagation model.

2.1. Wildfire propagation simulation

Wildfire propagation simulation provides short term wildfire spread forecasts for situation assessment and observation planning (and also, to generate synthetic fire scenarios for system development purposes). Due to the multiple physical and chemical processes the govern wildfires, no simple model is able to simulate all kinds of wildfires, but existing models [10, 11] are able to provide reasonably accurate forecasts fast enough to be used in real time.

The proposed wildfire simulator handles surface fires (other phenomena like crown and ground fires are not considered, but they could be added if necessary). It combines a local propagation model that describes the fire behavior in a particular location and time with a graph-based algorithm that simulates the fire spread, and outputs a wildfire map, which encodes the ignition time of every location in an area.

Local propagation. An essential ingredient for simulating a fire propagation is to know, for a given ignited point, the speed at which it will spread in any direction. This requires three elements:

- The *main propagation direction* in which the fire spread the fastest, mostly dependent on the wind and terrain slope.
- The *steady-state rate of spread (RoS)*, the propagation speed along the main propagation direction.
- Given the main direction and the RoS, a *shape model* defines the propagation speed in any direction.

The Rothermel forward propagation model [12] provides a way to compute both the main propagation direction and the RoS. The model relates the three environment factors that govern wildfires (fuel, wind and terrain slope) as follows:

$$RoS = \alpha_f(1 + \phi_w + \phi_s) \tag{1}$$

where α_f is a constant that encompasses all fuel-related inputs, ϕ_w is the wind factor, and ϕ_s the slope factor.

The fuel factor α_f depends on the physical and combustion properties of the burning material. It is empirically defined empirically for a standard set of vegetation types [13] that can be associated to land cover maps (e.g. the *CORINE Land Cover* database[1]).

The Rothermel model provides an estimate of the RoS in the main direction, and is completed with a *shape model*, that estimate back and flank fires propagation speed. We implement the double ellipse shape model described in [14].

[1] `https://land.copernicus.eu/pan-european/corine-land-cover`

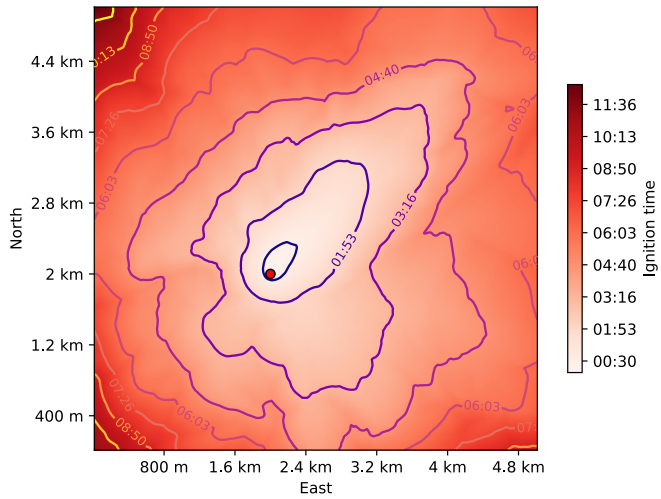


Figure 3: Wildfire propagation map in a 5 km by 5 km area with isochrons. The grid spatial resolution is 25×25m.

Integrating propagation with time. Running a simulation over extended time periods requires a *fire growth model*, several of which were proposed in the literature [15]. We retained the raster model for its simplicity and adaptation to our fire map structure.

The raster map encodes the ignition time of each cell and the simulation propagates the fire from an ignited cell to its neighbors, based on the principles described in [16]. The ignition time of a cell (x, y) is computed with:

$$\begin{aligned} \mathit{ignition}(x,y) = \min_{(x_n,y_n)\in N_{(x,y)}} \{&\mathit{ignition}(x_n,y_n) + \\ &+ \mathit{travel\text{-}time}((x_n,y_n),(x,y))\} \end{aligned} \tag{2}$$

where $N_{(x,y)}$ is the set of the eight neighbors of (x, y) and *travel-time* derives from the propagation speed of the local propagation model in the (x_n, y_n) to (x, y) direction.

This is similar to distance graphs in road networks, where each directed edge gives the *travel time* from one cell to its neighbors. The classic Dijkstra shortest path algorithm calculates the ignition time of all cells.

While fire spreads away from a particular location, fuel is consumed and ignition stops. As a simplification, it is considered that a cell ceases to be on fire when all the adjacent cells have caught fire. The end-of-ignition time $\mathit{ignition}_{end}(x, y)$ is then defined as:

$$\mathit{ignition}_{end}(x,y) = \max_{(x_n,y_n)\in N_{(x,y)}} \{\mathit{ignition}(x_n,y_n)\} \tag{3}$$

The set of cells that constitute the wildfire perimeter at time t are those fulfilling the condition $t \in [\mathit{ignition}, \mathit{ignition}_{end}]$, yielding a propagation map as illustrated in Figure 3.

2.2. Mapping wildfire from images

The problem of fire sensing with UAVs has been solved with computer vision using visual and infrared cameras in

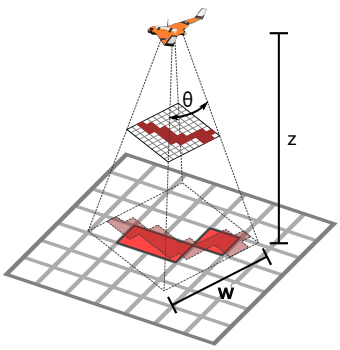


Figure 4: Mapping of the fire detected in a picture to a coarser raster map

combination with various detection algorithms [17]. This subsection describes a simple solution based on a three-step process:

1. *detecting* fire in the UAV-acquired infrared images by applying a threshold on the pixel values,
2. *mapping* by projecting pixels on fire on the ground and then into the local fire map,
3. *combining* local fire maps into the observed wildfire map.

Because the images acquired by the UAV thermal cameras are timestamped and geo-tagged and the digital elevation map (DEM) of the overflown terrain is known, one can easily find the map cells that correspond to the detected burning image pixels using Bresenham's algorithm [18]. Also, as the resolution of the wildfire map (typically 25 meter cells) is coarser than the the image resolution, multiple pixels will lay over one wildfire map cell as illustrated in Figure 4. A simple majority policy is used to mark a cell as on fire. If a cell is observed burning at different times, the oldest detection time is retained.

2.3. Fusion of observed and predicted maps

A handful of UAVs can not deliver a complete up-to-date view of the wildfire situation: it is necessary to estimate the wildfire map in areas where measurements are not available by combining actual observations with forecasts. Data assimilation processes [19, 20] can exploit observations to improve the input parameters of the fire propagation model, resulting in a propagation that better matches the observations. However such approaches require heavy computations and are not applicable for a real-time operational system.

Except in rare extreme conditions, the fire spread function is smooth. Hence we propose to update estimate the whole fire perimeter by combining the predicted map with the observations by warping the shape of the former to match the latter. This is reasonable as long as there are not too large discrepancies between the predicted and observed conditions, which is satisfied with regular updates.

The fusion is handled by an image warping technique. Considering wildfire maps as a function of $\mathbb{R}^2 \rightarrow \mathbb{R}$, like an image, the objective is to define a displacement function $z : \mathbb{R}^2 \rightarrow \mathbb{R}^2$ that stretches the wildfire propagation map so the forecast is coincident with the observed cells (x_o, y_o) as in Equation 4. The idea is to establish a mapping between the observed fire cells seen at time t, and a corresponding cell c_p in the predicted wildfire map with the same ignition time t so the function $\vec{z}(x, y) = (z_x, z_y)$ can be derived from this relationship.

$$(x_p, y_p) + \vec{z}(x_p, x_p) \rightarrow (x_o, y_o) \tag{4}$$

Finding c_p requires exploiting the gradient of the wildfire map that provides the direction of propagation from every cell, which is actually encoded in the propagation graph. Starting from cell c_o, c_p is found by searching for a cell that has the closest ignition time to c_o along the propagation graph.

The displacement of the cells that are not coupled to an observation is interpolated by a smooth function based on the known c_o to c_p displacements. Because nodes are not evenly distributed, a mesh-free interpolation algorithm is necessary. Radial basis function interpolation [21] with thin-plate splines is a suitable choice. It is defined as a weighted sum of radial basis functions $\phi(r)$ evaluated at the interpolation centers (Equation 5), using a function of the thin-plate spline family (Equation 6).

$$z^*(x, y) = \sum_{i=1}^{n} \lambda_i \, \phi \left(|| \, (x, y) - (x_p, y_p)_i \, || \right) \tag{5}$$

$$\phi(r) = -r^2 \ln r^2 \tag{6}$$

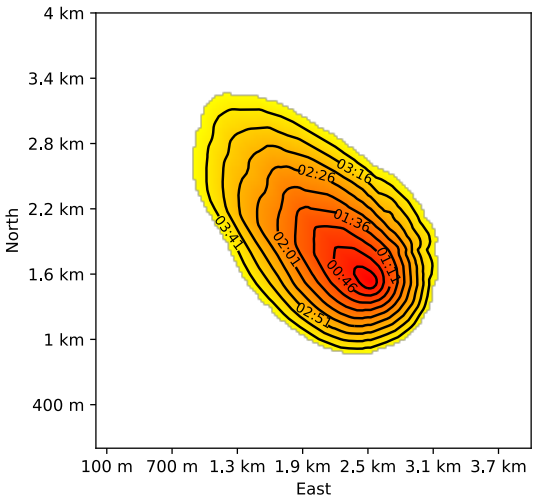
The weights λ_i are calculated by solving a system of linear equations that results from the interpolation requirement so the relationship of Equation 4 is respected for z^* at every $(x_p, y_p)_i$ [22]. Figure 5 depicts an example of the application of the fusion algorithm to a fire map and a set of observations.

3. Observation planning

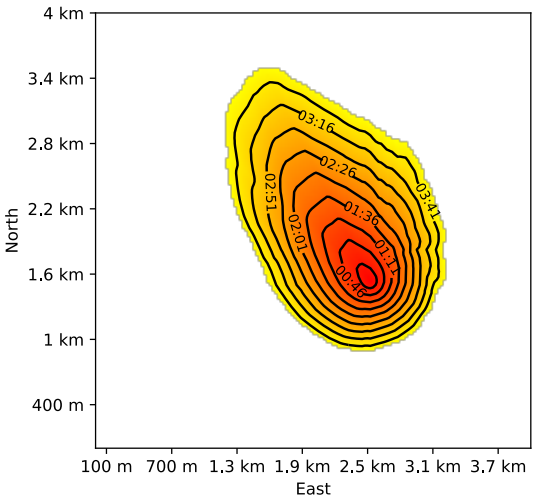
Wildfire monitoring involves following a large perimeter which is constantly evolving. Thus, planning efficient surveillance missions for a fleet of UAVs requires careful observation placement and sequencing with respect to the wildfire spread and UAV motion restrictions. This section proposes a formalization of the *Wildfire Observation Problem* (WOP) and introduces a planning algorithm derived from the Variable Neighbourhood Search (VNS) metaheuristic, equally applicable to single and multiple UAVs systems.

3.1. Challenges

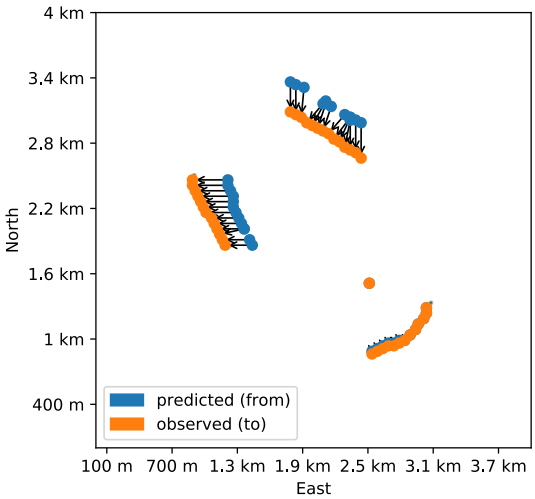
The planning algorithm must efficiently exploit the UAV flying time to observe as much as possible of the fire while



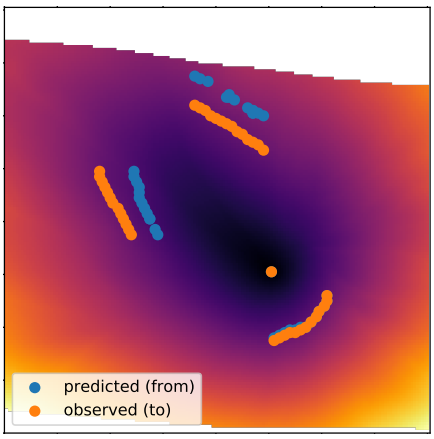
(a) Real wildfire



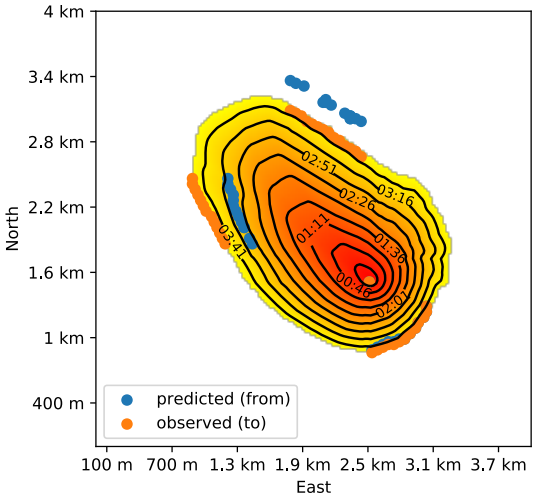
(b) Predicted wildfire



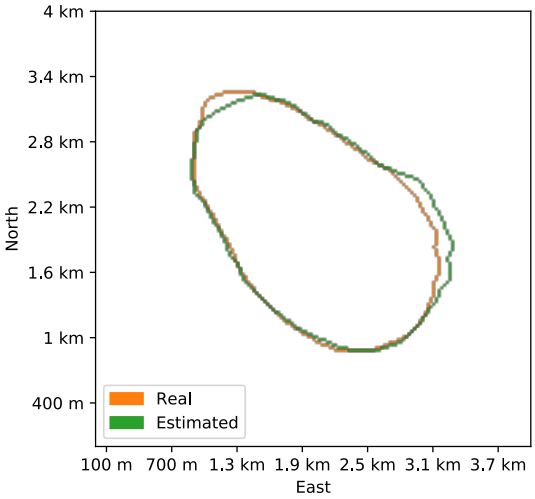
(c) Warping centers



(d) Deformation of the predicted wildfire map



(e) Estimated wildfire map



(f) Comparison between the real and estimated perimeters

Figure 5: Fusion of observed and predicted wildfire maps. 5a is the real wildfire (ground truth), and 5b the predicted fire map from imprecise inputs. UAVs partial measurements of the real wildfire perimeter (orange cells in 5c) are combined with the fire forecast 5b, using the deformation map 5d defined by the connection between predicted and observed cells established with the propagation graph (5c). This yields an estimation (5e) of the real wildfire map, 5f illustrates the ability of this fusion process to estimate a good approximation of the overall fire perimeter size and shape.

considering the various factors that conditions an observation mission. The WOP is defined by a series of characteristics:

1. *Wildfires are only observable at the fire front.* Monitoring of a live wildfire consists on following the evolution of its active perimeter[2].

2. *The wildfire perimeter is dynamic.* Wildfire propagation models provide coarse estimates on fire perimeter propagation that can be used to drive observation planning.

3. *Time is crucial.* A straightforward consequence of *1)* and *2)* is that the observation of active fire contours is constrained within narrow time windows.

4. *Observations must be prioritized.* Fully monitoring one or multiple wildfires over a large area can not be assured with only a handful of UAVs. Information will be inevitably lacking, and hence the selection of areas to observe must be driven by a notion of *utility*, a numerical value that has to be defined.

5. *UAV motions are constrained.* Fixed-wing UAV can only move forward and turn at a limited rate, and the effect of wind on their trajectory is significant, which adds more complexity to the planning problem[3]. Also, flight speed and endurance hamper the maximum spatial and temporal range to survey the terrain.

6. *Observation effort must be balanced between the UAVs.* As several UAVs monitor a large or several fire fronts, the observation plan must balance UAV workload while considering the combined limits in observation reward and motion cost for the whole mission duration.

Overall, the characteristics of the WOP impose many restrictions to the set of possible solutions to explore. Moreover, this is a multi-objective optimization problem that depends on a balance between several criteria and on numerous uncertain factors: finding an absolute optimal solution may not make much sense, and we propose a heuristic approach that efficiently finds solutions of good quality.

3.2. Models

The proposed approach for wildfire observation planning is based on the integration of realistic models to ensure the feasibility and suitability of the plans. Besides the wildfire propagation model introduced in section 2, these are UAV models for perception and motion, and a model of utility that assesses the interest of making an observation.

[2]Observing burnt areas is also an essential task to detect reignitions, but we only focus on the monitoring of live wildfires

[3]Only leveled flights are considered, a constraint imposed by the fire mapping process.

3.2.1. UAV motion model

Planning an observation of a given place and time calls for the definition of feasible and optimal flight trajectories: this is done thanks to the use of a UAV motion model encoding the particularities of flight dynamics, and that allows to estimate the duration of the trajectory.

Definition 1 (Waypoint). *A waypoint w is an intermediate point of the trajectory that a UAV has to reach. A waypoint is represented by a tuple (x, y, ψ) where x, y, correspond to East/North coordinates, and ψ is the course angle*[4]*.*

Definition 2 (Trajectory). *A trajectory T is defined as a tuple (uav, t_0, W) where uav is a UAV motion model, t_0 is the start time and $W = \langle w_0, \ldots, w_n \rangle$ an ordered sequence of waypoints.*

Travel time between consecutive waypoints is computed with the UAV model, which, given a trajectory start time t_0, allows to predict the associated time of arrival $t(w)$ of every waypoint. A trajectory is said to be *valid* iff it satisfies three conditions:

1. The path between any pair of consecutive waypoints must be feasible by uav.

2. The last waypoint w_n must coincide with w_0 or with a safe landing spot defined by the user.

3. The flight duration $(t(w_n) - t_0)$ must be lower than the maximum flight endurance of uav.

Assuming level flight and constant cruise speed, minimum length paths for fixed-wing UAVs are Dubins paths [23], composed of straight lines and minimum radius turns. But the influence wind can not be neglected on the UAV trajectories – especially for the WOP, as wildfires mostly occur in windy conditions. A UAV flying in presence of a steady wind of norm V_w, on a horizontal plane (x, y), at a constant airspeed V_a with a heading angle ψ obeys the following model named *Dubins-wind*:

$$\begin{aligned} \dot{x} &= V_a \cdot cos(\psi) + V_{wx} \\ \dot{y} &= V_a \cdot sin(\psi) + V_{wy} \\ \dot{\psi} &= u \end{aligned} \tag{7}$$

The presence of wind affects the UAV as an additive disturbance (V_{wx}, V_{wy}). As a result, the ground speed $(\dot{x}, \dot{y})$ differs from its constant airspeed V_a, and depends on the UAV heading angle with respect to the wind direction. When flying along straight line segments, the UAV heading angle differs from its flying direction, and constant bank angle trajectories that are circles in the absence of wind become trochoids.

[4]Assuming leveled flights, the altitude z is constant

(a) Wind blowing towards the south

(b) Wind blowing towards the east

Figure 6: Depiction of the effect of wind over Dubins paths. Depending on the wind speed and direction, the path linking the same waypoint sequence can drastically change in shape and length. Steering while facing wind results in a tighter turn radius due to the slower ground speed. Conversely, turning in the direction of wind leads to wider turns.

Consequently, standard circular Dubins paths under the presence of wind are sub-optimal and, while UAV guidance controllers can compensate some wind disturbance in order to follow the original optimal Dubins paths, it is more energy efficient to follow the optimal trajectories derived from the constant wind condition. As depicted in Figure 6, wind direction has a significant impact on trajectory length.

A numerical approach to solve the problem of finding time optimal paths under steady uniform winds is proposed by [24]. The strategy is to find the optimal path with wind by reformulating the problem as finding the no-wind path from a fixed position to a virtual moving destination that drifts in opposite direction to the wind vector. The goal of the redefined problem is to reach the virtual target at the right time with a regular Dubins path. When the planned no-wind path is transformed back by the action of wind, the disturbed path corresponds to time optimal trajectory that reaches the original destination.

3.2.2. Perception model

The perception model provides an estimate of the portions of land that can be observed by the UAV and the fraction of these expected to be on fire. It exploits the current predicted wildfire map, the pose of an UAV at a given time, and the field of view of the on-board camera. The information provided by the perception model is used by the planner to evaluate possible trajectories during the search, using a measure of utility to assess the amount of information they can bring.

Our model assumes a nadir pointed camera, and that only images acquired during straight lines are processed – which is currently a constraint for most commercial UAVs. It simply consists in recovering the map cells covered by the camera footprint on the ground, the state of the cells (burning or not) being used to define the utility of the observation. Figure 7 shows a portion of a UAV flight trajectory exhibiting the cells it allows to observe.

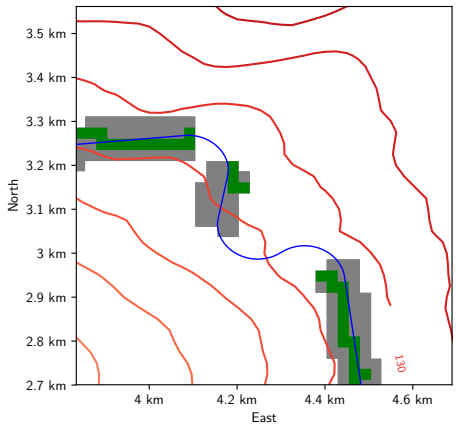


Figure 7: Illustration of the perception model outputs resulting from a UAV trajectory (blue line), overlaid on a predicted fire map (red lines are the fire isochrons). Observed cells are in in gray, and the observed burning cells in green. Note that cells are only observed during straight segments of the trajectory.

3.2.3. Utility model

Utility is a function that allows the observation planner to assess the quality of a solution. For the WOP, it is a performance measurement of the ability to track fire perimeters. Contrary to metrics rating the flight time or distance, defining a utility function considering the extent and the quality of the information acquired so far is not trivial. The utility function should indeed comply with various needs:

- It must favor trajectories leading the UAVs to the fire front.
- The fire front spread is rather regular in both space and time, and its speed is nearly negligible with respect to the UAVs speed. Hence consecutively observing two neighboring burning locations does not give much more additional information compared to

observing just one of them. Similarly, repeatedly tracking the same spot over time is not very advantageous compared to broadly monitoring the wildfire perimeter.

- Short term propagation forecasts are not expected to diverge too much from reality, and their observations should not be favored,
- Finally, the overall wildfire monitoring system is under the control of human operators who task the system by specifying the areas to observe, which is turned into a utility.

These concerns are encoded within a utility map U, that associates each cell c of the fire map to the utility $U(c) \in [0, 1]$ of being observed.

U is initialized with the intrinsic utility $U_B(c)$ that quantifies the information gain of observing a cell c independently of any other observation. Considering that faster parts of the perimeter have to be observed more frequently than slower ones, U_B is scaled linearly with the rate of spread:

$$U_B(c) = U_{min} + \left(\frac{RoS(c) - RoS_{min}}{RoS_{max} - RoS_{min}} \right) * (1 - U_{min})$$

where U_{min} is a constant setting the minimal utility. The utility map is initialized with the intrinsic utilities ($U(c) \leftarrow U_B(c)\ \forall c$).

When an observation is made, the system gains direct information on the observed cell. As highlighted and exploited in section 2, indirect information is also gained on the nearby locations due to the regularity and predictability of fire propagation. We capture this potential indirect information gain on cell c_{ind} after an observation of cell c_{obs} by the function:

$$u_{ind}(c_{obs}, c_{ind}) = U_B(c_{ind}) \cdot \alpha \cdot (1 - dist(c_{obs}, c_{ind})/d_{max})$$

where the indirect information gathered decreases with the distance up to d_{max}, and $\alpha \in [0, 1]$ is a scaling constant that represents the maximum ratio of information that can be indirectly gathered in a single observation.

When an observation is made, the OBS-UTILITY-UPDATE of Algorithm 1 is used to compute the information gain and decrease in U the utilities that remain to be harvested. It computes for each nearby cell c' the indirect information gained $\delta u_{c'}$ and retracts it from U, thus decreasing the potential of a future observation.

The demand of operators to focus on a particular area can be encoded by setting to zero the utility of all observation outside of the observation area.

3.3. Planning problem formulation

The problem to solve is to find an observation mission plan to determine a subset of places deemed on fire to visit by the fleet of UAVs, and in which order, so that the information gain about a wildfire is maximal and an allotted time budget is not exceeded. A plan is made up of trajectories (one per-UAV) confined in time by human operators.

Algorithm 1 Observation utility computation

```
function OBS-UTILITY-UPDATE(c, U)
    δu = U(c)
    U(c) ← 0
    for all cell c' within d_max distance of c do
        δu_c' = min { U(c'), u_ind(c, c') }
        U(c') ← U(c') − δu_c'
        δu ← δu + δu_c'
    return δu
```

Definition 3 (Flight Window). *A flight window $F = (uav, T, d_{max}, [t_{min}, t_{max}])$ represents the opportunity for uav to make a trajectory T and whose duration is at most d_{max}, the maximum flying time allowed by the UAV model, start and end times comprised within the interval $[t_{min}, t_{max}]$.*

Flight windows reflect the UAV temporal allocation for the complete duration of a mission, and allow operators to manage the endurance of individual UAVs and of the fleet long term (for instance, a couple of UAVs can be assigned alternating flying intervals to extend the duration of a monitoring mission).

Definition 4 (Plan). *Given a set of flight windows $F = \{F_0, \ldots, F_m\}$, a plan π associates each flight window F_i to a trajectory $T_i = \pi(F_i)$. A plan has an associated utility value $U(\pi)$ measuring the fitness of the trajectory set to observe a particular wildfire scenario.*

A plan π is *valid* if every trajectory T_i is valid and fits within its flight window F_i.

Definition 5 (Plan Utility). *The utility of a plan $U(\pi)$ is the sum of the utilities collected by each observation in the plan.*

3.4. Variable Neighborhood Search planner

The WOP resembles the Orienteering Problem (OP) [25] of operations research, a variant of the Vehicle Routing Problem. Given a graph where each vertex (observation) is given a score and each edge a traversal cost (travel time), the OP deals with the problem of finding a route through a subset of these vertices that maximizes the collected score without exceeding a travel budget. While the OP remains very simple in its formulation, it has seen multiple extensions that approaches our needs, like The Team Orienteering Problem [26] that considers multiple agents, the Orienteering Problem with Time Windows (OPTW) [27], the Team Orienteering Problem with Time Windows (TOPTW) [28], and the Generalized Orienteering Problem (GOP) that considers nonlinear objective functions [29]. Many variants of the OP and approaches to tackle them

are presented in a comprehensive survey [30]. As the OP is NP-hard, most successful approaches are based on known metaheuristics such as TABU search, Iterated Local Search, Genetic Algorithms, Ant Colony Optimization, and Variable Neighborhood Search (VNS) [28].

The VNS metaheuristic stands out as one of the most effective approaches according to benchmarks [30]. However, none of the surveyed VNS variants perfectly fits the characteristics of the WOP. As a result, the VNS metaheuristic has to be tailored to our problem with compatible UAV perception and motion models as well as a combination of neighborhoods adapted to the continuous and very large space definition of vertices.

3.4.1. VNS basics

VNS algorithms [31] are built on sequences of neighborhoods $\langle \mathcal{N}_1, \ldots, \mathcal{N}_m \rangle$, where each neighborhood aims at improving a particular aspect of the current plan. When applied to an existing plan, a neighborhood generates closely related plans, *e.g.* by swapping the order at which two locations are visited or changing the orientation of a waypoint.

Definition 6 (Neighborhood). *A neighborhood $\mathcal{N}$ defines for each valid plan π a set of neighbor plans $\mathcal{N}(\pi) \subseteq \Pi$ where Π is the set of valid plans.*

Given an initial plan and a sequence of neighborhoods the core of the VNS procedure is descent phase where:

1. One of the neighborhood is used to provide a set of candidate modifications to the current plan.
2. If a modification improves the current plan, it is kept, and the descent restarts from the first neighborhood,
3. Otherwise, the descent switches to the next neighborhood (or finishes if their is none).

Once a local minimum has been reached by the descent phase, a shuffling function can be used to apply random changes to the current solution before starting a new descent phase, with the hope of reaching better solution. The stop condition may be a maximum run time, a maximum number of iterations or a stabilization of the improvement rate.

The key benefit of this metaheuristic lies in its generic and adaptable definition. The VNS algorithm can be tailored to a specific problem by changing how the descent and perturbation phases behave and the sequence in which neighborhoods are explored. The challenge of designing a VNS approach to solve a given problem resides in the formulation of the problem and in the definition of a good set of neighborhoods for solving it in reasonable time.

For the WOP problem, the advantage of using a VNS algorithm is that observation plans are built for the fleet of UAVs as a whole: the problem of allocating UAVs to areas to observe is implicitly solved with careful neighborhood design. Also, as a VNS algorithm works by applying small incremental improvements to a plan, it can be stopped at any time or restarted from an existing plan. The later is especially interesting, because plans can be repaired and improved over time as wildfire forecasts are updated.

Another benefit of a heuristic approach is that good solutions can be rapidly found. Because plans will be updated frequently due to evolving fire conditions, planned monitoring missions do not need to be perfectly optimal.

Unfortunately, the basic VNS scheme is not all the way applicable for wildfire observation planning because the descent phase consisting in finding a local optimum is not feasible. Improving the current solution means adding, removing and updating waypoints, but the space of possibilities is very large. Furthermore, there is no deterministic optimization strategy to follow (no clear direction of descent), and an exhaustive search would be very time-consuming. Instead, the proposed descent strategy relies on sampling to produce a representative set of small local optimizations within the current neighborhood and frequent neighborhood changes.

3.4.2. Neighborhood-specific utility

Unlike in the classical VNS, each neighborhood $\mathcal{N}$ is associated with a utility function $u_{\mathcal{N}} : \Pi \to \mathbb{R}$ that gives the quality of a plan in the context of this neighborhood and may differ from the plan's utility function. For instance, a neighborhood aiming at optimizing trajectories could base its utility function only on the length of the plan. This neighborhood-dependent utility allows greater separation of concerns between different specialized neighborhoods, while the full problem remains mono-objective.

Given a plan $\pi \in \Pi$ and a neighborhood $\mathcal{N}$, the descent exploits this local utility through the *gen-neighbor*$_{\mathcal{N}}(\pi)$ function that returns either (i) a new valid plan $\pi' \in \mathcal{N}(\pi)$ such that $u_{\mathcal{N}}(\pi') < u_{\mathcal{N}}(\pi)$, or (ii) *nil* if the neighborhood failed to generate an improved neighbor.

3.4.3. Shuffling

In order to escape from local optima obtained in the descent phase, we define a perturbation function $\textsc{Shuffle}(\pi) : \Pi \to \Pi$ that produces a new plan by disturbing the plan π. The strategy applied consists in removing a sequence of waypoints from the current plan. The number of consecutive waypoints to be removed in each trajectory is randomly chosen between 0 and the maximum number of waypoints that can be removed, excluding the imposed start and end of the trajectory.

3.4.4. VNS Algorithm

Our VNS approach, fully depicted in Algorithm 2, takes as parameters an initial plan, a sequence of neighborhoods, a shuffling function and a maximum run time. Given an initial (possibly empty) plan $\pi_{initial}$, the descent phase of VNS attempts to generate plan improvements by systematically and sequentially trying all neighborhoods $\langle \mathcal{N}_1, \ldots, \mathcal{N}_m \rangle$ with *gen-neighbor* until a neighborhood $\mathcal{N}_i$ provides an improvement. If an enhancement according to the

Algorithm 2 Pseudo-code of the proposed Variable Neighborhood Search (VNS) algorithm. VNS takes as parameters an initial plan $\pi_{initial}$, a sequence of neighborhoods $\langle \mathcal{N}_1, \ldots, \mathcal{N}_m \rangle$, a real CPU_{max} indicating the maximum planning time and a function $Shuffle$ that is applied to the best plan on a restart.

```
function VNS(π_initial, ⟨N_1, ..., N_m⟩, CPU_max)
    initialization(π_initial)
    π_best ← π_initial
    num-restarts ← 0
    while runtime ≤ CPU_max do
        π ← SHUFFLE(π_best)
        i ← 1                    ▷ Select the first neighborhood
        while i ≤ m do
            π' ← gen-neighbor_{N_i}(π)
            if π' ≠ nil then
                π ← π'           ▷ Update current plan
                if U(π) > U(π_best) then
                    π_best ← π
                i ← 1            ▷ Select first neighborhood
            else
                i ← i + 1        ▷ Select next neighborhood
        num-restarts ← num-restarts + 1
    return π_best
```

plan global utility function is provided, the current plan is updated and the process restarts from the first neighborhood $\mathcal{N}_1$. When no neighborhood is able to generate an improvement, the best plan found so far is perturbed by the shuffling function and the descent phase restarts from the first neighborhood $\mathcal{N}_1$. This process is repeated until the total *runtime* goes over the allowed budget CPU_{max}, at which point the best plan found is returned.

As the VNS approach is able to start from any valid plan, $\pi_{initial}$ can be set to a previously computed plan π_{prev}. In this case, the VNS algorithm is constrained to improve only future parts of π_{prev}. First, an initialization function translates the future waypoints to locations expected to be on fire, removing from the plan the waypoints that can not be translated. Then, the current plan is refined following the same procedure for initial plans.

3.5. Definition of applicable neighborhoods

We define two classes of neighborhoods that have proved useful for the WOP problem:

- An insertion neighborhood
- A path optimization neighborhood

These neighborhoods respectively insert waypoints in trajectories to improve the utility of the plan and reduce the duration of those trajectories so more observations can be made. Both exploit the PROJECTWAYPOINT function, which tweaks the trajectories to observe the fire front, so as to ensure that the selected waypoints bring as much utility as possible.

3.5.1. Projection on fire front

Waypoints in a trajectory are only useful when they allow observing the fire front, i.e., if they are directly above an ignited cell. The PROJECTWAYPOINT function (Algorithm 3) provides a way to move an arbitrary waypoint w so that when reached by a *uav* the cell beneath it is ignited. For this, it computes the arrival time t_w at w from a previous waypoint w_{prev}. If the cell beneath it is not ignited at t_w, it computes the fire propagation direction at w and transfers w to be over the next cell in this direction (or in the opposite direction if the fire front already passed through w). This moves w to be one cell closer to the fire front and is repeated until the fire front is reached.[5]

Algorithm 3 Pseudo-code of the PROJECTWAYPOINT function. The algorithm relocates a waypoint w over a cell where the wildfire is active. The PROJECTTRAJECTORY function does the same thing for an entire trajectory (or any sequence of waypoints).

```
function PROJECTWAYPOINT(w, uav, t_prev, w_prev)
    while true do
        t_w ← t_prev + travel-time(uav, w_prev, w)
        if t_w ∈ [ignition(w), ignition_end(w)] then
            return w
        // compute direction of propagation at w
        p⃗ ← ∇ignition(w)
        // Move w towards the fire front
        if t_w < ignition(w) then
            w ← next-in-direction(w, p⃗)
        else
            w ← next-in-direction(w, −p⃗)
function PROJECTTRAJECTORY(⟨w_0, ..., w_n⟩, uav, t_0)
    for all i ∈ [1, n] do
        w_i ← PROJECTWAYPOINT(w_i, uav, t_{i−1}, w_{i−1})
        t_i ← t_{i−1} + travel-time(uav, w_{i−1}, w_i)
```

This function is useful for the waypoint insertion neighborhood, so that the new waypoint is placed at a valid location, and to correct the placement of the following waypoints (as illustrated figure 8). Also, after an update of the wildfire situation, the observation plan can be refined instead of starting from scratch. In such situations, the PROJECTWAYPOINT function is used to quickly update the original plan.

The associated PROJECTTRAJECTORY function acts similarly on an entire trajectory, shifting all waypoints of the sequence onto the firefront so that they lead to an actual observation. It can be used as a repair procedure to adapt a sequence waypoint when, e.g., a trajectory has been modified (during planning) or to adapt a trajectory to a new knowledge on the fire front (when replanning).

[5] For completeness one can check that the algorithm converges by ensuring that the ignition time at w is monotonically increasing or decreasing.

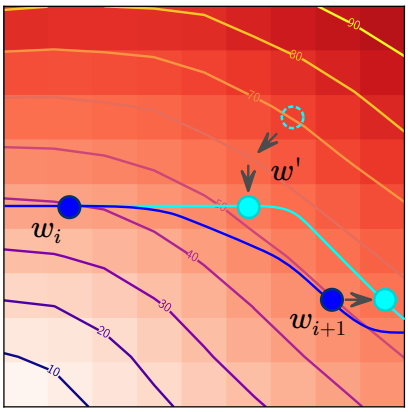


Figure 8: The waypoint insertion process. A randomly chosen waypoint w' (dashed, light blue) is inserted in a trajectory between w_i and w_{i+1} (dark blue). w' is re-projected into a previous isochron (light blue) whose time corresponds to the time needed to reach it from w_i. Finally, due to the increment in travel time between w_i and w_{i+1} caused by the insertion, w_{i+1} is moved to a later isochron.

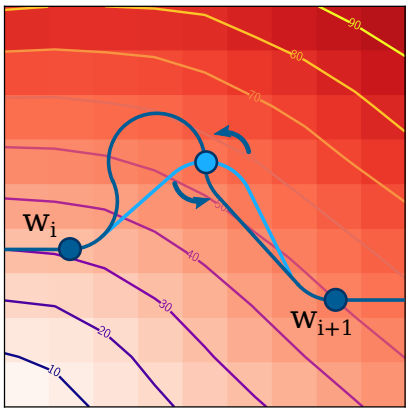


Figure 9: Illustration of the local path optimization process. The dark blue curve represents the original trajectory and the light blue curve the optimized one. The improved path contains longer straight line sections so more portions of the wildfire can be observed.

3.5.2. Insertion neighborhoods

A *waypoint insertion* neighborhood alters a plan by inserting a new waypoint w' in a trajectory T.

Since the number of potential waypoints is large, new waypoints are obtained through sampling. The insertion procedure selects a location at random and tries to find the best insertion order in a trajectory and across trajectories. Each candidate waypoint goes through ProjectWaypoint so the UAV is able to reach it within the associated $[ignition, ignition_{end}]$ range of the underlying cell.

Given a random waypoint w and a trajectory T with waypoints $\langle w_0, \ldots, w_n \rangle$, we construct a neighbor for each $i \in [0, n-1]$ by *(i)*: inserting after w_i the w' resulting from the use of ProjectWaypoint to place w in a valid location (feasible by the UAV and situated over an active fire perimeter when reached) and *(ii)*: adapting the rest $\langle w_{i+1}, \ldots, w_{n-1} \rangle$ of the trajectory with the ProjectTrajectory function to account for the added delay in reaching w_{i+1} (Figure 8).

The utility of a neighbor plan is assessed by the plan utility function, with ties broken by trajectory duration.

Several *waypoint insertion* neighborhoods can be defined that differs in how the expanded trajectory – and thus the affected UAV – is selected.

3.5.3. Path optimization neighborhoods

Waypoint orientation has an important effect on Dubins path length, and can greatly extend a trajectory with pointless turns. A path optimization neighborhood is necessary to reduce flight time, so more waypoints could be added within the limited flight duration.

Path length optimization of all waypoints being intractable, a local path optimization neighborhood applies a stochastic or deterministic rotation to a randomly chosen waypoint in the plan with the objective of reducing the duration of a trajectory. While the main purpose of waypoint rotation is not to improve the global plan utility value, well oriented waypoints typically yield improvements to the observation utility as the optimized path contains longer straight flight sections (Figure 9).

The path optimization neighborhood works by picking a random modifiable waypoint w_i (different from the take-off and landing spots w_0 and w_n) from a trajectory $T_j \in \pi$. Then, an orientation changer generator proposes a new orientation angle for this waypoint and the travel time from and to this waypoint is re-computed. If the candidate reduces the length of the trajectory, the move is accepted. Otherwise, the operation is repeated until a move that improves trajectory length is found or a maximum number of trials is reached.

Three orientation change generators have been found useful: *Random rotation*, *Flip* and *Mean angle*. The third generator is inspired from [32] and smooths the trajectory by giving a waypoint w_i the mean angle between w_{i+1} and w_{i-1}. These heuristics efficiently fix badly placed waypoints that are not oriented along the overall trajectory direction, which force UAVs to perform pointless sharp turns (as in Figure 9).

3.6. Implementation details and illustrations

This section provides some insight on the observation planner implementation and illustrates its application over typical wildfire scenarios.

3.6.1. VNS implementation details

Successful tailoring of the VNS metaheuristic to a particular problem requires thoughtful neighborhood design. The order according to which the neighborhoods are called by the VNS planner, the total planning time and the sampling parameters impact the efficacy of the planner. This subsection summarizes the results of [33] where several neighborhood configurations were evaluated on a previous version of the VNS planner. While improvements have

been made since, especially on the definition of observations, the findings about the configurations are still valid.

The best sequence of neighborhood structures found is composed of one instance of the path optimization neighborhood $\mathcal{N}_{opt}$ and three instances of the insertion neighborhood $\mathcal{N}_{ins}$:

$$\langle \mathcal{N}_{opt}, \mathcal{N}_{ins}^{all-best}, \mathcal{N}_{ins}^{1-best}, \mathcal{N}_{ins}^{rand} \rangle$$

Both the $\mathcal{N}_{opt}$ and $\mathcal{N}_{ins}$ classes of neighborhood rely upon sampling in order to propose improvements to the current plan. The number of samples introduce a trade off between plan quality and time spent. $\mathcal{N}_{opt}$ is configured to try 100 orientation-change samples for the illustrative examples in this section. The $\mathcal{N}_{ins}^{all-best}$, $\mathcal{N}_{ins}^{1-best}$ and $\mathcal{N}_{ins}^{rand}$ are set to 50, 200 and 200 waypoint-insertion samples respectively.

The difference between the three variations of $\mathcal{N}_{ins}$ resides in the specific strategy followed to insert random waypoints into plan trajectories:

1. $\mathcal{N}_{ins}^{all-best}$ systematically tries all possible insertion locations in every trajectory of the plan. As a result, this neighborhood tends to produce plans with large utility improvements where a single UAV is in charge of a group of nearby observations.

2. $\mathcal{N}_{ins}^{1-best}$ picks one trajectory at random and finds the best place to insert the waypoint.

3. $\mathcal{N}_{ins}^{rand}$ takes a random waypoint and inserts it in a random position of a random trajectory. Contrary to $\mathcal{N}_{ins}^{all-best}$, $\mathcal{N}_{ins}^{rand}$ produces neighbor plans with lesser utility improvements but favoring expansion to unexplored areas.

$\mathcal{N}_{ins}^{all-best}$ allows to quickly build an initial solution by promoting highly rewarding neighbor plans. However, once $\mathcal{N}_{ins}^{all-best}$ fails to generate improved solutions, the planner falls back to the broader $\mathcal{N}_{ins}^{1-best}$ and $\mathcal{N}_{ins}^{rand}$ neighborhoods.

The effect of $\mathcal{N}_{opt}$, which the VNS systematically goes back to on a plan update, is the reduction of trajectory duration so that more waypoints can be inserted afterwards. A secondary benefit of $\mathcal{N}_{opt}$ is the production of trajectories that are smooth.

3.6.2. Illustrations

The behavior of the VNS planner is illustrated over three distinct wildfire situations with single and multiple UAVs monitoring plans (statistical evaluations on a series of problems can be found for a former version of the planner in [33]). Figure 10 is a typical situation where an ongoing fire has to be mapped by UAVs with sufficient endurance to observe the complete perimeter. Figure 11 considers a bigger fire and reduced UAV autonomy. Figure 12 shows the case of multiple active wildfires of different sizes. In the 3 cases, results obtained with a single UAV and three UAVs are shown.

Observation plans have been generated with 30 seconds of planning time on an 2.90 GHz Intel Core i7-7820HQ CPU. UAVs in situation 1 and 3 have 30 minutes of flight endurance, and only 15 minutes in situation 2. Figure 10a and Figure 10b demonstrate the priority to observe the front fire with respect to the backfire as trajectories contain more waypoints over the former than the later. When UAVs do not have enough endurance to cover the complete perimeter (Figure 11a), the efforts are distributed (Figure 11b). In situation 3, UAVs have sufficient endurance and the planner finds the best way to switch between independent perimeters (Figure 12a). Although the trajectories in Figure 12b look redundant, UAV passage times are different: in this case adding more UAVs does not provide more information about the total perimeter, but more frequent updates as UAVs are scattered.

Impact of wind. Figure 13 and Figure 14 show situations where strong winds have a great influence on the resulting trajectory. In both cases, different flight patterns can be seen, depending on whether the UAV is flying into or against the wind: when traveling with tailwind, the added ground speed allows covering more areas in less time. Conversely, the extra speed results in wider turn radius that make tight trajectories more difficult and, at the end, require longer paths.

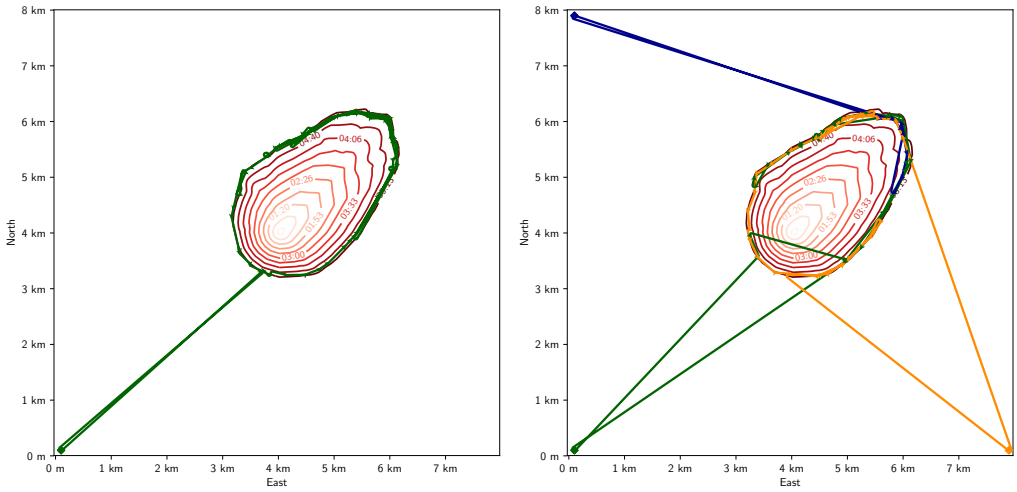
4. Integration, simulation and field tests

4.1. Integration architecture

The mapping and observation planning functionalities are integrated within a wholesome architecture that comprises the UAVs system and a supervision scheme (Figure 15), relying on the Robot Operating System (ROS) middleware.

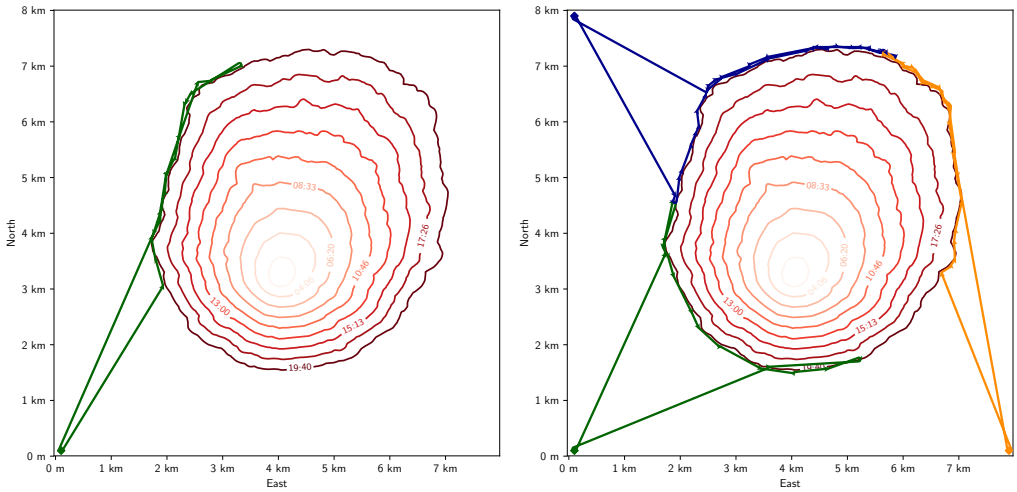
The situation assessment functionality is distributed across multiple locations of the SAOP system. Wildfire perception and mapping algorithms run on-board UAVs embedded computer, and the prediction of wildfire spread and fusion with the observed wildfire maps is done on ground. This separation is mainly due to communication constraints, as fire maps are smaller than images and do need to be fused at every image acquisition, and also because the fusion process is centralized, on the CPU that handles fire propagation. The observation planning process runs on ground, using the maps outputted by the situation assessment subsystem. The UAV platform is managed by the LSTS toolchain [34], a software suite developed by the University of Porto for the operation of heterogeneous fleets of unnamed aerial and marine vehicles.

In a fully autonomous system, a complete supervision component should be in charge of the control and synchronization of all the system processes. However, we opted for a less ambitious scheme, yet more realistic in current operational contexts: the supervision component is mainly the interface with the system operator (Figure 16). Through



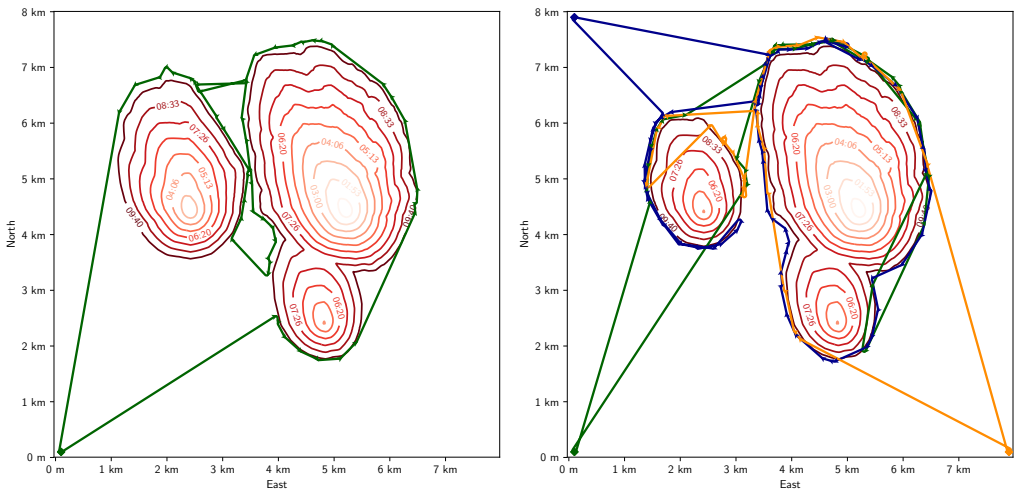
(a) One UAV (b) Three UAVs

Figure 10: Observation plans for situation 1



(a) One UAV (b) Three UAVs

Figure 11: Observation plans for situation 2



(a) One UAV (b) Three UAVs

Figure 12: Observation plans for situation 3

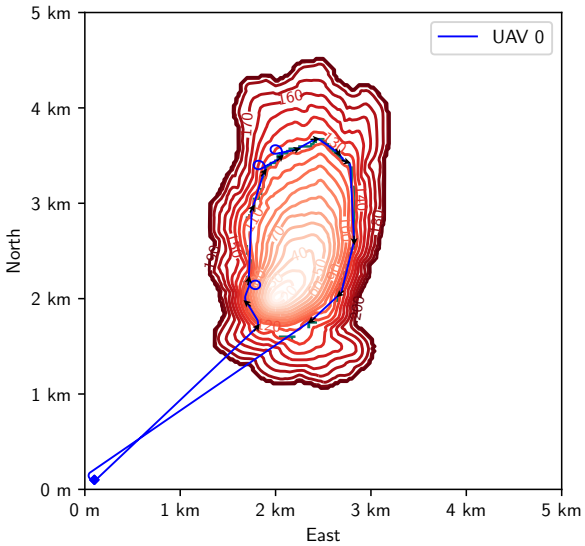


Figure 13: Illustration of an observation plan with strong wind blowing from the south. The twisting paths while flying northwards elongate the trajectory duration, but induce longer straight segments.

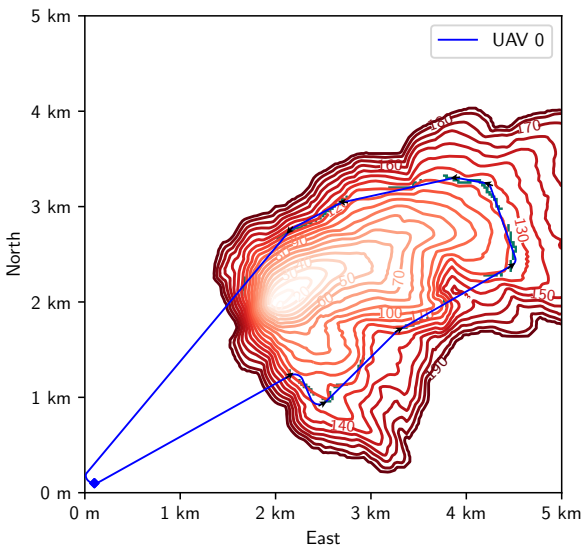


Figure 14: Illustration of an observation plan with strong wind blowing from the west. Note the turn radius is wider in the outbound part of the trip (south) than in the return section (north) due to the effect of wind.

this interface, the operator coordinates the behavior of SAOP components by issuing action commands, such as *Create plan*, *Stop plan* or *Predict wildfire*. It can also specify an area to observe, select which UAVs should participate in the mission, or decide the time to start the mission.

4.2. Mixed-reality simulation

Testing and experimenting the SAOP wildfire monitoring system in real life is of course challenging because of the nature of wildland fires. Prescribed or controlled fires could be good opportunities, but such fires can not be repeated twice with the same conditions, hampering the improvements at the system and sub-systems levels. Also, UAV operations are still tedious, time-consuming and costly, all the more for a UAV fleet.

Simulation is the only way to obtain arbitrary repeatable wildfire test scenarios, and is of course of primary importance for early testing and validation. We opted for a mixed-reality approach, similar to [35], which allows to have the either simulated or actual UAVs in the loop, while simulating the occurrence of a wildfire. The simulation leverages on the LSTS toolchain to simulate UAVs or to control actual UAVs, and exploits the Morse robotics simulator [36] to simulate wildfire image acquisitions according to each UAV position and orientation, be they real or simulated.

4.3. Field tests

The overall system has been tested during two demonstration campaigns held in Vigo (Spain) and near Porto (Portugal) in April and May 2019 respectively, in the context of the FireRS project[6] in collaboration with the University of Porto and the University of Vigo. The objectives of the tests were to assess the validity of the proposed approach and software integration, and to showcase the FireRS project to the media.

First tests. These first tests were meant to assess the communication infrastructure of SAOP with the UAVs. Upon the reception of an alarm that identifies a localized wildfire ourbreak, SAOP predicted the fire spread and generated an observation plan (Figure 17) – no actual UAV flights were planned for these tests.

Tests with a UAV. This campaign focused on wildfire observation planning and actual flights with a UAV, at a location near Porto (Figure 18). Because of the impossibility to have a real fire in this location, we resorted to the mixed reality simulation framework, with a simulated wildfire.

The goal was to test the ability of SAOP to map of a wildfire perimeter with an X8 flying wing UAV[7] operated by the University of Porto. Due to communication

[6] wildFIRE Remote Sensing, `http://www.fire-rs.com`
[7] `https://www.lsts.pt/index.php/systems/60`

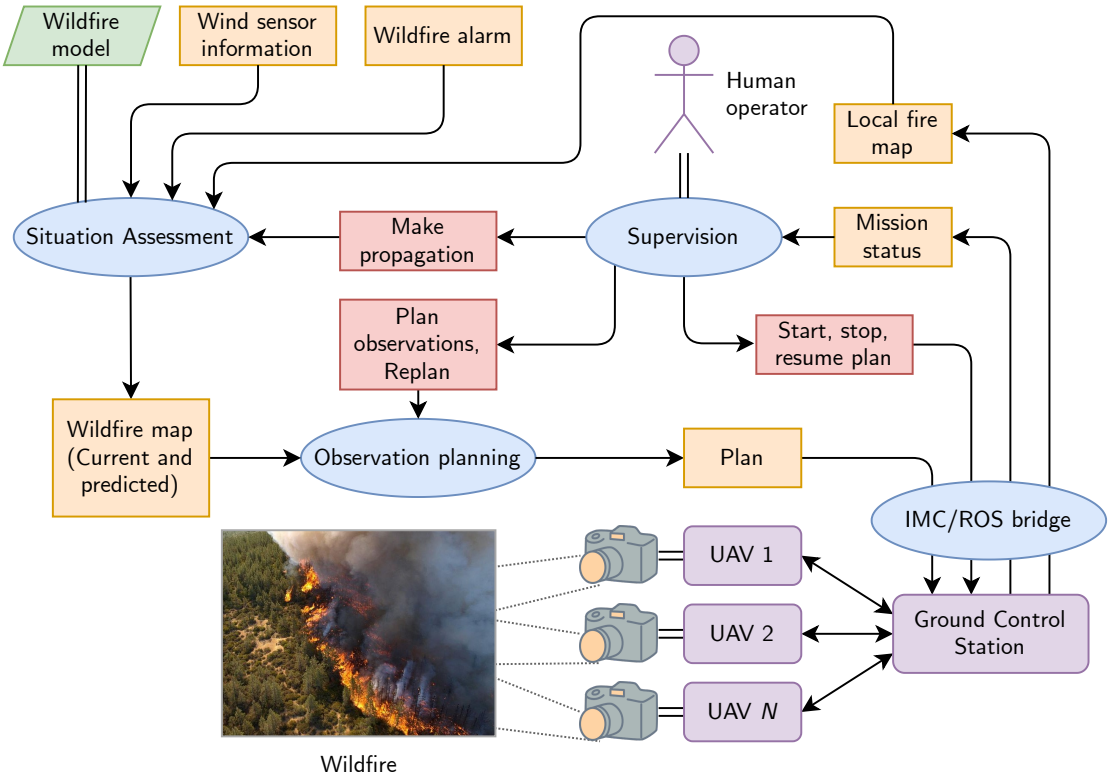


Figure 15: The SAOP system architecture. Blue ovals are ROS nodes, orange boxes represent data messages and red boxes command messages.

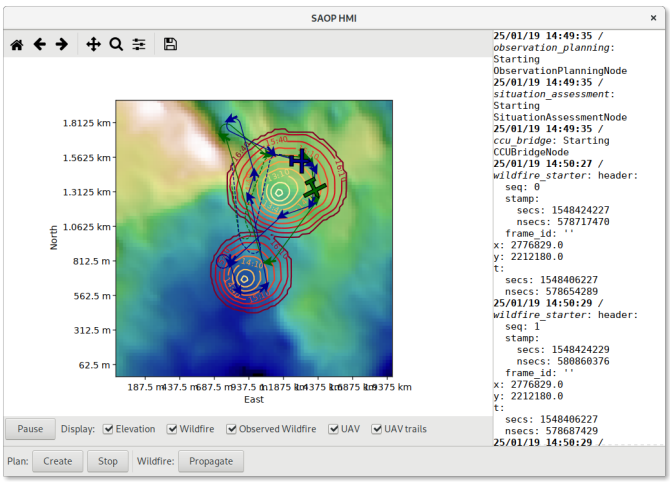


Figure 16: Screenshot of the SAOP human-machine interface

issues with the network between the UAVs and SAOP on the control station, the fire mapping module ran off-board rather than on the embedded UAV computer (processing images generated with the Morse simulator): this modification could rapidly be implemented on-site, thanks to the modular system architecture.

A first test showcased the execution of the flight plan produced by SAOP after a simulated wildfire outbreak, with the UAV initially in a flying loitering state. Two additional tests demonstrated the whole mapping / prediction / planning sequence over the front of the simulated wildfire 30 min and 1 h after the ignition. Figure 19 illustrates some of the results of SAOP during these tests.

5. Conclusion

This paper has introduced the design of a wholesome wildfire monitoring system using fleets of fixed-wing UAVs. The main contribution is the definition of an approach to tackle the Wildfire Observation Problem, resorting to a VNS approach to produce realistic observation plans, handling altogether the allocation and trajectory definition problems. A wildfire situation assessment process, that estimates the current wildfire spread from partial observations of the fire perimeter has also been presented and integrated. The approach exploits realistic UAV motion and perception models, as well as a realistic fire propagation model. It has been integrated in a modular global

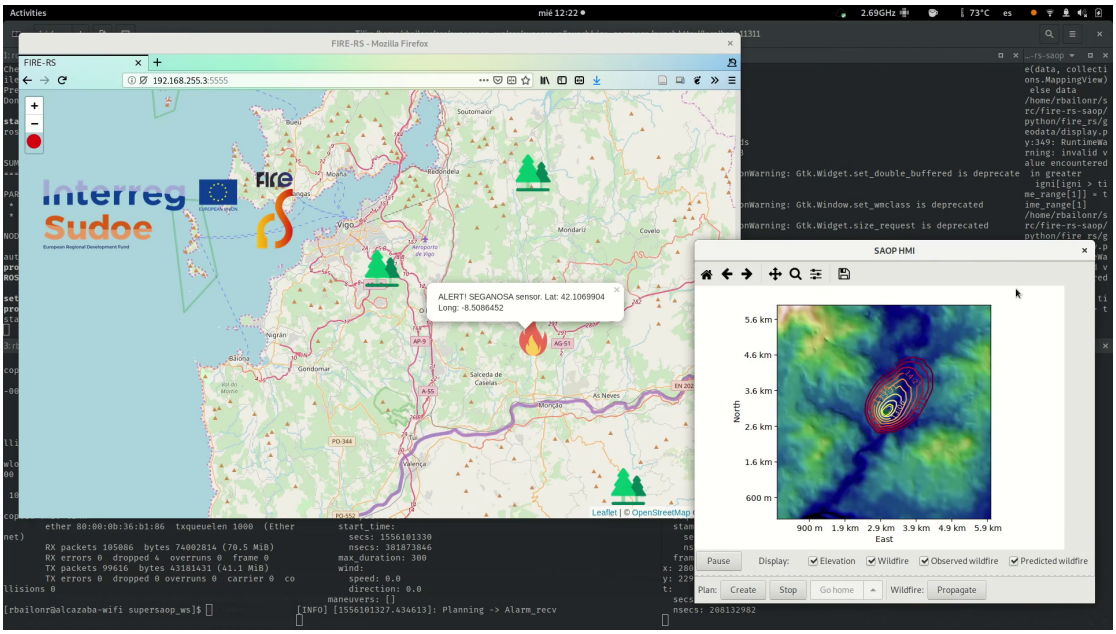


Figure 17: Screenshot of the alarm reception interface depicting the position of a detected fire alarm (left) and the SAOP HMI showing the prediction of the fire spread.

Figure 18: Detailed map of the flight site near Porto (Portugal). The 500 m radius red circle represents the area where the UAVs are allowed to operate, centered on the airstrip used for taking off and landing. The *flame* tag denotes the location of the simulated fire ignition.

architecture, validated in simulations and field tests, resorting to mixed reality scenarios.

The presented SAOP system is a proof-of-concept, within which various revisions and improvements must be achieved so as to foster the development of actual operational systems. We highlight below the three improvements which we deem essential, and sketch the way we think they should be tackled.

Wildfire situation assessment

Besides the integration of state of the art fire observation exploiting infrared and visible cameras, and of fusion algorithms that would explicitly handle the observation and prediction uncertainties, the wildfire situation assessment could benefit from several improvements.

By allowing to estimate the key parameters that govern the wildfire propagation (mainly the fuel information), data assimilation processes would allow to make more reliable predictions, as well as more precise estimates of the uncertainties. No solution able to process the gathered data in real time is currently available, but the future should bring the technological advances that will allow the deep integration of data assimilation into autonomous wildfire monitoring. In turn, having better estimates of the mapping and propagation uncertainties would allow a more sensible definition of the utility function, which is the main driver of the planning process.

But however precise are the models and the detection, propagation and fusion processes, the complexity of wildfires is such that the predictions will always yield uncertain forecasts, that will eventually threaten the quality of the observation plans. A sensible solution would be to endow the UAVs with the ability to adapt their trajectories in

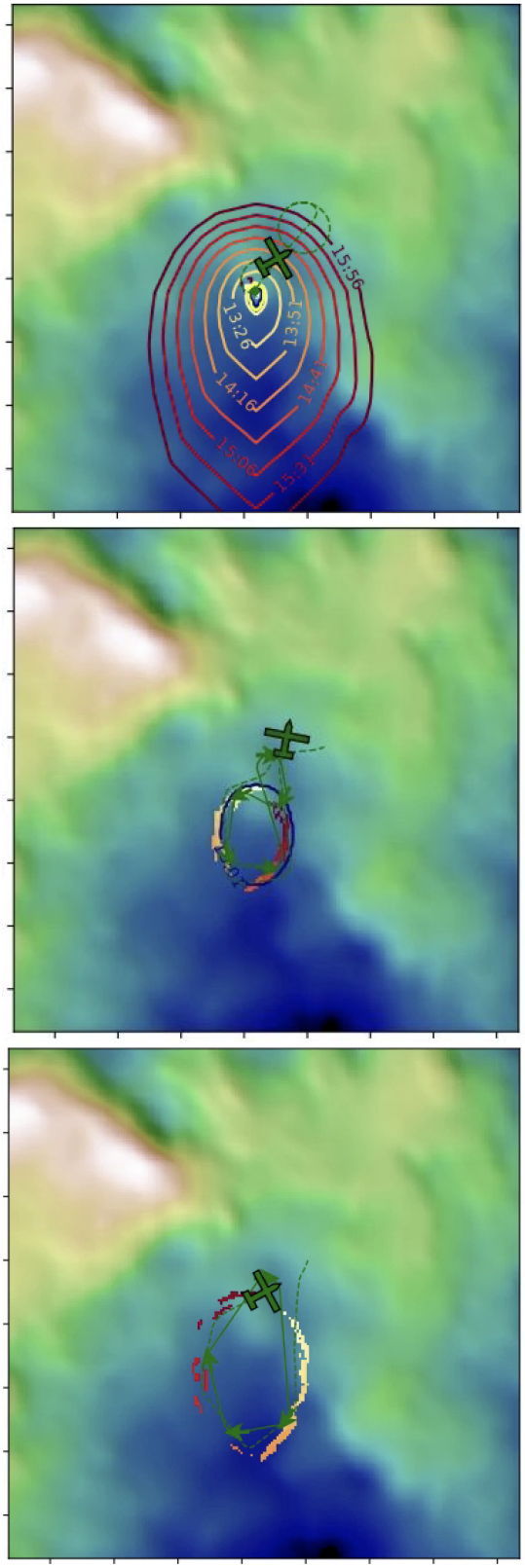


Figure 19: Illustrations of the SAOP results. From top to bottom: predicted fire spread after the reception of the fire outbreak; planned and executed flight at the end after a first call to SAOP with the mapped fire; and mapped wildfire after 30 minutes of execution. Note the mapped fire spread over time (the shown area size is 1.8km × 1.8km, tics are every 250m).

a closed loop with fire detection and local mapping, so that they can be controlled at a higher level (e.g. "track the fire perimeter") than with a sequence of waypoints to reach. Introducing such actions within the VNS planning can easily be done.

Controlling the UAVs altitude

We omitted to consider altitude variations for the UAV motion model. While this is a reasonable simplification for mostly flat terrains, it is in mountainous areas that wildfires raise the most difficulties, from any point of view: controlling the UAVs altitude then becomes necessary, so as to avoid collisions and improve the wildfire observations. As of today, small fixed-wings UAVs mostly interleave levelled flights with altitude changes using spiral-like flight patterns, but the extension of Dubins paths to consider 3D trajectories is tractable [37, 38, 39]. A straightforward consequence of introducing such trajectories in our approach would be a significant increase of the complexity of our waypoint-based planning approach, due to a sharp increase of the number of possible waypoints to sample from. More efficient sampling techniques could be defined, but more interestingly it is the introduction of higher level actions of the UAVs (e.g. "climb"), similar to the autonomous fire tracking ability mentioned above, that will allow to alleviate the algorithmic complexity of planning.

Dealing with long term operations

Our wildfire observation planner manages the UAV endurance through the definition of time windows and maximum trajectory durations, and is hence adapted to a time scale in the order of hours – the UAVs endurance. The planner has an overall greedy behavior, as there are always some locations to observe that increase the overall plan utility. As a result the UAV endurance can be depleted too early, impeding operations in the long run. Wildfires can however last up to several days, and call for an other level of fleet mission management, e.g. developed within a constraint satisfaction framework.

Adding this fleet management layer does not preclude the use of our planer, but it also calls for a higher supervision level. As of today in SAOP, supervision is mainly playing the role of a thin abstraction layer issuing high-level commands to components. While this allows for some operational autonomy, as it frees the user from the definition or low-level tasks or trajectories, the addition of a higher planning layer would call for more supervision capacities, e.g. to let the operators decide whether monitoring plans can allocate resources aggressively or conservatively, and also to help them control the long-term fleet capacities jointly with the higher level mission planner.

Acknowledgments. This work was partially supported by the European V Sudoe Interreg program, through the funding of the FireRS project.

References

[1] B. R. Christensen, Use of UAV or remotely piloted aircraft and forward-looking infrared in forest, rural and wildland fire management: Evaluation using simple economic analysis, New Zealand Journal of Forestry Science 45 (Sep. 2015). doi:10.1186/s40490-015-0044-9.

[2] D. Twidwell, C. R. Allen, C. Detweiler, J. Higgins, C. Laney, S. Elbaum, Smokey comes of age: Unmanned aerial systems for fire management, Frontiers in Ecology and the Environment 14 (6) (2016) 333–339. doi:10.1002/fee.1299.

[3] V. G. Ambrosia, S. Wegener, T. Zajkowski, D. V. Sullivan, S. Buechel, F. Enomoto, B. Lobitz, S. Johan, J. Brass, E. Hinkley, The Ikhana unmanned airborne system (UAS) western states fire imaging missions: From concept to reality (2006-2010), Geocarto International 26 (2, SI) (2011) 85–101. doi:10.1080/10106049.2010.539302.

[4] A. Restas, Forest fire management supporting by UAV based air reconnaissance results of Szendro Fire Department, Hungary, in: First International Symposium on Environment Identities and Mediterranean Area, IEEE, Corte-Ajaccio, France, 2006, pp. 84–88.

[5] J. R. Martínez-de Dios, L. Merino, F. Caballero, A. Ollero, Automatic Forest-Fire Measuring Using Ground Stations and Unmanned Aerial Systems, Sensors 11 (6) (2011) 6328–6353. doi:10.3390/s110606328.

[6] V. Ciullo, L. Rossi, T. Toulouse, A. Pieri, Fire geometrical characteristics estimation using a visible stereovision system carried by unmanned aerial vehicle, in: 2018 15th International Conference on Control, Automation, Robotics and Vision (ICARCV), IEEE, Singapore, 2018, pp. 1216–1221.

[7] M. M. Valero, O. Rios, C. Mata, E. Pastor, E. Planas, An integrated approach for tactical monitoring and data-driven spread forecasting of wildfires, Fire Safety Journal 91 (Supplement C) (2017) 835–844. doi:10.1016/j.firesaf.2017.03.085.

[8] V. C. Moulianitis, G. Thanellas, N. Xanthopoulos, N. A. Aspragathos, Evaluation of UAV Based Schemes for Forest Fire Monitoring, in: Advances in Service and Industrial Robotics, Mechanisms and Machine Science, Springer International Publishing, Cham, 2019, pp. 143–150. doi:10.1007/978-3-030-00232-9_15.

[9] D. W. Casbeer, D. B. Kingston, R. W. Beard, T. W. McLain, Cooperative forest fire surveillance using a team of small unmanned air vehicles, International Journal of Systems Science 37 (6) (2006) 351–360. doi:10.1080/00207720500438480.

[10] A. L. Sullivan, Wildland surface fire spread modelling, 1990 - 2007. 1: Physical and quasi-physical models, International Journal of Wildland Fire 18 (4) (2009) 349. doi:10.1071/WF06143.

[11] A. L. Sullivan, Wildland surface fire spread modelling, 1990 - 2007. 2: Empirical and quasi-empirical models, International Journal of Wildland Fire 18 (4) (2009) 369. doi:10.1071/WF06142.

[12] R. C. Rothermel, A mathematical model for predicting fire spread in wildland fuels, USDA Forest Service Research Paper INT-115 (1972).

[13] J. H. Scott, R. E. Burgan, Standard fire behavior fuel models: A comprehensive set for use with Rothermel's surface fire spread model, Tech. Rep. RMRS-GTR-153, U.S. Department of Agriculture, Forest Service, Rocky Mountain Research Station, Ft. Collins, CO (2005). doi:10.2737/RMRS-GTR-153.

[14] H. E. Anderson, Predicting wind-driven wild land fire size and shape, USDA Forest Service Research Paper INT (USA) (1983).

[15] A. L. Sullivan, Wildland surface fire spread modelling, 1990 - 2007. 3: Simulation and mathematical analogue models, International Journal of Wildland Fire 18 (4) (2009) 387. doi:10.1071/WF06144.

[16] M. A. Finney, Fire growth using minimum travel time methods, Canadian Journal of Forest Research 32 (8) (2002) 1420–1424. doi:10.1139/x02-068.

[17] C. Yuan, Y. Zhang, Z. Liu, A survey on technologies for automatic forest fire monitoring, detection, and fighting using unmanned aerial vehicles and remote sensing techniques, Canadian Journal of Forest Research 45 (7) (2015) 783–792. doi:10.1139/cjfr-2014-0347.

[18] J. E. Bresenham, Algorithm for computer control of a digital plotter, IBM Systems journal 4 (1) (1965) 25–30.

[19] M. C. Rochoux, B. Cuenot, S. Ricci, A. Trouvé, B. Delmotte, S. Massart, R. Paoli, R. Paugam, Data assimilation applied to combustion, Comptes Rendus Mécanique 341 (1) (2013) 266–276. doi:10.1016/j.crme.2012.10.011.

[20] O. Rios, E. Pastor, M. M. Valero, E. Planas, Short-term fire front spread prediction using inverse modelling and airborne infrared images, International Journal of Wildland Fire 25 (10) (2016) 1033–1047. doi:10.1071/WF16031.

[21] F. Bookstein, Principal warps: Thin-plate splines and the decomposition of deformations, IEEE Transactions on Pattern Analysis and Machine Intelligence 11 (6) (1989) 567–585. doi:10.1109/34.24792.

[22] D. S. Broomhead, D. Lowe, Radial basis functions, multivariable functional interpolation and adaptive networks, Tech. rep., Royal Signals and Radar Establishment Malvern (United Kingdom) (1988).

[23] L. E. Dubins, On Curves of Minimal Length with a Constraint on Average Curvature, and with Prescribed Initial and Terminal Positions and Tangents, American Journal of Mathematics 79 (3) (1957) 497–516. doi:10.2307/2372560.

[24] T. G. McGee, S. Spry, J. K. Hedrick, Optimal path planning in a constant wind with a bounded turning rate, in: AIAA Guidance, Navigation, and Control Conference and Exhibit, Reston, VA, 2005, pp. 1–11.

[25] B. L. Golden, L. Levy, R. Vohra, The orienteering problem, Naval Research Logistics (NRL) 34 (3) (1987) 307–318. doi:10.1002/1520-6750(198706)34:3<307::AID-NAV3220340302>3.0.CO;2-D.

[26] I.-M. Chao, B. L. Golden, E. A. Wasil, The team orienteering problem, European Journal of Operational Research 88 (3) (1996) 464–474. doi:10.1016/0377-2217(94)00289-4.

[27] M. G. Kantor, M. B. Rosenwein, The Orienteering Problem with Time Windows, Journal of the Operational Research Society 43 (6) (1992) 629–635. doi:10.1057/jors.1992.88.

[28] F. Tricoire, M. Romauch, K. F. Doerner, R. F. Hartl, Heuristics for the multi-period orienteering problem with multiple time windows, Computers & Operations Research 37 (2) (2010) 351–367.

[29] J. Silberholz, B. Golden, The effective application of a new approach to the generalized orienteering problem, Journal of Heuristics 16 (3) (2010) 393–415. doi:10.1007/s10732-009-9104-8.

[30] P. Vansteenwegen, W. Souffriau, D. V. Oudheusden, The orienteering problem: A survey, European Journal of Operational Research 209 (1) (2011) 1–10. doi:10.1016/j.ejor.2010.03.045.

[31] P. Hansen, N. Mladenović, J. A. Moreno Pérez, Variable neighbourhood search: Methods and applications, Annals of Operations Research 175 (1) (2010) 367–407. doi:10.1007/s10479-009-0657-6.

[32] D. G. Macharet, M. F. M. Campos, An Orientation Assignment Heuristic to the Dubins Traveling Salesman Problem, in: Advances in Artificial Intelligence – IBERAMIA 2014, Lecture Notes in Computer Science, Springer International Publishing, 2014, pp. 457–468.

[33] A. Bit-Monnot, R. Bailon-Ruiz, S. Lacroix, A local search approach to observation planning with multiple uavs, in: Twenty-Eighth International Conference on Automated Planning and Scheduling, 2018.

[34] J. Pinto, P. S. Dias, R. Martins, J. Fortuna, E. Marques, J. Sousa, The LSTS toolchain for networked vehicle systems, in: OCEANS'13 MTS/IEEE Bergen, IEEE, Bergen, Norway, 2013, pp. 1–9. doi:10.1109/OCEANS-Bergen.2013.6608148.

[35] F. López Peña, P. Caamaño, G. Varela, F. Orjales, A. Deibe, Setting up a mixed reality simulator for using teams of autonomous UAVs in air pollution monitoring, International Journal of Sustainable Development and Planning 11 (4) (2016) 616–

626. doi:10.2495/SDP-V11-N4-616-626.

[36] G. Echeverria, S. Lemaignan, A. Degroote, S. Lacroix, M. Karg, P. Koch, C. Lesire, S. Stinckwich, Simulating Complex Robotic Scenarios with MORSE, in: Simulation, Modeling, and Programming for Autonomous Robots, Vol. 7628, Springer Berlin Heidelberg, Berlin, Heidelberg, 2012, pp. 197–208. doi:10.1007/978-3-642-34327-8_20.

[37] H. Chitsaz, S. M. LaValle, Time-optimal paths for a Dubins airplane, in: 2007 46th IEEE Conference on Decision and Control, 2007, pp. 2379–2384. doi:10.1109/CDC.2007.4434966.

[38] L. D. Filippis, G. Guglieri, F. Quagliotti, Path Planning Strategies for UAVS in 3D Environments, Journal of Intelligent & Robotic Systems 65 (1-4) (2012) 247–264. doi:10.1007/s10846-011-9568-2.

[39] P. Váňa, J. Faigl, J. Sláma, R. Pěnička, Data collection planning with Dubins airplane model and limited travel budget, in: 2017 European Conference on Mobile Robots (ECMR), 2017, pp. 1–6. doi:10.1109/ECMR.2017.8098715.