# Surveillance system using cheap sensors and drones

José Coelho, Daniel Silvestre, Rita Cunha
*Laboratory of Robotics and Systems in Engineering and Science (LARSyS)*
*ISR/IST, University of Lisbon,*
Lisbon, Portugal
josescoelho@tecnico.ulisboa.pt

***Abstract*—This work addresses the problem of creating a real time surveillance system to provide early detection warning related to forest fires that is based on an uncertainty map. The dissertation begins by motivating the reader regarding the need to monitor natural events that can have catastrophic consequences for daily life. Following a logic of prevention and timely action, the creation of a real-time surveillance system for wildfires is proposed, aiming to maximize the useful information collected by an autonomous aerial vehicle.**

**In a first chapter, we tackle the problem of optimizing non-convex functions by proposing a novel hybrid iterative algorithm that is able to adapt its behavior to have fast convergence to a neighborhood of a local solution and reduced oscillation around the maximizer. The algorithm is extensively tested with results illustrating its ability to converge to a local maximum with a faster rate than state-of-the-art methods present in the literature.**

**The proposal of this thesis is to use an algorithm to generate waypoints and address the problem of surveillance under the mild assumption of an aerial vehicle capable of taking local measurements with onboard sensors. Modeling the uncertainty map with Gaussian mixtures allows for a general solution that can cope with any type of utility function. Resorting to the optimization algorithm proposed herein to generate the waypoints, it is then generated a smooth path using B-spline. In order to create a control law for tracking this path, a path following algorithm is also studied, which should function as an outer loop for the vehicle in question. The performance of the proposed solution is evaluated using the simulation software Gazebo which incorporates the dynamics of the drone, demonstrating the ability of the proposed solution to guide the vehicle through surveillance areas of high uncertainty.**

***Index Terms*—Wildfire surveillance; Hybrid optimization algorithms; Gaussian-mixture; B-spline; Path following**

## I. Introduction

During the last decades, the effects of wildfires have been a major problem all over the world. According to an European Commission's publication [1, *Super Case Study 4*], just in 2017, Portugal reported 21 000 wildfires, resulting in 539 920 ha of burned area (almost 6% of the total area of Portugal), claiming 117 human lives, including firefighters. The damages were estimated at approximately 1.5 billion euros. These events cause huge losses to populations, either directly, due to the destruction of agricultural resources and private properties, or indirectly due to effects on public infrastructures such as energy networks, roads, and telecommunications.

In Portugal, the wildfire dynamics follow a critical concentration of multiple events in a short period of time [1]: almost two-thirds of the burned area in Portugal in 2016 is the result of fires that occurred in the space of only 10 days. This fact raises two important aspects: $i$) the solution to this problem must follow a philosophy of prevention, lowering the probability of having a critical concentration of events; $ii$) when prevention measures are not sufficient, early detection increases the odds of having an efficient extinction of wildfires.

## II. Envisioned Surveillance Architecture

This work aims to develop an algorithm that can generate an optimal trajectory for an unmanned aerial vehicle (UAV) in real-time based on a dynamic uncertainty map. This map represents the relevance of a particular position at a given time for drone inspection to detect wildfire ignition early.

The proposed algorithm can be viewed as an overlay layer that complements any pre-existing surveillance system. For the purpose of this document, it is assumed that a previously developed surveillance system using an *a priori* map is responsible for maintaining the uncertainty map.

There are two possible sources of uncertainty that shape the uncertainty map, namely:

- Noisy measurements and technological constraints can lead to inaccuracies in the data collected by the surveillance system;
- The uncertain dynamics that govern how the risk in each area involves in-between the update of the map using the availability of fuel, temperature, wind, etc.

Due to the inability of having an exact evolution of the map, the current thesis proposes the use of UAVs equipped with sensors, such as cameras or smoke detectors that can inspect high-risk zones that would be inaccessible using patrols on the ground. Such a vehicle has to have a control algorithm to generate a trajectory through the areas with the highest uncertainty levels, enabling the onboard sensors to infer the existence of a fire. This process adds new information to update the uncertainty map. The framework is illustrated in Fig. 1, where the drone flight creates a closed feedback surveillance system that continuously gathers new information. Our work will primarily focus on the green boxes. We will assume the availability of an uncertainty map and develop a solution to analyze this map. The goal is to propose an optimal trajectory for the aerial vehicle to follow. Additionally, we will study a path-following algorithm to enhance the vehicle's ability to accurately track the desired path.

The proposed framework can have a significant and positive impact on firefighting operations. By reducing uncertainty

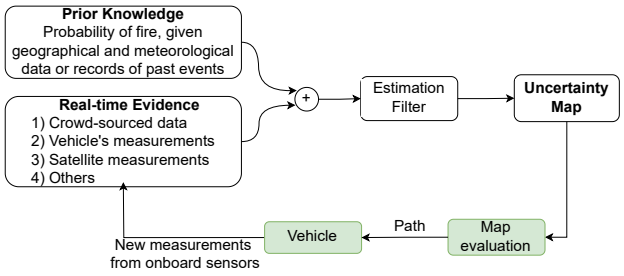


Fig. 1. Architecture of the Envisioned Surveillance. The green boxes represents the main contributions of this thesis

levels in a given area, the system can help coordinate firefighting authorities from an operational perspective, allowing resources to be concentrated in strategic positions. With more accurate and timely information firefighting teams can more effectively allocate resources to contain and extinguish fires. Early detection of wildfires also increases the likelihood of successful firefighting operations and minimizes damage to property and the environment.

### A. Mathematical Definition

Assuming that the estimation filter provides a new uncertainty map to the path planner at a given frequency rate, the map received at time $k$ by the planner can be viewed as a function $h_k(x) : \Re^2 \rightarrow \Re$, where $x \in \mathcal{X}$ where $\mathcal{X}$ represents the surveilled space, and $k \in [0, 1, ..., K-1]$ represents the number of maps received until the instant $k$.

Using $h_k(x)$, the path planner generates a discrete trajectory $\varphi^{[k]} \in \Re^{2\times N}$, composed of $N$ discrete waypoints $[\varphi_1^{[k]}, ..., \varphi_N^{[k]}] \subset \mathcal{X}$. We also pose the assumption that at each position $x \in \mathcal{X}$, a drone possesses a circular measurement area centered at $x$ with radius $r$, denoted by $C(x, r)$.

An optimal trajectory obtained from $K$ sequential maps can be represented as a set of waypoints $\varphi \in \Re^{2\times K\cdot N}$ in the form

$$\varphi = [\varphi_1^{[0]}, ..., \varphi_N^{[0]}, ..., \varphi_1^{[k]}, ..., \varphi_N^{[k]}, ..., \varphi_N^{K-1}]. \tag{1}$$

This trajectory can be obtained by solving the optimization problem presented in Eq. 2. In this equation, the drone is modeled as a non-linear system $f(s)$, with state $s$ and actuated by a control signal $u$. The function $g(s, u)$ represents the constraints imposed on the state and/or control signal of the drone.

$$\begin{aligned} \underset{\varphi}{\text{maximize}} \quad & \sum_{k=0}^{K-1} \int_{\gamma(\varphi^{[k]})} h_k(x)dx, \\ \text{subject to} \quad & \dot{s} = f(s) \\ & g(s, u) \leq 0 \\ & \varphi_n^{[k]} \in \mathcal{X} \end{aligned} \tag{2}$$

where

$$\gamma(\varphi^{[0]}) = \bigcup_{n=1}^{N} C(\varphi_n^{[0]}, r)$$

The function $\gamma(\varphi^k) : \Re^{2\times N} \rightarrow \Re^2$ is essential in preventing the drone from stopping at a local uncertainty maximizer. Refer to Fig. 2 and Fig 3 to understand how the integration area is acquired using this union of neighborhoods. This function ensures that overlapping regions are only considered once during the uncertainty integration process for each new map.

However, it is possible that the drone may not be able to reduce the uncertainty of a location during its first visit. To address this, the neighborhood's union, $\gamma(\varphi^k)$, has to be reset whenever a new map is received. This allows for repeated integration of a position across different maps while still avoiding the risk of getting stuck in local maximizers of the current map.

The proposed formulation enables the planner to compute an optimal trajectory on-the-fly using only the most recently received map. To better understand this on-the-fly property, we can decompose the original problem from Eq. 2 into $K$ separate sub-problems:

*Sub-problem 0 up to K-1 - change from problem $k-1$ to problem $k$ when a new map arrives. Here $k$ starts in 1 and is a constant for each problem:*

$$\begin{aligned} \underset{\varphi^{[k]}}{\text{maximize}} \quad & \int_{\gamma(\varphi^{[k]})} h_k(x)dx, \\ \text{subject to} \quad & \dot{s} = f(s) \\ & g(s, u) \leq 0 \\ & \varphi_n^{[k]} \in \mathcal{X}, \ \ \forall n \in \{1, 2, ..., N\} \\ & \varphi_1^{[k]} = \varphi_N^{[k-1]} \end{aligned} \tag{3}$$

The blue constrain is added to guarantee concordance between problems. Therefore, for $K = 0$, this constrain must not be considered.

Despite the separation of the problem into $K$ sub-problems, each sub-problem still requires solving a non-convex optimization function with non-convex constraints. Additionally, the cost function involves an integral computation that does not have a closed-form solution in the general case. Therefore, the computation of this optimal trajectory may require significant processing power and time resources.

In conclusion, a potential solution to address this problem must be able to compute close to optimal trajectories for each of the received maps since this is a vital feature for the envisioned application of wildfire detection. Fig. 2 and Fig 3 shows an example trajectory for a generic map $k$ with $N = 4$.

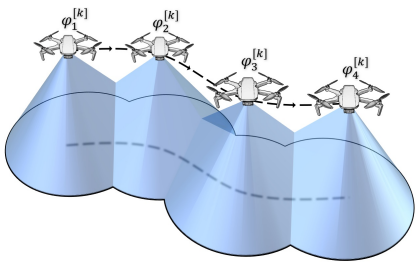


Fig. 2. 3 dimensional view of a path for a generic map $k$ and $N = 4$

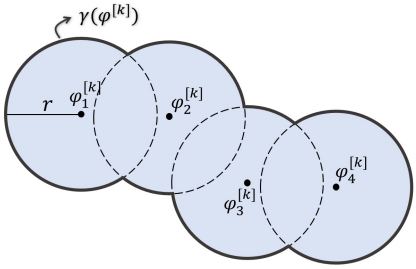


Fig. 3. Top view of a path for a generic map $k$ and $N = 4$

## III. Non Convex Function Optimization

The primary objective of this section is to assess the efficacy of different algorithms in locating local maxima of non-convex functions. Such an algorithm is the primary building block required for the development of a solution to the described problem.

In this section, we are focusing on addressing unconstrained problems of the form:

$$\underset{x}{\text{minimize}} \quad f(x) \tag{4}$$

where $x$ stands for the decision variable belogingng to the set $\mathbb{R}^n$. The objective function $f(x)$ reflects the specific goals or criteria associated with the problem.

### A. Background Theory

In most practical cases, solving optimization problems analytically is not feasible. Even using computational tools to help with algebraic manipulation may be too burdensome. Consequently, the most valuable approaches are numerical algorithms of first and second order. A typical format of first order algorithms is

$$x_{k+1} = x_k + \alpha_k d_k. \tag{5}$$

where $x_k$ denotes the current estimate, $\alpha_k$ is the step size and $d_k$ the search direction. The difference between each first-order algorithm is the philosophy behind the choice of the search direction $d_k$ and the step size $\alpha_k$.

*Definition 1:* $d_k$ is a valid search direction to minimize a function $f(x)$, if it respects

$$d_k^T \nabla f(x) < 0 \tag{6}$$

such that there always exists a $\bar{\alpha} > 0$ that makes

$$f(x_k + \alpha d_k) < f(x_k) \quad \text{for all } 0 < \alpha \leq \bar{\alpha}. \tag{7}$$

A common and computationally efficient approach to define the parameter $\alpha_k$ is to choose a constant step size and let the gradient norm determine both the direction and the magnitude of the estimate's movement. This fixed value may depend on a compromise for the convergence rate over all possible initializations.

The most popular first-order algorithms are:

- Gradient Descent:

$$x_{k+1} = x_k - \alpha \nabla f(x_k) \tag{8}$$

- Heavy Ball:

$$x_{k+1} = \widetilde{x}_k - \alpha \nabla f(x_k), \tag{9}$$

where $\widetilde{x}_k = x_k + \beta(x_k - x_{k-1})$

- Nesterov's

$$x_{k+1} = \widetilde{x}_k - \alpha \nabla f(\overline{x}_k), \tag{10}$$

where $\overline{x}_k = x_k + \beta(x_k - x_{k-1})$

Other algorithms have been suggested in the literature, such as the Triple Momentum approach proposed in [2], where the authors extend the last two momenta Nesterov's approach to the overall estimation:

$$x_{k+1} = x_{k+1}^N + \delta(x_{k+1}^N - x_k^N), \tag{11}$$

where $x_k^N$ is the Nesterov's estimate.

An alternative to using momentum terms in algorithms is to employ adaptive step approaches. Almeida & Silva et al. [3] propose updating the step size individually for each dimension based on the gradient signal in that dimension. This approach can be summarized as follows:

$$\alpha^k = \begin{cases} u \cdot \alpha^{k-1} & \text{if } \frac{df}{dx}(x_{k-1}) \cdot \frac{df}{dx}(x_k) > 0 \\ d \cdot \alpha^{k-1} & \text{otherwise} \end{cases}, \tag{12}$$

where $0 < u < 1$ and $d > 1$ are respectively the "up" and "down" update constants.

In this work, we will be using a normalized version of the gradient when applying the adaptive step algorithm. Normalizing the gradient ensures that the step size is independent of the magnitude of the gradient, making the algorithm more robust and less sensitive to changes in the objective function's scale.

To avoid manual tuning of parameters, the authors of [4] and [2] define a procedure to find optimal parameters for the gradient methods, in the sense that guarantee minimal worst-case convergence rate $\rho_{worst}$, as in defined in the following definition:

*Definition 2:* A sequence $x_k$ is converging with convergence rate $0 < \rho < 1$ if there exists a positive constant $C > 0$ such that

$$||x_{k+1} - x^*|| \leq C\rho^k ||x_0 - x^*|| \tag{13}$$

These parameters, present in Tab I, are obtained from a set of strongly convex functions with $m$-strongly convexity and $L$-Lipschitz continuous gradient for a given $0 < m < L$. In other words, the optimal parameters are computed for functions $f : \Re^n \to \Re$ such that $\forall x, y \in \Re^n$:

$$m||x-y||^2 \leq (\nabla f(x) - \nabla f(y))^T (x-y) \leq L||x-y||^2. \tag{14}$$

After considering momentum terms, adaptive steps, and optimal parameter tuning, another approach is the use of hybrid algorithms capable of adapting their behavior throughout the iterations. In [5], the authors propose the use of a hybrid algorithm for optimizing convex functions, which aims to

TABLE I
OPTIMAL PARAMETERS FOR EACH METHOD AND THEIR CORRESPONDING WORST-CASE CONVERGENCE RATES. $\kappa = L/m$ REPRESENTS THE CONDITION NUMBER.

| Method | Optimal Parameters | Convergence Rate, $\rho_{\max}$ |
|---|---|---|
| Gradient Descent | $\alpha = \frac{2}{L+m}$ | $\frac{\kappa-1}{\kappa+1}$ |
| Nesterov's | $\alpha = \frac{4}{3L+m}$<br>$\beta = \frac{\sqrt{\kappa+1}-2}{\sqrt{3\kappa+1}+2}$ | $1 - \frac{2}{\sqrt{3\kappa+1}}$ |
| Heavy Ball | $\alpha = \frac{4}{(\sqrt{L}+\sqrt{m})^2}$<br>$\beta = \left(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\right)^2$ | $\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}$ |
| Triple Momentum | $\alpha = \frac{2-1/\sqrt{\kappa}}{L}$<br>$\beta = \frac{(1-1/\sqrt{\kappa})^2}{1+1/\sqrt{\kappa}}$<br>$\gamma = \frac{(1-1/\sqrt{\kappa})^2}{(2-1/\sqrt{\kappa})(1+1/\sqrt{\kappa})}$<br>$\delta = \frac{(1-1/\sqrt{\kappa})^2}{1-(1-1/\sqrt{\kappa})^2}$ | $1 - \frac{1}{\sqrt{\kappa}}$ |

achieve fast convergence, reduced oscillations, and robustness. The algorithm utilizes two versions of a heavy ball method, each characterized by a specific set of parameters: $(\alpha_1, \beta_1)$ and $(\alpha_2, \beta_2)$. The algorithm can be defined as follows:

$$x_{k+1} = x_k - \alpha_{\sigma(k)} \nabla f(x_k) + \beta_{\sigma(k)}(x_k - x_{k-1}). \tag{15}$$

The function $\sigma(k) \in \{1, 2\}$ is updated in every iteration and takes values from the set $\{1, 2\}$ based on a supervisor.

### B. Problem Statement

Given that the objective function for the wild fire detection is going to be the result of an estimation process, related to different characteristics like the weather, terrain, etc., the cost function $h(x) : \mathbb{R}^2 \to \mathbb{R}$ will be nonconvex and we pose the assumption of being represented as a Gaussian mixture model. In this section, we remove the constraints and integrals from the previous optimization problem, and solve the problem:

$$\underset{x}{\text{maximize}} \quad h(x). \tag{16}$$

A GM model consists of a weighted sum of $K$ Gaussian distributions, where the weights $w_k$ indicates the contribution of each Gaussian to the overall function, i.e.,

$$h(x) = \sum_{k=1}^{K} w_k \mathcal{N}_k(x), \tag{17}$$

where $\sum_{k=1}^{K} w_k = 1$, and $\mathcal{N}_k(x)$ represents the Gaussian probability density function for component $k$ with mean $\mu_k$ and covariance $\Sigma_k$:

$$\mathcal{N}_k(x) = \frac{1}{2\pi\sqrt{||\Sigma_k||}} e^{-1/2(x-\mu_k)^T \Sigma_k^{-1}(x-\mu_k)}.$$

In order to gather intuition related to the use of Gaussian Mixture models, we present an illustrative example that will guide the design in the remaining of this section. T We start by comparing three groups of algorithms: gradient descent ("GD"), momentum algorithms ("HB", "N", and "TM"), and gradient descent with adaptive step ("Adaptive"). These algorithms will be evaluated using fixed parameters obtained through manual tuning using several runs to improve their values.

The experiments is conducted using a Gaussian mixture with K = 1, $\mu = \begin{bmatrix} 0 & 0 \end{bmatrix}^T$ and $\Sigma = \begin{bmatrix} 1 & 0 \\ 0 & 5 \end{bmatrix}$, as shown in Fig. 4. Two initial points are tested for each algorithm: $x^0 = \begin{bmatrix} -1.5 & 2 \end{bmatrix}^T$ and $x^0 = \begin{bmatrix} -5 & -5 \end{bmatrix}^T$. The first initial point was chosen to represent the behavior of each algorithm when initialized near a maximum, where the gradient is meaningful whereas the second point starts far away in a plateau region. The stopping criteria was defined as the $k$ value such that

$$||x^{k-1} - x^k|| \leq 10^{-4}. \tag{18}$$

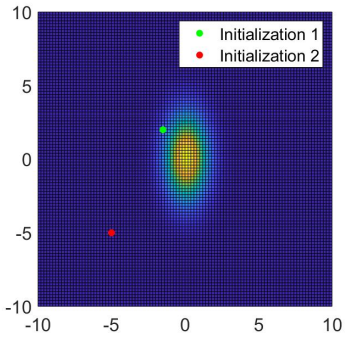


Fig. 4. Top view of the objective function and tested initialization

For the first initial point, $x^0 = \begin{bmatrix} -1.5 & 2 \end{bmatrix}^T$, the performance can be seen in Fig. 5. All algorithms were able to find the maximum of the Gaussian function and converge to it. However, the gradient descent algorithm and its momentum versions showed better overall performance, reaching the stopping criteria after around 40 iterations.

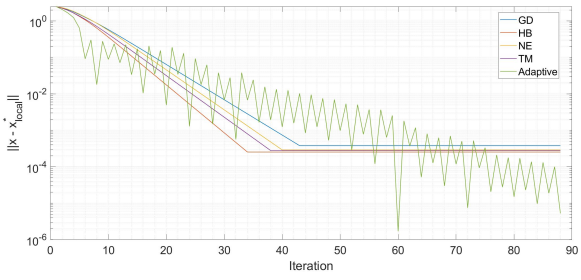


Fig. 5. Distance to a local maximum in each iteration using initialization point 1

For the second initial point, $x^0 = \begin{bmatrix} -5 & -5 \end{bmatrix}^T$, the performance can be seen in Fig. 6. This example highlights the challenge regarding plateau regions. Plateau regions are characterized by the magnitude of gradients almost negligible,

where the algorithms need high $\alpha$ parameters to be able to move their estimate in such a low gradient magnitude area. However, given these high-value parameters, a problem may arise when the estimate gets near the non-plateau regions. There, the algorithm may become unstable using too large steps for the magnitude of the gradient in this region. The adaptive step algorithm is an exception to this problem because it uses normalized gradients, making the algorithm robust to variations in the gradient's magnitude.

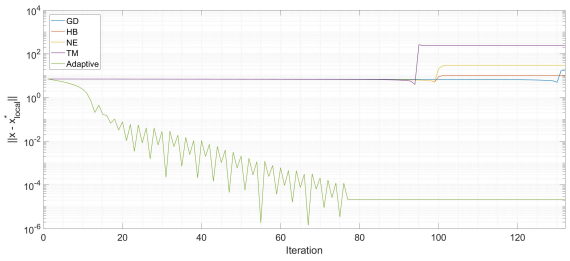


Fig. 6. Distance to a local maximum in each iteration using initialization point 2

In conclusion, Gaussian mixture functions pose a significant challenge for optimization algorithms due to the presence of plateau regions in the search space. Even when algorithm parameters are tuned to address these regions, there is no guarantee of good performance in regions called non-plateau regions, near the maximum, where the gradient increases. This example highlights the need to employ an algorithm that can adapt its behavior while optimizing a Gaussian mixture objective function.

### C. Proposed Solution

Taking into account the challenges identified during the illustrative result, we propose a hybrid algorithm capable of effectively converging to local maximums of Gaussian mixture functions, even when the functions contain multiple Gaussians and the initial estimate resides in a plateau region.

The hybrid algorithm aims to combine the strengths of two optimization algorithms: a global strategy and a local strategy. The global strategy is designed to show fast convergence in plateau regions, while the local strategy focuses on reducing oscillations in non-plateau regions. To enable this hybrid approach, the algorithm relies on information about the $L$-Lipschitz and $m$ strongly convex constants, which are assumed to be supplied by an oracle. The relevance of knowing these constants is related to the usage of the optimal parameters from Tab. I. This way, the parameters of each gradient descent algorithm can be automatically tuned, minimizing the need for human intervention and increasing the adaptability to different optimization environments.

We start by suggesting the design of a specific Gaussian mixture oracle. However, our hybrid algorithm is pretended to work with any oracle capable of providing such constants.

*1) Gaussian mixture Oracle design:* We have developed an oracle for estimating the $L$-Lipschitz constant of a given Gaussian mixture. Additionally, it also estimates the $m$-convexity constant of the non-plateau regions around the local maxima. For that, without loss of generality, we go through the design assuming diagonal covariance matrices in the form $\Sigma_k = \begin{bmatrix} \sigma_1^2 & 0 \\ 0 & \sigma_2^2 \end{bmatrix}$.

In doing so, the Gaussian probability density on Eq.17 turns into

$$\mathcal{N}_k(x) = \frac{1}{2\pi\sigma_1\sigma_2} e^{\left(\frac{-(x_1-\mu_1)}{2\sigma_1} + \frac{-(x_2-\mu_2)}{2\sigma_2}\right)}. \tag{19}$$

Our oracle employs a second-order Taylor approximation for each Gaussian in the Gaussian mixture function. Taking into account some vanishing terms due the null gradient at $x = \mu$, and due the assumed diagonal covariance matrix, each expansion is centered in a Gaussian mean as:

$$\begin{aligned} \mathcal{N}(x) \approx \mathcal{N}(\mu) + \frac{(x_1-\mu_1)^2}{2}\frac{\partial^2\mathcal{N}}{\partial x_1^2}(\mu) \\ + \frac{(x_2-\mu_2)^2}{2}\frac{\partial^2\mathcal{N}}{\partial x_2^2}(\mu). \end{aligned} \tag{20}$$

This results in a function that reasonably approximates the Gaussian distribution around its mean point, as shown in Fig. 7 for a one dimensional Gaussian function.

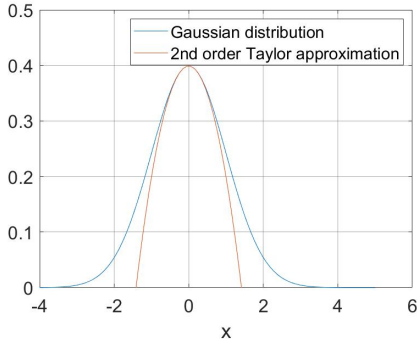


Fig. 7. Example of a one dimensional Gaussian function and the respective second-order Taylor's approximations

At this point, our oracle applies the definition of the constants $L$ and $m$, as given in Eq. 14 to each quadratic function obtained from the second-order Taylor's expansions. Specifically, for 2-dimensional quadratic functions, the definition implies that $L$ and $m$ correspond, respectively, to the eigenvalues of maximum and minimum absolute value of the Hessian matrix:

$$H = \frac{1}{2\pi\sigma_1\sigma_2}\begin{bmatrix} 1/\sigma_1^2 & 0 \\ 0 & 1/\sigma_2^2 \end{bmatrix}, \tag{21}$$

associated with each quadratic term. Since Eq. 21 is a $2 \times 2$ diagonal matrix, its eigenvalues can be obtained directly from the non-zero entries of the matrix.

From the set of $K$ pairs $(L_k, m_k)$, we now can choose the largest $L_k$ and the smallest $m_k$ as the output of our oracle.

*2) Hybrid Algorithm:* The intuitive idea behind the algorithm is to divide its behavior based on the proximity to a maximum or to a plateau region. At each iteration, this decision is made using a finite differences second-order gradient of the objective function, computed for each dimension. It is worth noting that the algorithm does not compute the full Hessian matrix, but only the second-order derivatives with respect to each dimension. For the current iteration $x^k = \begin{bmatrix} x_1^k & x_2^k \end{bmatrix}^T$, the second-order derivative approximations, $\widetilde{\nabla}^{2nd}(x^k) = \begin{bmatrix} \widetilde{\nabla}_1^{k^{2nd}} & \widetilde{\nabla}_2^{k^{2nd}} \end{bmatrix}^T$ are given by:

$$\widetilde{\nabla}_1^{k^{2nd}} = \frac{f(x_1^k + \delta x) - 2f(x_1^k) + f(x_1^k - \delta_x)}{\delta_x^2}, \tag{22}$$

$$\widetilde{\nabla}_2^{k^{2nd}} = \frac{f(x_2^k + \delta_x) - 2f(x_2^k) + f(x_2^k - \delta_x)}{\delta_x^2}, \tag{23}$$

where $\delta_x$ represents the step size for the finite difference approximation.

Our oracle is able to characterize reasonably well the non-plateau regions. Furthermore, the $L$-Lipschitz and $m$-convexity constant of a function can be used as the upper and lower bounds of the eigenvalues of its Hessian matrix, which are related to the second-order derivatives of the function. Thus, the hybrid algorithm compares the minimum value of the finite differences second-order derivative approximation with the $m$ constant obtained from our oracle. This comparison is based on the fact that the $m$-convexity condition implies that the curvature of non-plateau regions is greater than or equal to $m$, which in turn is greater than the curvature of plateau regions. This condition enables the algorithm to use the second-order derivative approximation to determine whether the current estimation is in a plateau or non-plateau region.

The subroutine responsible for performing these computations serves as a supervisor that triggers a global algorithm (designed for fast convergence) or a local algorithm (tailored to avoid oscillations). Fig. 8 formalizes the state machine used.

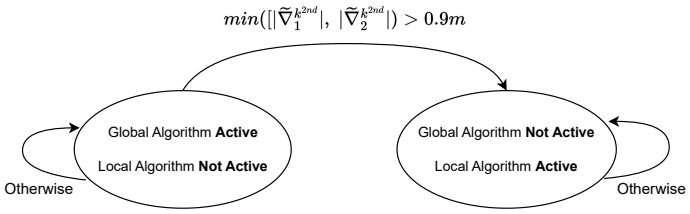


Fig. 8. Hybrid algorithm supervisor decision

### D. Hybrid Algorithm Results

This section focuses on evaluating the performance of our hybrid algorithm using a range of local and global methods against the adaptive step by measuring the distance to the closest local maximum of the Gaussian mixture function. The following options were considered:

- Adaptive step as the global algorithm and gradient descent with parameters defined using the oracle output and Tab. I as the local algorithm - Identified as "A+GD".
- Adaptive step as the global algorithm and Heavy Ball with parameters defined using the oracle output and Tab. I as the local algorithm - Identified as "A+HB".
- Adaptive step as the global algorithm and Nesterov's with parameters defined using the oracle output and Tab. I as the local algorithm - Identified as "A+NE".
- Adaptive step as the global algorithm and Triple Momentum algorithm with parameters defined using the oracle output and Tab. I as the local algorithm - Identified as "A+TM".

*1) Gaussian mixture with $K = 6$ near Gaussians:* We evaluate the performance of each algorithm using a Gaussian mixture function with 6 overlapping Gaussians. We use this function to assess whether our local algorithms can converge to local maxima when the individual Gaussians overlap. This test is interesting because the oracle computes the $L$ and $m$ constants using the individual Gaussians without considering their proximity.

The test consists on running each algorithm using 50 random initializations, as shown in Fig. 9. Each algorithm was run for every initialization, and we recorded the number of iterations required to meet the stopping criteria for each initialization. A cumulative average is reported in Fig. 10. In addition to successfully converging to a local maximum in each initialization, the results indicate that the gradient descent algorithm and its momentum versions outperform the adaptive algorithm. Among the gradient descent algorithms, the momentum versions also outperform the gradient descent.

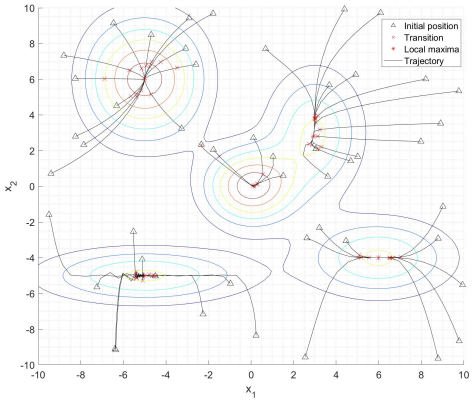


Fig. 9. Test 3 - Initial estimate and hybrid algorithm with gradient descent trajectories for each one of the 50 random initializations

## IV. UAV Path Planning and Path Following

In this section, we explore an approach for solving the problem of optimal trajectory generation from multiple uncertainty maps, as discussed in the introductory section. We will focus on a specific case of that original problem. This

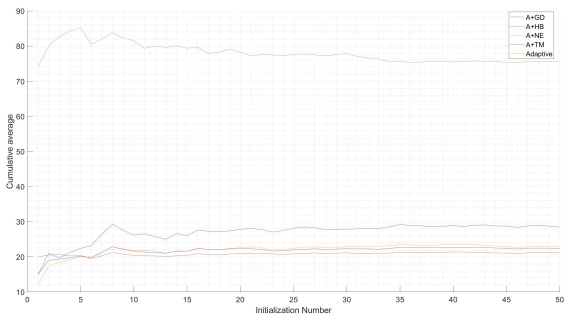


Fig. 10. Test 3 - Cumulative average of iterations required by each algorithm to meet the stopping criteria

approach allows us to consider certain simplifications while still achieving effective solutions. We will:

- Narrow our scope to a single map problem, where $K = 1$
- Adopt an empirical approach to determine the set of waypoints $\varphi$ that will be utilized by the path planner, instead of directly addressing the sub-problem outlined in Equation 3
- Consider that the uncertainty map $h(x)$ is modeled as a discrete Gaussian mixture function. Furthermore, we assume that a previously developed surveillance system is responsible for maintaining the uncertainty map.

Regarding the second consideration, our decision is driven by a trade-off between accuracy and complexity. Solving the original problem using integral calculations and area unions would be computationally expensive. Instead, by leveraging our intuition regarding the map's structure we aim to find an effective solution to the problem at hand.

### A. Proposed Solution

Our proposed algorithm is based on the observation that the uncertainty map is likely to contain multiple local maxima, which correspond to areas of higher uncertainty where the quadrotor needs to conduct detailed surveillance. Intuitively, the vehicle should navigate towards one of these local maxima, allowing it to gather new measurements and reduce uncertainty in those specific areas. Hence, each local maximum will serve as a target location for the quadrotor to reach. Once it arrives at the target location, it carries out a surveillance mission using a spiral path to systematically scan the region.

Fig. 11 provides a visual representation of the entire process.

### B. Waypoint Generation

The generation of waypoints in our system is implemented using a state machine approach, as depicted in Figure 12. The waypoint generation process is performed online, at regular intervals. Whenever a new waypoint is computed, it is sent to the path following ROS package through a ROS publication. The waypoint generation state machine is divided into two main parts.

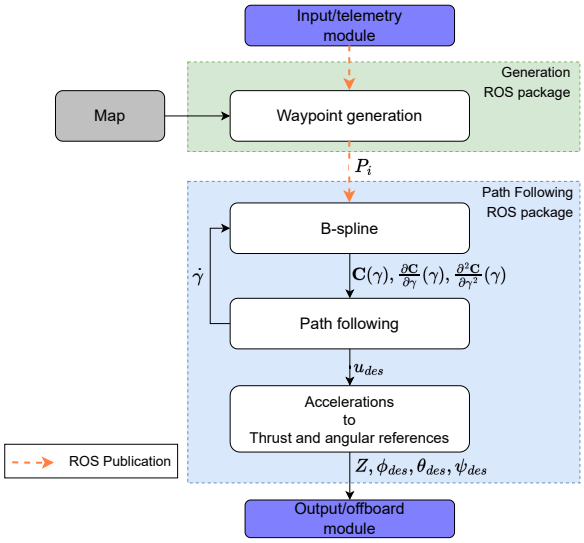


Fig. 11. Diagram of the proposed solution. Each module is further explored in the following section

In the first part, corresponding to states 1 and 2, the state machine employs the hybrid algorithm previously proposed.

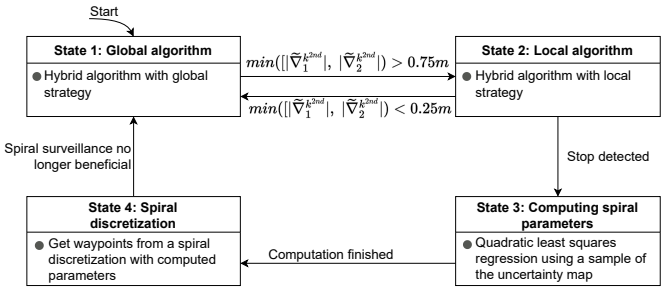


Fig. 12. State machine for waypoint generation

The second part, corresponding to states 3 and 4, is activated when the state machine is in state 2 and successfully converges to a local maximum of the uncertainty map. These states are responsible for defining and discretizing a spiral path that allows the quadrotor to enhance its inspection of the area with high uncertainty using its onboard sensors. Defining $V_{CLV}$ as the magnitude of the first derivative of the spiral curve, and $\Delta r$ as the spiral pitch or sampling pitch along the radial dimension (a measurement for the growth rate of the radius), we define the paremtrization as:

$$\begin{aligned} r_x(\gamma) &= r_y(\gamma) = \gamma, \\ \theta(\gamma) &= \gamma, \end{aligned} \tag{24}$$

where

$$\begin{aligned} r_x(\gamma) = r_y(\gamma) &= \sqrt{\frac{\Delta r V_{CLV}}{\pi}}\gamma, \\ \theta(\gamma) &= \sqrt{\frac{V_{CLV} 4\pi}{\Delta r}}\gamma. \end{aligned} \tag{25}$$

This parameterization allows us to easily generate waypoints that are uniformly distributed along the entire spiral. However, it assumes a circular spiral with $r_x(\gamma) = r_y(\gamma)$, which does not fully meet the second consideration.

To address this limitation we employ a neighbor map based on the current drone position. This neighborhood corresponds to the grid position where the drone is located and the five neighboring grids in each direction, resulting in a total of $(2 \times 5 + 1)^2$ cell values. These values are utilized to conduct a regression using the least squares principle and the quadric model:

$$\hat{h} = \begin{bmatrix} x & y \end{bmatrix} \underbrace{\begin{bmatrix} a & b/2 \\ b/2 & c \end{bmatrix}}_{Q} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} d & e & f \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}. \tag{26}$$

to adapt the spiral radius to the desired shape we can find the major and minor axis of the ellipse defined by

$$\begin{bmatrix} x & y \end{bmatrix} Q \begin{bmatrix} x \\ y \end{bmatrix} = 1 \tag{27}$$

To do so, we compute the eigenvalues, $\lambda_{1,2}$ , and eigenvectors, $v_{1,2}$, of the matrix $Q$, which provides an estimate for the shape of the area around the local maximum of the uncertainty map. Defining the magnitude of the major and minor axis, as $a$ and $b$ and the rotation of major axis as $\alpha$:

$$a = \frac{1}{\sqrt{\lambda_1}}, \qquad b = \frac{1}{\sqrt{\lambda_2}}, \qquad \alpha = \arctan\left(\frac{v_1^{(2)}}{v_1^{(1)}}\right), \tag{28}$$

where $\lambda_1 < \lambda_2$, and $v_i = \begin{bmatrix} v_i^{(1)} & v_i^{(2)} \end{bmatrix}^T$ denotes the eigenvector corresponding to the eigenvalue $\lambda_i$ of the matrix $Q$.

To prevent the magnitude of the standard deviation of a Gaussian from affecting the parameters $a$ and $b$, which are intended to characterize the shape of the Gaussian rather than its magnitude, a normalization step is performed. Specifically, the values of $a$ and $b$ are normalized to satisfy the condition $||a + b|| = 1$.

Using these shaping parameters, and defining $P_{center}$ as the waypoint where the local maximum was detected, the spiral path is parametrized as

$$\mathbf{C}(\gamma) = P_{center} + \begin{bmatrix} \cos(\alpha) & -\sin(\alpha) \\ \sin(\alpha) & \cos(\alpha) \end{bmatrix} \begin{bmatrix} a \cdot r(t)\cos(\theta(\gamma)) \\ b \cdot r(t)\sin(\theta(\gamma)) \end{bmatrix}, \tag{29}$$

### C. B-spline

The path following ROS package receives individual waypoints and accumulates them for use in the B-spline module. This module parameterizes the path to be followed by the vehicle using a 2-dimensional uniform cubic B-spline in the form

$$\mathbf{C}(\gamma) = \frac{1}{6} B_i(\gamma) \begin{bmatrix} P_i \\ P_{i+1} \\ P_{i+2} \\ P_{i+3} \end{bmatrix}, \tag{30}$$

where

$$B_i(\gamma) = \begin{bmatrix} (\gamma - i)^3 \\ (\gamma - i)^2 \\ (\gamma - i) \\ 1 \end{bmatrix}^T \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 0 & 3 & 0 \\ 1 & 4 & 1 & 0 \end{bmatrix}. \tag{31}$$

### D. Path Following

The key idea behind our path following strategy is to drive the vehicle to a virtual target that moves along the desired path defined as a uniform cubic B-spline, denoted as $p_d(\gamma) \in \Re^2$. Fig. 13 illustrates this approach.

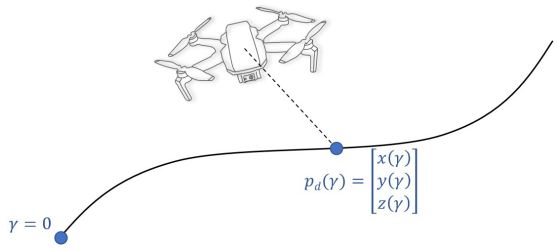


Fig. 13. Virtual target for path following representation

Our path following algorithm aims to generate a set of desired accelerations, denoted as $u_{des} \in \Re^3$. For that, it is designed to achieve two main objectives:

- Convergence of the quadrotor's position $p(t)$ to a tube around the desired position $p_d(\gamma)$.This convergence can be made arbitrarily small, effectively reducing the norm $||p(t) - p_d(\gamma)||$ to a neighborhood of the origin.
- Convergence of the speed of the virtual target to a desired speed profile denoted $v_d(\gamma)$. In other words, ensuring that $|\dot{\gamma} - v_d(\gamma)| \to 0$ as $t \to \infty$. The speed of the virtual target can be seen as the progression rate of the parametrize variable that is moving along the path.

Let us define the position and velocity errors between the vehicle, located at $p(t)$, and the virtual target as follows:

$$e_p \quad := p(t) - p_d(\gamma) \tag{32}$$

$$e_v \quad := \dot{p} - \frac{\partial p_d}{\partial \gamma} v_d(\gamma) \tag{33}$$

Our path following controller is inspired by a Proportional Derivative controller proposed by Jacinto et al. [6], and futher extented to become a Proportional Integral Derivative controller:

$$u_{des} = h(\gamma) v_d(\gamma) - K_p e_p - K_v e_v - K_i \int e_p, \tag{34}$$

where $\int e_p$ represents a discrete integration process of $e_p$, while $K_p$, $K_i$ and $K_v$ are the control gains. The function $h(\gamma)$ is derived from the expression of the path's acceleration when the virtual target converges to the desired speed profile, i.e, when $\dot{\gamma} = v_d(\gamma)$. Therefore, we can obtain the corresponding expression for $h(\gamma)$ as

$$\frac{d^2 p_d}{dt^2}(\gamma) = \underbrace{\left[\frac{\partial^2 p_d}{\partial \gamma^2} v_d(\gamma) + \frac{\partial p_d}{\partial \gamma}\frac{\partial v_d}{\partial \gamma}\right]}_{h(\gamma)} \dot{\gamma} \tag{35}$$

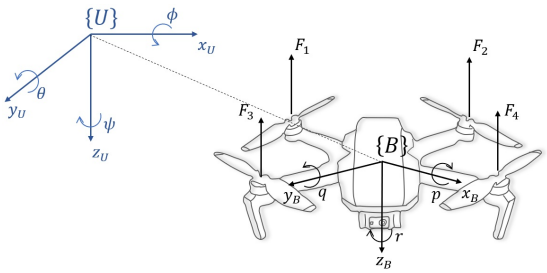


Fig. 14. Adopted reference frames

For controlling the speed of the virtual target $\dot{\gamma}$ we assume that it can precisely follow the desired speed profile, resulting in $\dot{\gamma} = v_d(\gamma)$.

### *E. Acceleration to Thrust and angular references*

Consider two frames used to characterize the dynamics of the vehicle, denoted as follows:

- Frame $\{B\}$: A body-fixed frame rigidly attached to the geometric center of mass of the quadrotor.
- Frame $\{U\}$: An inertial reference frame.

Let's define the following notations:

- $\eta_2 = [\phi, \theta, \psi]^T$: Orientation of frame $\{B\}$ with respect to frame $\{U\}$, expressed in Euler angles.
- $F_{RB} = [X, Y, Z]^T$: External forces measured in frame $\{B\}$.

Consider now that the quadrotor is modeled as a double integrator system given by

$$\ddot{p} \coloneqq \ddot{\eta}_1 = -\frac{Z}{m} {}^U_B R(\eta_2) e_3 + g e_3 \tag{36}$$

Where ${}^U_B R(\eta_2)$ represents the rotation matrix from the body reference frame to the inertial frame and can be defined as

$$\begin{aligned} {}^U_B R(\eta_2) &= R_z(\psi) R_y(\theta) R_x(\phi) \\ &= \begin{bmatrix} c\theta c\psi & s\phi s\theta c\psi - c\phi s\psi & c\phi s\theta c\psi + s\phi s\psi \\ c\theta s\psi & s\phi s\theta s\psi + c\phi c\psi & c\phi s\theta s\psi - s\phi c\psi \\ -s\theta & s\phi c\theta & c\phi c\theta \end{bmatrix} \end{aligned} \tag{37}$$

Where $c$ and $s$ denote the trigonometric functions $cos(.)$ and $sin(.)$ respectively.

We will utilize the inner loop controller provided by the PX4 autopilot, which takes input as a set of angular references $[\phi_{des}, \theta_{des}, \psi_{des}]^T$ and total thrust $Z$. We will consider the yaw angle reference, $\psi_{des}$, as a free variable defined by the outer-loop controller. Specifically, we will set $\psi_{des}$ to a constant value of zero.

Expanding Eq. 36:

$$u_{des} = \ddot{\eta}_1 = -\frac{1}{m} R_z(\psi_{des}) [R_y(\theta) R_x(\phi) Z] e_3 + g e_3 \tag{38}$$

Let's consider an auxiliary variable $u^*$ defined by

$$u^* \coloneqq R_y(\theta) R_x(\phi) Z \tag{39}$$

Once we receive the input $u_{des}$ from the outer-loop, it is possible to compute the value of the auxiliary variable $u^*$ by replacing Eq. 39 in Eq. 38:

$$u^* = -m R_z^T(\psi_{des})(u_{des} - g e_3) \tag{40}$$

The auxiliary variable can be related to the total required thrust by

$$||u^*|| = Z. \tag{41}$$

Finally, to get the desired angular references, we can use the following relation:

$$\frac{u^*}{||u^*||} = \begin{bmatrix} \cos(\phi) \sin(\theta) \\ \sin(\theta) \\ \cos(\phi) \cos(\theta) \end{bmatrix} \tag{42}$$

By solving Equation Eq. 42 with respect to the attitude angles and considering that $u^* = [u_1^*, u_2^*, u_3^*]^T$, we obtain the following expressions:

$$\phi_{des} = \arcsin\left(-\frac{u_2^*}{Z}\right) \tag{43}$$

$$\theta_{des} = \arccos\left(\frac{u_1^*}{u_3^*}\right) \tag{44}$$

## V. Results

A discretized version of the previously introduced map consisting of 6 Gaussians is used to test the overall solution in . In the continuous version of this function, the hybrid algorithm successfully converged to local maximums. Therefore, we expect the same capability in this test. However, it remains crucial to evaluate the performance of the waypoint generation package in terms of the spiral path. The waypoint generation module should be capable of extracting the shape of the neighborhood of each local maximum to infer how to parameterize the spiral and enhance the surveillance task. This particular map has its Gaussians relatively separated from each other, providing an interesting starting point to evaluate this capability.

Fig. 15 depicts two runs of the complete algorithm with different initializations: $(-9, 9)$ and $(9, -9)$. In both runs, it is evident that the waypoint generation algorithm successfully inferred the shape of each spiral, allowing the quadrotor to adjust its flight path to measure the regions with higher uncertainty effectively. Moreover, the generated waypoints, combined with the B-spline module, created a smooth path that the vehicle was able to follow accurately.

Fig. 16 and Fig. 17 illustrate the position tracking error and the speed of the vehicle during the first run of the test, with an initial position at $(-9, 9)$. The vehicle successfully followed the proposed path, with the error between the desired position of the quadrotor and the actual position converging to approximately zero. However, it is visible some occasional peaks in the tracking error of around $0.5$ meters in the more aggressive parts of the trajectory. These peaks are observed at the start of the flight, as well as when the vehicle hovers in a local maximum and begins its spiral trajectory. During these instances, the quadrotor needs to accelerate and overcome

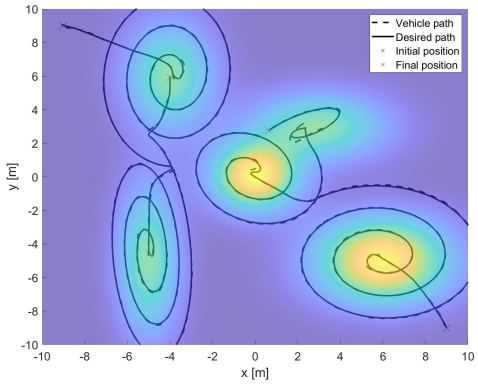


Fig. 15. Top view of the desired and performed path for two initializations

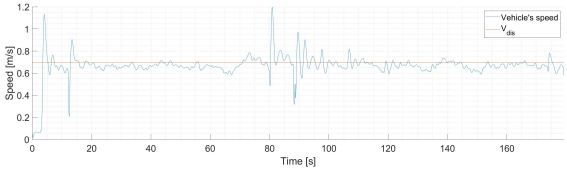


Fig. 16. Position error for the first run with initial location $(-9, 9)$

its dynamics to converge to the virtual target. An example of this behavior can be seen around $t = 90$ seconds when the vehicle is at the local maximum located approximately at $(-5, -5)$. The speed of the quadrotor exhibits a similar pattern. It remains relatively constant at $V_{dis} = 0.7m/s$, showing higher deviations from this value when the tracking error increases.

## VI. Conclusion

In conclusion, this master's thesis has focused on the development of a real-time surveillance system specifically designed for wildfire monitoring. The primary objective was to design a system that utilizes an uncertainty map to identify the most relevant location for drone inspections. Through a comparative analysis of existing surveillance systems, this thesis proposed the idea of adding an overlay layer that can complement other solutions with drones to inspect high-priority zones. The inspection problem is formulated mathematically as a maximization of the uncertainty value within the measured areas along the trajectory followed by the vehicle. Such an approach has the potential to have an early detection of wildfire ignitions.

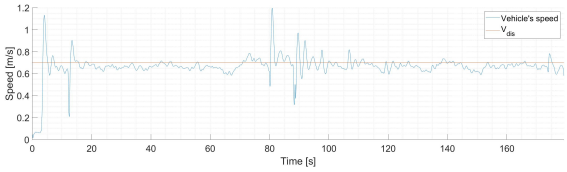


Fig. 17. Speed and desired speed for the first run with initial location $(-9, 9)$

Chapter 2 was dedicated to the creation of a hybrid algorithm capable of converging to local maximums of non-convex functions. By modeling non-convex functions as Gaussian mixtures, it was seen that an alternative algorithm was necessary to address plateau regions with very low gradient magnitudes while avoiding divergence in regions with high gradient magnitudes. Our algorithm was tested and shown to overcome these challenges, converging to a local maximum in every test performed. Furthermore, the presented algorithm is also able to auto tune its parameters based solely in the estimate of two parameters to characterize the objective function.

Chapter 3 presents a real-time solution designed to guide a quadrotor through the most uncertain areas of an uncertainty map. We begin by formulating a trade-off solution between the accuracy and complexity of the original problem. The proposed solution relies on an empirical approach to analyze the received uncertainty map and determine the set of waypoints to be extracted. These waypoints can be generated using either the hybrid algorithm proposed in this study or a spiral parametrization technique that adapts its shape to the non-plateau Gaussian region where the vehicle is located. We explore how to parametrize this adaptable spiral parametrization using a sample from the objective function.

In the subsequent sections, we develop a path following algorithm capable of utilizing the generated waypoints to compute a smooth cubic B-spline path. To follow the desired path, we suggest the use of a PID outer-loop controller capable of computing acceleration references, which are later translated into thrust and angular values.

To evaluate the performance of our approach, we conducted tests in a detailed simulation environment provided by Gazebo. This simulation environment closely emulates real-world flight arenas, ensuring the accuracy and fidelity of our path following algorithm. The results obtained from these tests affirm the success of our approach, as we observed a maximum root mean square error of the tracking position of $0.098$ meters. Furthermore, the trajectory is successfully tailored to address the shape of the uncertainty region in which the drone was conducting surveillance.

## References

[1] Commission and J. R. Centre, Science for disaster risk management 2020: acting today, protecting tomorrow, I. Clark, K. Poljansek, A. Casajus Valles, and M.Martín Ferrer, Eds. Publications Office, 2021

[2] B. V. Scoy, R. A. Freeman, and K. M. Lynch, "The fastest known globally convergent first-order method for minimizing strongly convex functions," IEEE Control Systems Letters, vol. 2, pp. 49–54, 1 2018

[3] F. M. Silva and L. B. Almeida, "Speeding up backpropagation," in Advanced neural computers. Elsevier, 1990, pp. 151–158

[4] L. Lessard, B. Recht, and A. Packard, "Analysis and design of optimization algorithms via integral quadratic constraints," SIAM Journal on Optimization, vol. 26, pp. 57–95, 2016

[5] D. M. Hustig-Schultz and R. G. Sanfelice, "A robust hybrid heavy ball algorithm for optimization with high performance;," American Control Conference (ACC), 2019

[6] M. F. Jacinto, "Cooperative motion control of aerial and marine vehicles for environmental applications," Master's thesis, 2021.