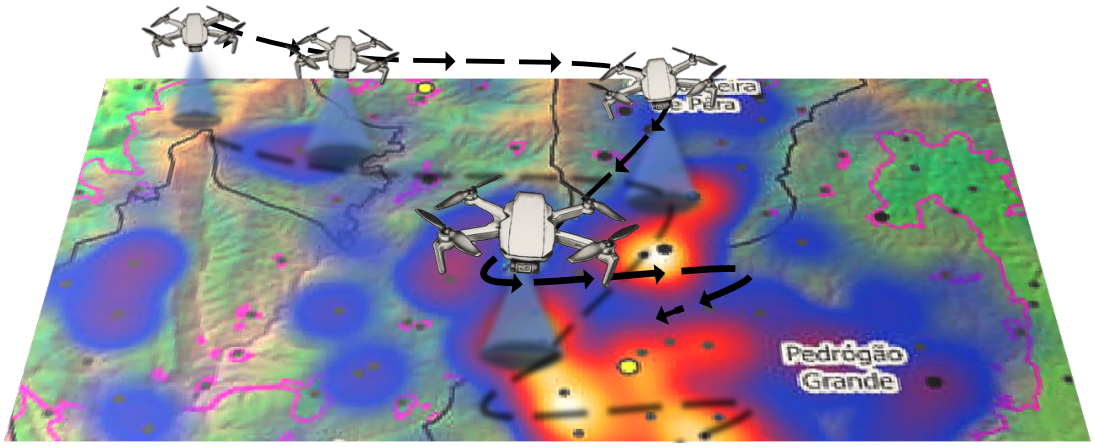
TÉCNICO LISBOA



Surveillance system using cheap sensors and drones

José Paulo Simões Coelho

Thesis to obtain the Master of Science Degree in

Electrical and Computer Engineering

Supervisors: Prof. Rita Maria Mendes de Almeida Correia da Cunha
Prof. Daniel Matos Silvestre

Examination Committee

Chairperson: Prof. João Manuel de Freitas Xavier
Supervisor: Prof. Rita Maria Mendes de Almeida Correia da Cunha
Member of the Committee: Prof. Pedro Daniel Graça Casau

June 2023

I declare that this document is an original work of my own authorship and that it fulfills all the requirements of the Code of Conduct and Good Practices of the Universidade de Lisboa.

Acknowledgments

The scientific method is developed using solid foundations that support its evolution. My journey through this institute was no different. I would like to dedicate this dissertation to all my Foundations, who throughout these 5 years made it possible to overcome all the obstacles and challenges that arose while allowing me to have a unique and irreplaceable experience.

I want to thank my family, the Foundation I call home. Thank you to my father for the continuous support and constant concern that the best version of myself was available for this marathon. Thank you to my mother for the words of support at the right moments. Thank you to my sisters for being my partners in extracurricular life and for being role models. Thank you to Sara for being able to consistently refresh my study playlists on Spotify.

I want to thank my friends, the Foundation I call my second home. Those with whom I had the opportunity to experience the peaks, valleys, and loops that the roller coaster called Técnico offers. They were the ones who made the long hours spent in classrooms and amphitheaters lighter. If there's one thing I want to take with me from these years for the rest of my life, before circuit analysis, propagation and radiation of electromagnetic waves, or differential calculations, it's you. It was you who showed me that the best work arises from a balance between long hours of effort and (more or less regular) breaks for conversations, jokes, complaints, and venting.

Finally, a I want to thank my Supervisor Professor Daniel Silvestre, for the opportunity he gave me to develop this dissertation in an area of my interest and in conjunction with a research scholarship, which provided the opportunity to understand what is done in the world of scientific research. Thank you for all the guidance, explanations, suggestions, and (much-needed) revisions that undoubtedly made this final phase of my studies more fruitful and enjoyable. I would also like to thank Professor Rita Cunha for being available to help whenever necessary and for facilitating contacts and opportunities for important parts of this work to be tested in the flight arena.

This work was partially supported by the Portuguese Fundacão para a Ciência e a Tecnologia (FCT) through Institute for Systems and Robotics (ISR), under Laboratory for Robotics and Engineering Systems (LARSyS) project UIDB/50009/2020, through project PCIF/MPG/0156/2019 FirePuma and through COPELABS, University Lusófona project UIDB/04111/2020

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 at 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.

Keywords

Wildfire surveillance; Hybrid optimization algorithms; Gaussian-mixture; B-spline; Path following

Resumo

Este trabalho aborda a criação de um sistema de vigilância em tempo real para fornecer alertas precoces relacionados com incêndios florestais baseado num mapa de incertezas. A dissertação começa por motivar o leitor em relação à necessidade de monitorizar eventos naturais com consequências catastróficas para o quotidiano. Seguindo uma lógica de prevenção e ação atempada, é proposta a criação de um sistema de vigilância de incêndios florestais, que tem como objetivo a maximização da informação útil recolhida por um veículo aéreo autónomo.

Numa primeira fase, abordamos o problema de otimizar funções não-convexas, propondo um novo algoritmo híbrido iterativo capaz de adaptar o seu comportamento de modo a apresentar convergência rápida para a vizinhança de uma solução local e reduzir oscilações em tordo do maximizante. O algoritmo é extensivamente testado, apresentando resultados que mostram a sua habilidade em convergir para máximos locais com uma taxa superior à de algoritmos de estado-da-arte apresentados na literatura.

A proposta desta tese é a de utilizar um algoritmo capaz de gerar pontos de referência e abordar o problema de vigilância sob o prossuposto de existir um veículo aéreo capaz de obter medições locais recorrendo a sensores a bordo. Modelar o mapa de incerteza como uma mistura de Gaussianas permite recorrer a uma solução geral que pode lidar com qualquer função objetivo. Recorrendo ao algoritmo de otimização proposto nesta dissertação para a geração de pontos de referência, é posteriormente gerado um caminho suave utilizando uma B-spline. De modo a criar uma lei de controlo para o seguimento do caminho, um algoritmo de path following que deverá funcionar como um ”outer loop” para o veículo em causa, é também abordado. A solução proposta é avaliada recorrendo ao software de simulação Gazebo que incorpora de forma pormenorizada a dinâmica do drone, demonstrando a capacidade de

guiar o veículo pelas áreas de vigilâncias de grande incerteza.

Palavras Chave

Vigilância de incêndios; Algoritmos de otimização hibridos; Gaussian-mixture; B-spline; Seguimento de caminhos;

Contents

List of Figures

List of Tables

List of Algorithms

Acronyms

RADNET	Alert Radiological Monitoring Network
CCTV	Closed-Circuit Television
CBRN	Chemical, Biological, Radiological and Nuclear
FWI	Fire Weather Index
IPMA	Instituto Português do Mar e da Atmosfera
GM	Gaussian Mixture
ha	Hectares
LQR	Linear Quadratic Regulator
KKT	Karush-Kuhn-Tucker
ROS	Robot Operating System
UAV	Unmanned Aerial Vehicle
ISR	Institute for Systems and Robotics

1

Introduction

Contents

1.1 Motivation

In nature, some events such as wildfires, radiological threats, or floods are occasional, unpredictable, and cause severe consequences on the normal functioning of populations. In the human quotidian, some activities might need monitoring in a very similar fashion as natural phenomenon. For example, criminal behaviors like robbery or private property trespassing are often unpredictable, reduce safety and disrupt social order. Events with mass gatherings are an example where, although usually anticipated, the high density of people can make it difficult for authorities to detect, locate and help in emergencies. In all such instances, there is a clear benefit of having a robust surveillance system that can accelerate early detection while decreasing possible consequences. At their core, surveillance must be a combination of one or many technologies able to continuously gather data regarding the target in each application.

1.1.1 Wildfires

During the last decades, the effects of wildfires have been a major problem all over the world. According to an European Commission's publication [2, *Super Case Study 4*], just in 2017, Portugal reported 21 000 wildfires, resulting in 539 920 Hectares (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 [2]: 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.

The search for solutions in the scientific community has been a field of interest for governments and authorities. The Portuguese state has been promoting scientific research and innovation to improve the national forest defense system against wildfires, opening calls for research and development projects on topics such as behaviors towards wildfire prevention and fighting, and land management [3].

1.1.2 Chemical, Biological, Radiological and Nuclear threats

Chemical, Biological, Radiological and Nuclear (CBRN) threats, created either by natural processes or human activity, are one of the major challenges of the 21st century. In 2011, a tsunami caused a complete station blackout on the Fukushima Daiichi nuclear power plant in Japan. As a result, hydrogen

explosions were produced, large amounts of radionuclide were released into the atmosphere, and highly radioactive water was directly drained into the sea [2, *Super Case Study 2*]. The amount of released radioactive products and the area of contamination in CBRN disasters need to be continuously analyzed in the following months or years to evaluate the need of evacuating populations and to further understand the effect of radiation on both humans on non-human species. Such a scenario poses threats to the health of humans when the operations are performed by people. Therefore, an autonomous surveillance system would also benefit in this type of accidents.

Also on this topic, governments and authorities have been searching for solutions in the scientific community. The European Union launched the *European Atomic Energy Community (EURATOM) Treaty* which declares that "each Member State shall establish the facilities necessary to carry out continuous monitoring of the level of radioactivity in the air, water, and soil and to ensure compliance with the basic standards." [4, Article 35]. Portugal has a network of sensors capable of detecting an abnormal increase of radioactive elements in the environment, called Alert Radiological Monitoring Network (RADNET) (see Figure 1.1(c)). Additionally to RADNET, the Portuguese Environment Agency performs regular analysis of samples of aerosols, surface waters, and food chain components.

1.1.3 Human activity monitoring

Closed-Circuit Television (CCTV) systems are possibly the main surveillance technology for human activities. Over the last decades, CCTV systems coverage has rapidly grown with the first system reported in 1985 in the United Kingdom. First generation CCTV lacked automation since they rely almost solely on human monitoring, which may lead to data swamping, human error, deliberated profiling, and privacy issues. Although these limitations and the absence of a large number of valid studies about its effectiveness, first-generation systems were effective in certain crime reduction. In [5], the author concluded that there was a statistically significant three-percent crime reduction when studying 12 city center and 2 public housing in comparison with a 45 percent reduction in car parks when compared against the control areas. The results were not so satisfactory when looking for 4 public transport cases, concluding that there was no reduction in this crime typology supported by statistical data.

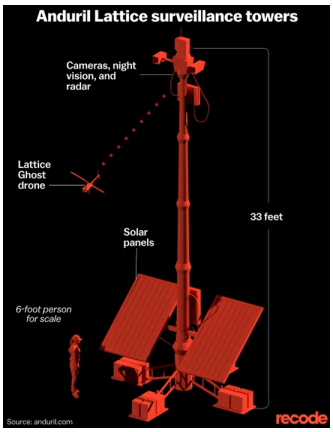
In 2004, Ray Surette [6] suggested that the effectiveness in reducing crime seems to depend on the emotional nature of the crime, the quality of the monitoring staff, and the offender's perception of increased risk of getting caught (which is directly related to the quality and robustness of the surveillance system). Given these limitations, Surette suggests the advantages of using second-generation systems, characterized by automatic digital image processing. While second-generation systems cannot resolve all previously mentioned issues, they can reduce data swamping and other human limitations.

Information extracted from digital images can be combined with information gathered by other technological systems (sensors, emotion analysis of social media publications, crowd reports using mobile

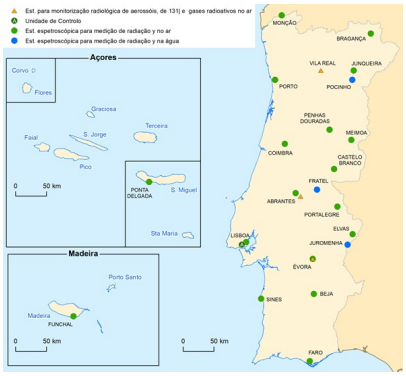
applications) or with human-based monitoring approaches (for instance face to face surveys or traditional security guard surveillance). This way, robustness and redundancy are improved, offering authorities filtered and refined information.

This kind of complex systems are already in use to increase community safety. A Portuguese newspaper [7] collected and compared the number of public cameras authorized by the state to be installed in cities: while in 2013 there were only 38 authorized cameras, in 2021 there were more than 850. Since 2021, the United States Custom and Border Protection has autonomous surveillance towers for West Texas and New Mexico border patrol [8] (see Figures 1.1(a) and 1.1(b)). Finally, the European Commission is searching for companies to develop semi-autonomous vessels, naval collaborative surveillance, and space-related assets in intelligence, surveillance, and reconnaissance [9].

(a) Autonomous surveillance tower used by U.S Custom and Border Protection



(b) Surveillance tower made by Anduril Company, similar to the one used on the U.S border



(c) Sensor network distribution associated with RADNET

Figure 1.1: Examples of surveillance systems

1.2 State of the Art on Surveillance Systems

Creating a surveillance system can be seen as a puzzle challenge. To select the pieces required to assemble, i.e, which technological or non-technological method(s) should be used to gather information, one needs to define the goal and purpose of the surveillance and find suitable methods. Then, to choose the best architecture for those pieces (i.e., how to make them work together), one needs to account for the specification of each method - its reliability, advantages and constraints. The data fusion algorithms defined for the particular architecture will have correlate, integrate and recognize a vast quantity of data in order to obtain useful information. The last milestone is to specify the protocols in place to react to new events either automatically using event-triggered strategies or relegating that task to human users

that assess a high-level summary of the environment. Thus, there can be a symbiotic work of several sub-systems to acquire, process, and manage relevant data from heterogeneous sources.

Any surveillance system based on one technology alone will lack robustness because due to the specific limitations associated with that acquisition method causing uncertainty in its measurements. Table 1.1 summarizes various surveillance methods highlighting limitations and advantages. We remark that some techniques can be viewed as equivalent but we separate them given the different levels of interest from the scientific community. As as example, Social Media Monitoring can be categorized as a crowdsensing system based on the source of information. Moreover, crowdsensing and robot surveillance methods can be seen as a sensor network where each node moves dynamically within the surveilled space. Distinguishing static and dynamics sensor networks paints a finer picture of the literature with specific research examples.

1.2.1 Wildfires Surveillance Systems

Wildfire surveillance systems started as a human-centered method in its early stages. However, covering a large area with often limited accessibility in deep forests is quite difficult to implement resorting to forest rangers alone. A next step has been in implementing observation towers to increase the field of view, which increases the area that each human is responsible to monitor, decreasing detection resolution. Cameras installation in these towers minimizes these issues, either by assisting humans (first generation CCTVs) or by replacing them with second generation CCTVs) [10]. Resorting solely to optical and infrared cameras can hinder detection due to i) not combining it with humidity and pressure variations; and ii), object occlusion and lighting conditions.

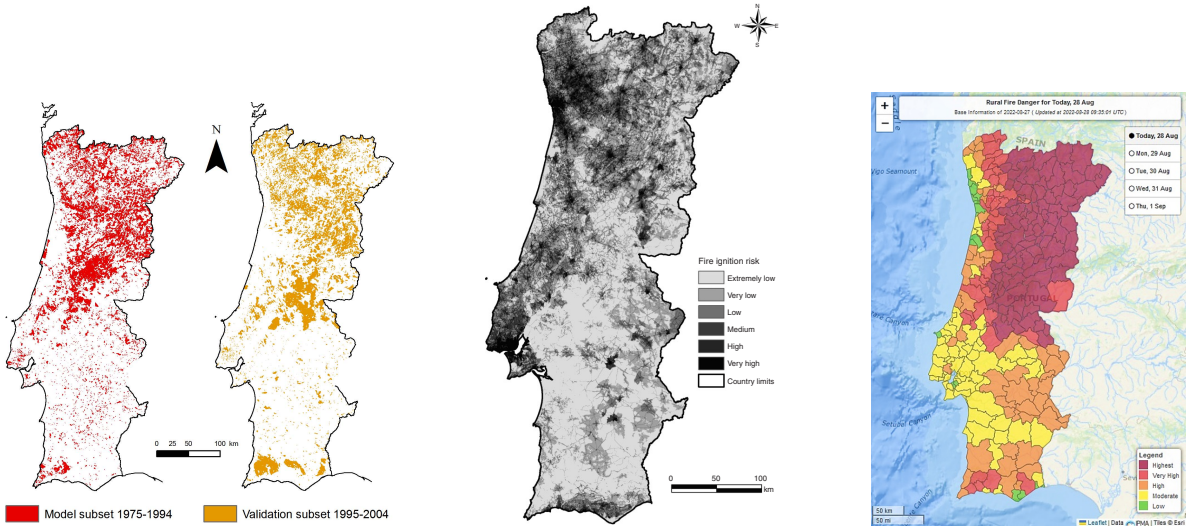
A network of several sensors deployed in a forest can refine the detection and early ignition localization [11, 12] by measuring heat signatures. However, sensor networks bring back a coverage issue since they can be extremely difficult and expensive to deploy over large areas due to the sheer number of sensors required in operation even in a small country like Portugal.

The aforementioned techniques have different profiles in terms of accuracy, resolution, cost and availability. Therefore, combining information from different sources allows the generation of interest maps that can be used by firefighting authorities. A valid interest map in wildfire detection is, for example, the risk measurement based on the social-economic damage that a wildfire can create, weighted by a probability of ignition in a given area. Specific examples of interest maps can be found in the literature. The work in [13] suggests using a regression model with four input variables: population density, land cover (urban-rural, agriculture, shrublands, sparsely vegetated or forest), elevation, and distance to roads, which results in an ignition probability map for the entire Portuguese mainland. Another approach in [14] weights the probability of fire with terrain susceptibility to create a hazard map. In Portugal, the official agency Instituto Português do Mar e da Atmosfera (IPMA) posts the Conjunctural and Meteoro-

Table 1.1: Most used Surveillance Methods

Method	Definition	Pros	Cons	Technological Challenges	References
Human-based Surveillance and Crowdsensing	Surveillance is performed by professionals or by civil citizens on a voluntary base. Data may come from reports or from sensors located on crowd's devices	Voluntary-based reports are highly scalable and cost-effective Humans can adapt and interpret unpredicted cases	Faulty or late alarms Prone to human limitations	Efficient communication Filtering false information possibly injected by some nodes	[17, 18]
Sensors Network	Set of (static) sensors capable of converting a physical phenomenon to an electrical signal. Usually, the sensors work within a network, where each node is placed in a previously defined location.	Low cost of a single sensor Small size A sensor doesn't need a direct line of sight to the phenomenon	Low measurement range of a single unit. Need for exploiting several units. Sensors need to be close to the phenomenon, which can lead to sensor and batteries destruction	Network architecture Bandwidth occupancy when the network is wireless Power consumption	[11, 12, 19]
Cameras and CCTVs	Observation of an environment using analog cameras watched by humans (first generation) or automated image processing of digital cameras (second generation).	Can get a lot of information from static and dynamic environments. Extensive research area. Can use a combination of many types of cameras, such as optical, infrared, depth	Complex algorithms with a high computational burden Prone to environmental conditions (light, obstacle, etc) Privacy issues	Image processing algorithms' efficiency Define the best places to deploy cameras	[19, 20]
Satellite Systems	Geostationary or low earth orbit satellites equipped with optical or infrared cameras	Large field of view Can be used in isolated locations (open waters, forests, deserts)	Lack of continuous measurements, low resolution and possible cloud obstruction High cost of deployment and operated by skilled workers	Algorithm's efficiency Bandwith occupancy	[21, 22]
Social Media Monitoring	Analysis of social media content (photos, interactions, publications) using automatic algorithms. For example, emotion analysis and imagelabeling	The emergence of social media Doesn't require deploying or operating any equipment	Some delays between the event and publications	Is still a relatively recent research area Some emotions, such as irony, are difficult to extract	[18, 23]
Robot Surveillance	Aerial, ground, or marine robots can carry a collection of the previously mentioned systems, processing computers, and power supplies to make surveillance systems more autonomous.	Can adapt and overcome environmental difficulties, such as obstacles or lighting occlusion. Robots can autonomously choose what to do or follow instructions from human users	Need skilled workers for some tasks	Robots' guidance, navigation and control Task allocation	[24–26]

logical Index [15] that is calculated daily using the Canadian Fire Weather Index (FWI) [16] and the Rural Fire Hazard Index. The former combines temperature, wind speed, and relative humidity measured at solar noon along with rain precipitation during the previous 24 hours. This data is combined to get a fuel moisture distribution and effective available fuel over the space and calculate the Initial Speed Index expressing how fast a fire may move. Therefore, the FWI provides a general fire intensity potential index. Illustrative examples of the maps produced by each approach are presented in Figure 1.2.



(a) Ignition probability map from [13] in both training and validation sets.

(b) Hazard Risk map from [14]

(c) Conjunctural and Meteorological Index used by IPMA in [15]

Figure 1.2: Examples of interest maps generated by measurement models on wildfires

Although extremely useful for firefighting authorities to plan their actions, it is relevant to remark that these methods generate almost static maps, which may not suit the goal of creating a real-time surveillance system. The work presented in [13], like all machine learning techniques, needs a lot of data to train, validate and update its model. Moreover, citing the authors of [14], "we did not consider variables that could be best used in dynamic mapping (e.g., wind speed and direction), mostly when fire is already progressing, as our purpose was to map susceptibility in the long term, as a property of the territory". Finally, [15] is updated daily and is not updated with the latest available data. Nevertheless, real-time surveillance systems may complement these indices by interpreting them as *a priori* knowledge. New information from other sources can be fused to update the map providing the *posterior* view after the measurements.

The most common approach for a formulation that combines *a priori* knowledge and real-time evidence are variations of Bayesian Filters, like Kalman Filter [27, 28] or Particle Filter [29]. Several maps may result from this configuration, such that ignition probability, currently active fire probability or prob-

ability of an undetected map, depending on the objectives set forth by the designer of the surveillance system.

1.3 Envisioned Surveillance Architecture

Based on the discussion in the previous sections, it is evident that designing a real-time solution to monitor events can offer significant advantages. This work aims to develop an algorithm that can generate an optimal path 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. It takes into account the technological and operational constraints of drones to maximize the inspection process. For the purpose of this document, it is assumed that a previously developed surveillance system using an *a priori* map, such as the one shown in Figure 1.2, 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 evolves 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 path 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 Figure 1.3, 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 path 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.

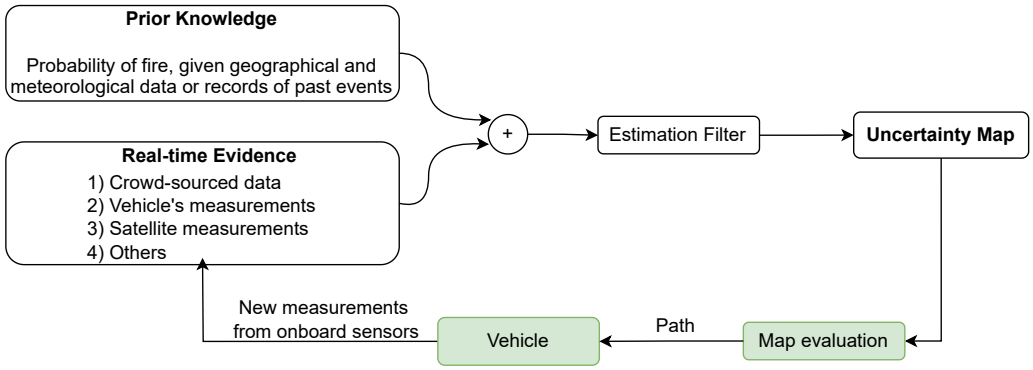


Figure 1.3: Architecture of the Envisioned Surveillance. The green boxes represents the main contributions of this thesis

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

1.3.1 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 path $\varphi^{[k]} \in \Re^{2\times N}$, 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 path obtained from K sequential maps can be represented as a set of waypoints $\varphi \in \Re^{2\times K\cdot N}$ in the form

$$\varphi = [\varphi_1^{[0]}, ..., \varphi_N^{[0]}, ..., \varphi_1^{[k]}, ..., \varphi_N^{[k]}, ..., \varphi_N^{K-1}]. \tag{1.1}$$

This path can be obtained by solving the optimization problem presented in Equation (1.2). 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.

$$\begin{aligned} \underset{\varphi}{\text{maximize}} \quad & \sum_{k=0}^{K-1} \int_{\gamma(\varphi^{[k]})} h_k(x)dx \quad \text{, where } \gamma(\varphi^{[k]}) = \bigcup_{n=1}^{N} C(\varphi_n^{[k]}, r). \\ \text{subject to} \quad & \dot{s} = f(s) \\ & g(s,u) \leq 0 \\ & \varphi_n^{[k]} \in \mathcal{X} \end{aligned} \tag{1.2}$$

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 Figure 1.4(b) 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 path 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 Equation (1.2) into K separate sub-problems:

Sub-problem 0 - while only the first map has been received, $k = 0$:

$$\begin{aligned} \underset{\varphi^{[0]}}{\text{maximize}} \quad & \int_{\gamma(\varphi^{[0]})} h_0(x)dx \quad \text{, where } \gamma(\varphi^{[0]}) = \bigcup_{n=1}^{N} C(\varphi_n^{[0]}, r). \\ \text{subject to} \quad & \dot{s} = f(s,u) \\ & g(s,u) \leq 0 \\ & \varphi_n^{[0]} \in \mathcal{X}, \ \ \forall n \in \{1,2,...,N\} \end{aligned} \tag{1.3}$$

Sub-problem 1 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:

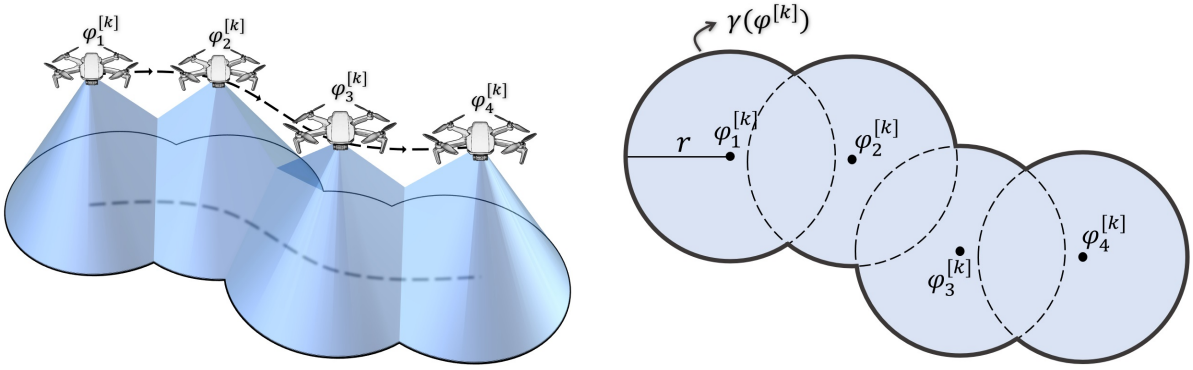
$$\begin{aligned} \underset{\varphi^{[k]}}{\text{maximize}} \quad & \int_{\gamma(\varphi^{[k]})} h_k(x)dx \quad \text{, where } \gamma(\varphi^{[k]}) = \bigcup_{n=1}^{N} C(\varphi_n^{[k]}, r). \\ \text{subject to} \quad & \dot{s} = f(s) \\ & g(s,u) \leq 0 \\ & \varphi_n^{[k]} \in \mathcal{X}, \ \ \forall n \in \{1,2,...,N\} \\ & \varphi_1^{[k]} = \varphi_N^{[k-1]} \end{aligned} \tag{1.4}$$

In Equation (1.4) the blue constraint is added to guarantee concordance between problems.

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 path 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. Figure 1.4 shows an example path for a generic map k with $N = 4$.



(a) 3 dimensional view

(b) Top view

Figure 1.4: Example of a path for a generic map k and $N = 4$

1.4 Thesis Outline

The present document is divided in 4 chapters:

- **Chapter 2 (Non Convex Function Optimization)**: proposes a hybrid optimization algorithm capable of converging to local maxima of non-convex functions, particularly when these functions are modeled as Gaussian mixture. The algorithm has shown convergence in all conducted tests.
- **Chapter 3 (UAV Path Planning and Path Following)**: develops a real-time solution for monitoring wildfire events, based on an empirical observation of the expected uncertainty map, and making use of the hybrid optimization algorithm developed in the second chapter. The proposed algorithm derives a path following control scheme to allow a quadrotor to track the desired path. The complete approach is tested and analyzed, using a powerful simulation tool to emulate the vehicle dynamics.
- **Chapter 4 (Conclusions)**: provides a summary of the work developed in this thesis, and highlights the most important aspects to be retained from this work. A discussion of future work and suggestions is given that highlights additional points that can improve the efficacy of the method.

2

Non Convex Function Optimization

Contents

> "People optimize. Investors seek to create portfolios that avoid excessive risk while achieving a high rate of return. Manufacturers aim for maximum efficiency in the design and operation of their production processes. Engineers adjust parameters to optimize the performance of their designs.
>
> Nature optimizes. Physical systems tend to a state of minimum energy. The molecules in an isolated chemical system react with each other until the total potential energy of their electrons is minimized. Rays of light follow paths that minimize their travel time." [30]
>
> *Jorge Nocedal*
> *Stephen J. Wright*

The primary objective of this chapter 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 problem described in Chapter 1.

To establish the necessary theoretical foundation for the subsequent chapters, Section 2.1 provides an overview of the relevant background theory and introduces the notation used throughout the chapter. In Section 2.2, a summary of commonly used first-order optimization algorithms from the literature is presented. This section provides an understanding of the existing approaches and their strengths and limitations. Moving on to Section 2.3, the problem we aim to solve is formally stated, and an illustrative example is provided to demonstrate the challenges posed by Gaussian Mixture (GM) functions. To address these challenges, we propose a new hybrid algorithm in Section 2.4. Finally, in Section 2.5, the performance of the proposed solution is evaluated. This includes experimental results and analysis that demonstrate the effectiveness and potential of the proposed algorithm.

2.1 Background Theory

Optimization is a mathematical tool widely used in a broad spectrum of scientific areas. Formulating a controller as the solution of an optimization problem requires some steps such as: *i)* modelling the system, i.e, creating a set of equations that govern the evolution of the variables affecting the process; *ii)* define a criteria to quantify the optimality of the solutions; *iii)* finally, identify constraints to the decision variables that need to be respected.

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

$$\underset{x}{\text{minimize}} \quad f(x) \tag{2.1}$$

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 solution x^* to the optimization can be classified

as a global minimizer if $f(x^*)$ is smaller or equal than all the remaining points in the feasible set. On the other hand, if x^* only satisfies that property locally, it is called a local minimizer. Both definitions are relevant to the study at hand and are presented in Definitions 1 and 2.

Definition 1. *A point x^* is a global solution for Equation (2.1), also called a global minimizer, if and only if*

$$f(x^*) \leq f(x) \quad \forall x \in \mathcal{S}. \tag{2.2}$$

Definition 2. *A point x^* is a local solution for Equation (2.1), also called a local minimizer, if and only if there exist an $\epsilon > 0$ such that*

$$f(x^*) \leq f(x) \quad \forall x \in \mathcal{S}, ||x - x^*|| < \epsilon. \tag{2.3}$$

In the context of unconstrained optimization problems, when the objective function $f(x)$ is differentiable, a necessary condition to be satisfied by any x^* is given by

$$\nabla f(x^*) = \mathbf{0}_n, \tag{2.4}$$

where $\nabla f(x^*) \in \Re^{n \times 1}$ stands for the gradient of $f(x)$ evaluated at x^*, and $\mathbf{0}_n$ is a zero vector of size n. For a constrained problem, Equation (2.4) evolves into the Karush-Kuhn-Tucker (KKT) conditions (for further details, the reader is directed to [30]).

Another useful classification is with respect to the convexity of the objective function and the feasibility set as presented in the following definitions, and illustrated in Figure 2.1

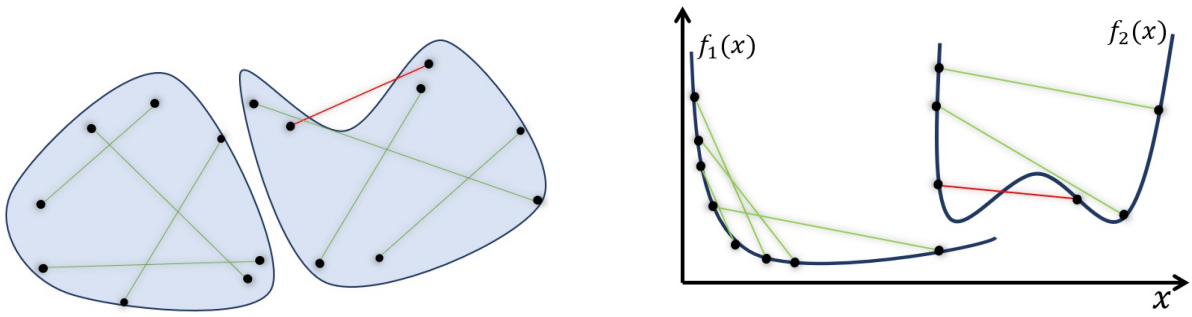
Definition 3. *A set $\mathcal{S} \in \Re^n$ is convex if any straight line connecting any two points in $\mathcal{S}$ lies entirely inside $\mathcal{S}$, see Figure 2.1(a). This is equivalent to say that, for any pair $x, y \in \mathcal{S}$, the following inclusion is verified:*

$$\alpha x + (1 - \alpha) y \in S, \quad \forall \alpha \in [0, 1]. \tag{2.5}$$

Definition 4. *A function f is convex if its domain $\mathcal{S}$ is convex and for any two points x and y in S, the following property is verified (see Figure 2.1(b)):*

$$f\left(\alpha x + (1 - \alpha) y\right) \leq \alpha f(x) + (1 - \alpha) f(y), \quad \forall \alpha \in [0, 1]. \tag{2.6}$$

The importance of this classification is that convexity of f in Equation (2.1) implies that any local minimizer is also a global minimizer.



(a) Left set is convex, but right convex is non-convex

(b) $f_1(x)$ is a convex function, but $f_2(x)$ is a non-convex

Figure 2.1: Examples of convex and non-convex sets and functions

2.1.1 Optimization algorithms

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. The distinction lies in whether they rely solely on the first gradient or also utilize the second gradient of the objective function, known as the Hessian matrix. Computing the Hessian matrix may require significant computational resources and may not always justify the additional complexity. For that reason, we focus on first-order optimization methods. A typical format of first order algorithms is

$$x_{k+1} = x_k + \alpha_k d_k. \tag{2.7}$$

where x_k denotes the current estimate, α_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 α_k.

Definition 5. *d_k is a valid search direction to minimize a function $f(x)$, if it respects*

$$d_k^T \nabla f(x) < 0 \tag{2.8}$$

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

$$f(x_k + \alpha d_k) < f(x_k) \quad \textit{for all } 0 < \alpha \leq \bar{\alpha}. \tag{2.9}$$

To understand Definition 5, let us define an auxiliary function for the current iteration, $\phi(\alpha) = f(x_k +$

$\alpha d_k)$. Recall that this function assumes that: i) we fixed the current iteration, x_k, and ii) a descent direction is already chosen. We can improve our current estimate if we impose that $\dot{\phi}(0) < 0$. Using the chain rule we have

$$\dot{\phi}(0) = d_k^T \nabla f(x_k) < 0, \tag{2.10}$$

which leads to Equation (2.8). Therefore, there exits infinite possibilities for d_k with their correspondent α to improve the current estimate.

The backtracking subroutine is a possible solution to chose the step size α_k. It searches for a step size that improves the current estimate, at least, on a given quantity defined by

$$\gamma \nabla f(x_k)^T (\alpha_k d_k) = \gamma \phi(0) \alpha_k < 0. \tag{2.11}$$

Algorithm 2.1 shows the procedure: β is the step-size update for each backtracking iteration, $\hat{\alpha}$ is the initial step-size value, and γ is a tunable parameter on how much improvement one wants to guarantee.

Algorithm 2.1: Backtracking subroutine to compute the step size $\alpha_k > 0$

begin
- Choose backtracking parameters $\hat{\alpha} > 0$, $0 < \gamma < 0.5$ and $0 < \beta < 1$
- $\alpha_k \longleftarrow \hat{\alpha}$
- **while** $f(x_k + \alpha_k d_k) \geq f(x_k) + \gamma \nabla f(x_k)^T (\alpha_k d_k)$ **do**
  - $\alpha_k \longleftarrow \beta \alpha_k$

Other options include setting a constant step size and letting 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. We can then summarize a general first-order as given in Algorithm 2.2 where a stopping criteria $||\nabla f(x_k)|| < \epsilon$ was used with $\epsilon > 0$.

Algorithm 2.2: Template for a first-order algorithm

begin
- Choose initial estimate x_0
- $k \longleftarrow 0$
- **while** *Stopping criterion is not met* **do**
  - Compute $\nabla f(x_k)$
  - Set descent direction d_k using $\nabla f(x_k)$
  - Select α_k
  - Update $x_{k+1} \longleftarrow x_k + \alpha_k d_k$
  - $k \longleftarrow k + 1$

This work focuses on algorithms that do not utilize the backtracking subroutine as this entails addi-

tional function evaluations to verify the decreasing condition from Equation (2.11). For the problem at hand, the reduction in the number of iterations is not expected to compensate for the increased computational overhead. Moreover, backtracking requires the tuning of additional parameters, making the algorithm sensitive to these choices.

2.2 State of the Art on Gradient Search Algorithms

2.2.1 Gradient Descent

Among various first-order optimization algorithms, the gradient descent version is the most popular one. This version uses an intuitive conclusion from Equation (2.10): if we want to use a descent direction so that $\dot{\phi}(0)$ is maximized, we may look for a solution to

$$\cos\left(\angle(d_k, \nabla f(x_k))\right) = -1 \iff d_k = -\nabla f(x_k). \tag{2.12}$$

This comes from the fact that Equation (2.10) is the same as the inner product $\langle d_k, \nabla f(x_k)\rangle = ||d_k|| \cdot ||\nabla f(x_k)|| \cos\left(\angle(d_k, \nabla f(x_k))\right)$. Another intuitive reason is that the gradient of an n-dimensional function is a vector pointing to the direction of its greatest increase. Therefore, the best direction (assuming no other knowledge) is to move x_k for the greatest decrease of f.

Due to its easy implementation and low computational cost, the gradient descent algorithm has been widely used in various fields, including neural network training [31] and control of sensor networks [32]. However, it exhibits two primary challenges:

- Slow convergence in certain cases, even when the step size is determined based on prior knowledge of the objective function. Moreover, a poor specification of this parameter can lead to a divergence of the estimation from the true solution.

- Relying solely on the gradient value at the current iteration provides limited information localized around the current point, which may delay the discovery of the global solution. To address these limitations and improve convergence rates, enhanced versions of gradient descent have been developed, including stochastic gradient descent and momentum gradient descent.

2.2.2 Adaptive Step

To mitigate the reliance on prior knowledge of the objective function, various techniques have been introduced in the literature to update the step size α at each iteration. These methods aim to enhance the performance of fixed step sizes while avoiding the need for backtracking algorithms.

One of these approaches is proposed by Almeida & Silva [33]. They start by considering a single-dimensional function $f(\theta) : \Re \rightarrow \Re$ and conjecture that the sign of the gradient provides information about the direction of steepest ascent or descent, which can be used to guide a step size updating process. Specifically, it is proposed that if two consecutive iterations have opposite directions for the gradient, it means that the method already jumped over a minimum. On the other hand, if two consecutive iterations have the same direction, then the minimum has not been reached. In the first case, the step size should decrease, and in the second case, the step size can increase to accelerate the convergence rate. This approach can be summed up as

$$\alpha^k = \begin{cases} u \cdot \alpha^{k-1} & \text{if } \frac{df}{dx}(x_{k-1}) \cdot \frac{df}{dx}(x_k) > 0 \\ d \cdot \alpha^{k-1} & \text{otherwise} \end{cases} \quad , \tag{2.13}$$

where $0 < d < 1$ and $u > 1$ are respectively the "up" and "down" update constants. For multidimensional functions, Almeida & Silva apply the update individually to each dimension.

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. This is a valid approach for this algorithm because the value of the parameter α can be adapted along the run.

2.2.3 Momentum Algorithms

This last approach opens the possibility to use information gathered in the last iterations, instead of resorting only to instantaneous information. Polyak proposes an extra term to Equation (2.7), in [34], which adds an inertia or momentum contribution to the current estimation. This momentum term "will lead to motion along the *essential* direction" creating a method inspired by the movement of a Heavy Ball. Nesterov's Accelerated method extends the Heavy Ball momentum concept to the gradient's computing point. It can be seen as a second momentum term. A third momentum is proposed in [35], where the authors extend the last two momenta to the overall estimation.

These three methods are summarized in Equations (2.14) to (2.16). The terms $\widetilde{x}_k$ and $\overline{x}_k$ represent respectively the first and second momentum terms. The Triple Momentum can be seen as an inertial movement applied to the Nesterov acceleration method, with momentum constant δ.

$$x_{k+1} = \widetilde{x}_k - \alpha \nabla f(x_k) \quad , \quad \text{where} \quad \widetilde{x}_k = x_k + \beta(x_k - x_{k-1}) \qquad \text{(Heavy Ball)} \tag{2.14}$$

$$x_{k+1} = \widetilde{x}_k - \alpha \nabla f(\overline{x}_k) \quad , \quad \text{where} \quad \overline{x}_k = x_k + \beta(x_k - x_{k-1}) \qquad \text{(Nesterov's)} \tag{2.15}$$

$$x_{k+1} = x^N_{k+1} + \delta(x^N_{k+1} - x^N_k) \quad , \quad \text{where } x^N_k \text{ is Nesterov estimate} \qquad \text{(Triple Momentum)} \tag{2.16}$$

2.2.4 Optimal Parameters

The authors of [36] define a procedure to find optimal parameters for the gradient method, the Heavy Ball method, and Nesterov's method. This methodology has also been followed to find parameters for various algorithms devoted to optimization and solution of linear equations as reported in [37]. Optimal parameters are defined as guaranteeing a minimal worst-case convergence rate ρ_{worst}, as in Definition 6.

Definition 6. *A sequence x_k is converging with convergence rate $0 < \rho < 1$ if there exists a positive constant $C > 0$ such that*

$$||x_{k+1} - x^*|| \leq C\rho^k ||x_0 - x^*|| \tag{2.17}$$

Definition 7. *A linear dynamical system represented by n state variables and influenced by d inputs can be represented by a state vector $\xi_k \in \Re^n$, that can be iteratively updated by a set of recursive linear equations of the form*

$$\begin{aligned} \xi_{k+1} &= A\xi_k + Bu_k, \quad \textit{where } u_k \in \Re^d \\ x_k &= C\xi_k + Du_k, \quad \textit{where } y_k \in \Re^m \end{aligned} \tag{2.18}$$

We will represent state space systems as $\left[\begin{array}{c|c} A & B \\ \hline C & D \end{array}\right]$.

A first-order optimization algorithm can be modeled as a linear dynamic system as in Equation (2.18), where x_k represents the current estimate, ξ_k denotes an internal state of the algorithm, and $u_k = \nabla f(x_k)$. In doing so, it is possible to unify the convergence study using the techniques for linear systems where each algorithm in Equations (2.14) to (2.16) is represented as:

$$\left[\begin{array}{c|c} I_n & -\alpha I_n \\ \hline I_n & 0_n \end{array}\right]_{Gradient} \qquad \left[\begin{array}{cc|c} (1+\beta)I_n & -\beta I_n & -\alpha I_n \\ I_n & 0_n & 0_n \\ \hline I_n & 0_n & 0_n \end{array}\right]_{HeavyBall}$$

$$\left[\begin{array}{cc|c} (1+\beta)I_n & -\beta I_n & -\alpha I_n \\ I_n & 0_n & 0_n \\ \hline (1+\beta)I_n & -\beta I_n & 0_n \end{array}\right]_{Nesterov's} \tag{2.19}$$

Please recall that the Heavy Ball and Nesterov algorithms have two internal states, corresponding to the current and previous estimates. I_n denotes an n-by-n identity matrix, and 0_n represents an n-by-n matrix of zeros.

Suppose the objective function is a quadratic function given by $f(x) = \frac{1}{2}x^TQx - p^Tx + r$, with $Q \in \Re^{n\times n}$ symmetric, $mI_n \preceq Q \preceq LI_n$ in the positive definite ordering and $0 < m < L$. The gradient of the objective function is $\nabla f(x) = Qx - p$ and the optimal solution is $x^* = Q^{-1}p$. In this context, the input u_k can expressed as

$$u_k = \nabla f(x_k) = Qx_k - p = Q(x_k - x^*) = QC(\xi_k - \xi^*). \tag{2.20}$$

Please note that once the system converges to the optimal solution x^*, the state equations from Equation (2.18) become $x^* = C\xi^*$, and $\xi^* = A\xi^*$.

By substituting Equation (2.20) in Equation (2.18), one gets

$$\xi_{k+1} = A\xi_k + BQC(\xi_k - \xi^*) \Leftrightarrow \xi_{k+1} - \xi^* = (A + BQC)(\xi_k - \xi^*), \tag{2.21}$$

meaning that the error evolves with the state transition matrix $T \coloneqq A + BQC$. A necessary and sufficient condition for ξ_k to converge to ξ^* is that the spectral radius of T is strictly less than 1. Thus, the worst-case convergence rate for a first-order optimization algorithm applied to a quadratic function is given as

$$\rho_{worst} = \underset{mI_n \preceq Q \preceq LI_n}{\text{maximize}}\ \rho(T), \quad \text{where } T = A + BQC. \tag{2.22}$$

The analysis of the spectral radius of T has been conducted in [36] for the Gradient descent, Heavy-ball Nesterov's accelerated methods with the corresponding optimal parameters. A similar analysis has also been conducted for the case of Nesterov's method with time-varying parameters as in [38] and the case of quadratic functions with $m = 0$ and an infinite number of minimizers [39].

For the Triple Momentum method, the work in [35] does a similar framework to get optimal parameters for a more general type of functions: m-strongly convex functions with L-Lipschitz continuous gradient for a given $0 < m < L$. In other words, the optimal parameters for Triple Momentum are computed for a function $f : \Re^n \to \Re$ that satisfies Equation (2.23)

$$m||x - y||^2 \leq (\nabla f(x) - \nabla f(y))^T (x - y) \leq L||x - y||^2, \quad \forall x, y \in \Re^n. \tag{2.23}$$

Please note that a quadratic function, with the previously defined Q matrix, is an example of an m-strongly convex function with L-Lipschitz continuous gradient. Table 2.1 shows the expressions for the optimal parameters.

Table 2.1: Optimal parameter for each method and correspondent worst-case convergence rate. $\kappa = L/m$ represents the condition number.

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

The optimal parameters derived from the worst-case analysis enable us to select appropriate values for each algorithm, taking into account the L-Lipschitz and m-strongly convex constants and ensuring convergence for strongly convex functions. However, it is important to note that these constants, L and m, describe the behavior of the function across its entire domain, which may not accurately reflect local behaviors of the function in some regions where these constants can differ significantly from their global counterparts. Therefore, it is essential to develop hybrid algorithms that can dynamically adapt the algorithm's parameters based on the current estimation, allowing for better performance and convergence in various regions of the optimization domain.

2.2.5 Hybrid Algorithm

The authors of [40] propose the use of a hybrid algorithm for optimizing convex functions, which aims to 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: (α_1, β_1) and (α_2, β_2). The algorithm can be defined as follows:

$$x_{k+1} = x_k - \alpha_{\sigma(k)} \nabla f(x_k) + \beta_{\sigma(k)}(x_k - x_{k-1}), \quad \text{where } \sigma(k) \in \{1, 2\}. \tag{2.24}$$

The function $\sigma(k)$ is updated in every iteration and takes values from the set $\{1, 2\}$ based on a supervisor. The paper suggest two supervisors, one using both the objective function and its gradient, and another using only its gradient.

The choice of parameters is key for the algorithm performance. The first set of parameters is tuned to guarantee fast convergence when the estimation is far from the solution, whereas closer to the minimizer, the authors are selecting values adjusted to prevent oscillation. This work focused on the case of convex functions suing 2 versions of the Heavy ball. In this thesis, since the optimization function is nonconvex, we draw inspiration in this type of hybrid algorithms and propose a different version for nonconvex functions after testing different combinations and formulations for the supervisor.

2.3 Problem Statement

In the previous section, we have reviewed first-order methods with the analysis concluding the need for hybrid algorithms algorithms even for the case of strongly convex functions. In this section, we will be overviewing the characteristics of the tackled problem that will shape the design of the proposed method. Given that the objective function for the wildfire detection is going to be the result of an estimation related to different characteristics related to 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 GM model. In this chapter, since

we have stripped the constraints from the optimization, we recover the same formulation as in the past section:

$$\underset{x}{\text{maximize}} \quad h(x). \tag{2.25}$$

However, given the nonconvex nature of h, it may possess multiple local maximizers and exhibit zones with very asymmetric values for the Lipschitz constant. In the next subsections, we highlight specific characteristics that will shape the proposed algorithm.

2.3.1 Gaussian mixture as a non-convex function

Given that our original problem (as shown in Equation 1.2) involves an objective function that relies on an uncertainty map obtained from sensor data, in this chapter, we employ GM models as the objective function $h(x)$. GM functions are extensively utilized in various domains, including statistics, machine learning, and signal processing, due to their inherent flexibility and capability to represent complex functions. Moreover, in theory, any function can be approximated by a GM model provided a sufficient number of Gaussian components are used, and their weights and parameters are appropriately chosen. Specifically, GMs prove to be particularly useful when modeling real-world data obtained from sensors.

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.,

$$h(x) = \sum_{k=1}^{K} w_k \mathcal{N}_k(x), \tag{2.26}$$

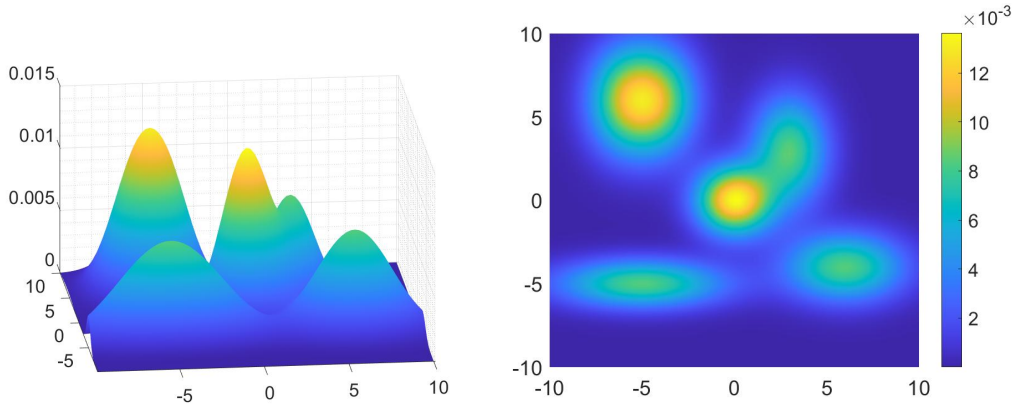
$$\text{where} \qquad \sum_{k=1}^{K} w_k = 1 \qquad \mathcal{N}_k(x) = \frac{1}{2\pi\sqrt{|\Sigma_k|}} e^{-1/2(x-\mu_k)^T \Sigma_k^{-1}(x-\mu_k)}.$$

The function $\mathcal{N}_k(x)$ represents the Gaussian probability density function for component k with mean μ_k and covariance Σ_k.

Figure 2.2 exemplifies a possible instance of $h(x)$ with 6 Gaussians. Please note that 2 of the Gaussians are very close to each other, almost overlapping, which creates a large peak around $x = (-5, 6)$.

2.3.2 Discrete Objective Function

The surveillance problem described in Chapter 1 motivated the formulation in Equation (2.25) and posed the assumption of a continuous map. However, if we are resorting to fire hazard indices, these values will be given as a discrete map. This is motivated because the surveillance-gathering filter is not able



(a) 3 dimensional view **(b)** 2 dimensional view

Figure 2.2: Example of an GM function with 6 Gaussians, generated using Equation (2.26)

to generate a continuous function $h(x)$ as the measurements are taken in squares of territory. Such motivates the use of discretized functions following the next definition.

Definition 8. *A discrete function* $h^{(d)}(n_1, n_2) : \mathbb{N}^2 \to \Re$ *is obtained from a continuous function* $h(x) : \Re^2 \to \Re$ *by dividing the domain into a grid of cells with resolution* Δ*. Specifically, we have:*

$$h^{(d)}(n_1, n_2) = h([n_1\Delta, n_2\Delta]). \tag{2.27}$$

Given that continuous gradient values cannot be used in this context, an alternative pseudo-gradient, denoted as $\nabla^{(d)}$, will be used instead. This is achieved by means of a finite difference gradient, defined as:

$$\nabla^{(d)} = \frac{1}{2\Delta}\left[h^{(d)}\left(n_1+1, n_2\right) - h^{(d)}\left(n_1-1, n_2\right)\right) \ , \ h^{(d)}(n_1, n_2+1) - h^{(d)}(n_1, n_2-1))\right]. \tag{2.28}$$

When using this pseudo-gradient to obtain the next estimate, it is highly probable that the result may fall outside a discretized position. This will also require the method to perform a selection of the nearest discretized position.

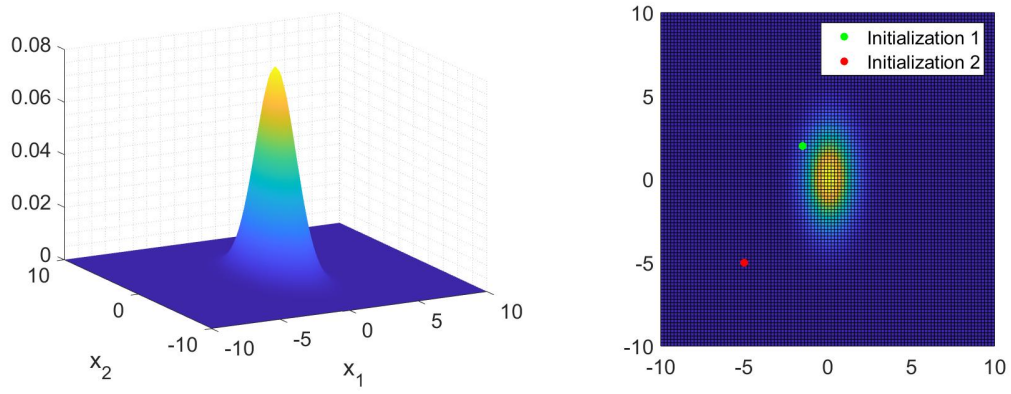
2.3.3 Illustrative example

In order to gather intuition related to the use of GM models and the challenges arising from plateau regions in the search space that produce negligible gradients, we present an illustrative example that will guide the design in the remaining of this chapter. We start by comparing three groups of algo-

rithms: gradient descent (Section 2.2.1), momentum algorithms (Section 2.2.3), and gradient descent with adaptive step (Section 2.2.2). These algorithms will be evaluated using fixed parameters obtained through manual tuning using several runs to improve their values. We will also assume access to the continuous functions.

The experiments were conducted using a GM with K = 1, $\mu = \begin{bmatrix} 0 & 0 \end{bmatrix}^T$ and $\Sigma = \begin{bmatrix} 1 & 0 \\ 0 & 5 \end{bmatrix}$, as shown in Figure 2.3. 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

$$||x^{k-1} - x^k|| \leq 10^{-4}. \tag{2.29}$$



(a) GM used as objective function

(b) Each algorithm was tested using 2 different initialization

Figure 2.3: Objective function and algorithms initialization

The final parameter values used for each algorithm are listed in Table 2.2. Manual tuning is not an ideal option when dealing with complex and unpredictable objective functions. For this reason, one of the goals of the upcoming sections is to reduce human intervention in the parameter selection process. Regarding the adaptive step, these parameters mean that the first iteration uses $\alpha = 0.5$ in both x_1 and x_2 directions. Then, after each iteration cycle, α is updated independently for each direction using the updating factors d and u, as defined in Equation (2.13).

Table 2.2: Parameters used in each simulation for both initialization 1 and 2

Method	Initialization 1		Initialization 2	
Gradient Descent (GD)	$\alpha = 20$		$\alpha = 50000$	
Nesterov's (NE)	$\alpha = 15$	$\beta = 0.25$	$\alpha = 50000$	$\beta = 0.25$
Heavy Ball (HB)	$\alpha = 20$	$\beta = 0.1$	$\alpha = 60000$	$\beta = 0.1$
Triple Momentum (TM)	$\alpha = 19$ $\gamma = 0.1$	$\beta = 0.15$ $\delta = 0.3$	$\alpha = 60000$	$\beta = 0.15$ $\delta = 0.3$
Adaptive	$\alpha^0 = 0.5$ $d = 0.8$	$u = 1.05$ Normalized gradient	$\alpha^0 = 0.5$ $d = 0.8$	$u = 1.05$ Normalized gradient

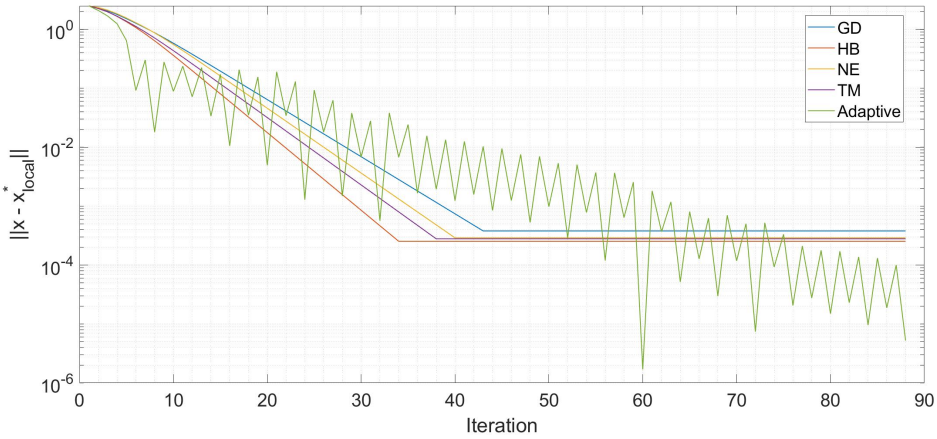
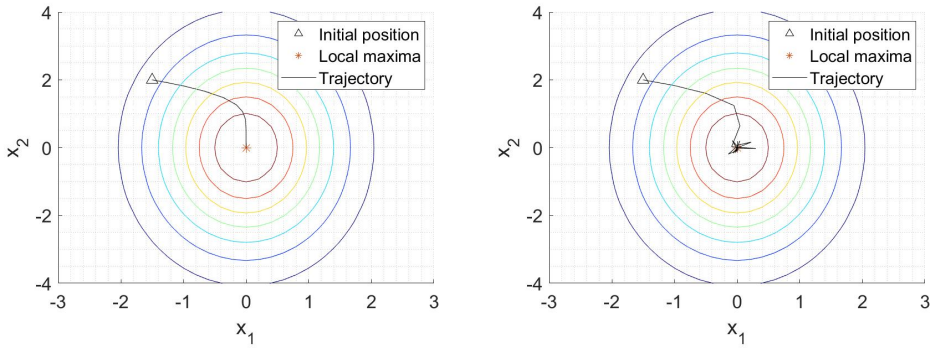


Figure 2.4: Distance to the maximum in each iteration using initialization point 1

For the first initial point, $x^0 = \begin{bmatrix} -1.5 & 2 \end{bmatrix}^T$, the performance can be seen in Figure 2.4. 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. On the other hand, the adaptive step algorithm required nearly 90 iterations due to the time needed to reduce the α parameter while the estimate oscillated around x^*. Regarding gradient descent and its momentum version, the Heavy Ball and Nesterov seem to show better performance when compared to the gradient descent and Triple Momentum.

The trajectory created by the gradient descent and the adaptive step with the first initialization is shown in Figure 2.5. We omit the remaining algorithms since the trajectories are very similar and the major difference is the adaptive step oscillation near the maximum.



(a) Gradient descent trajectory **(b)** Adaptive step trajectory

Figure 2.5: Trajectory created by the estimations of 2 optimization algorithm with initialization 1

For the second initial point, $x^0 = \begin{bmatrix} -5 & -5 \end{bmatrix}^T$, the performance can be seen in Figure 2.6, while the trajectory of the gradient descent and the adaptive step algorithms are shown in Figure 2.7. This example highlights the already expected challenge regarding plateau regions, where the algorithms need high α parameters to be able to move its estimate in such a low gradient magnitude area. However, 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.

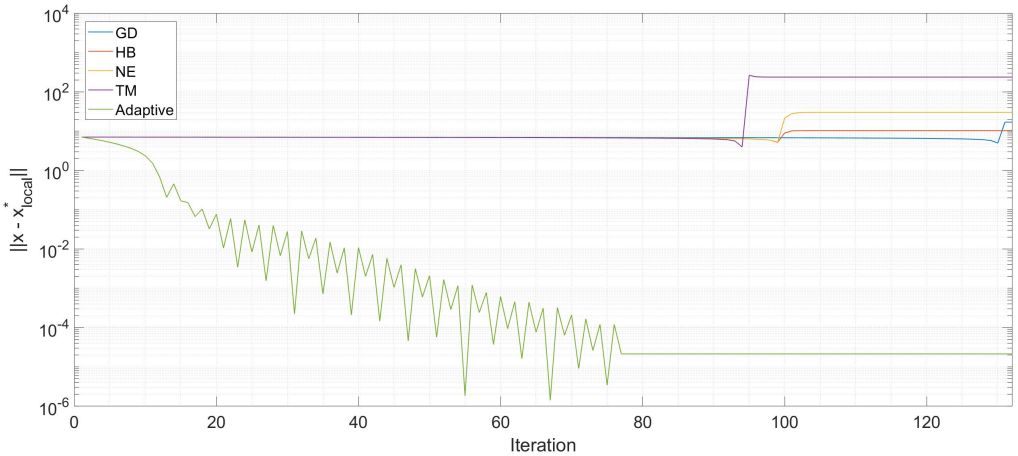
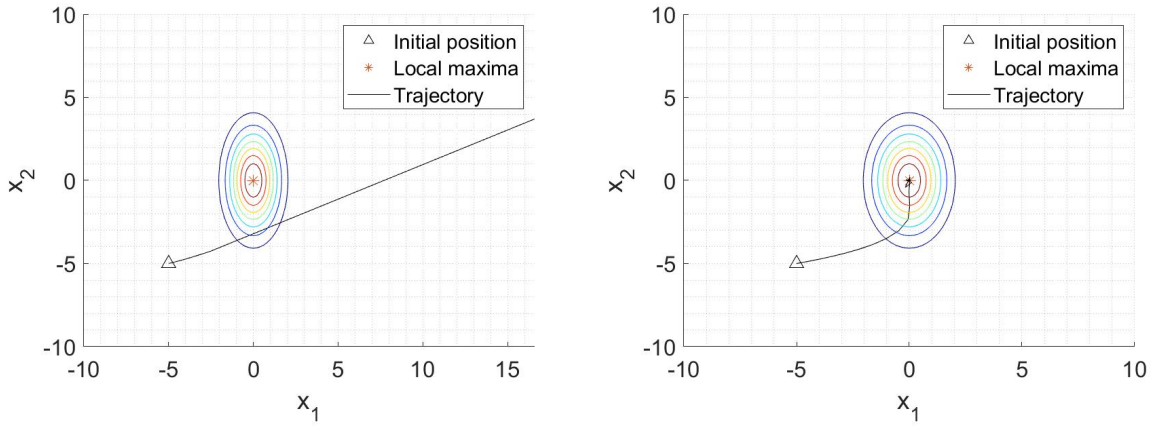


Figure 2.6: Distance to the maximum in each iteration using initialization point 2



(a) Gradient descent trajectory

(b) Adaptive step trajectory

Figure 2.7: Trajectory created by the estimations of 2 optimization algorithm with initialization 2

2.3.4 Conclusion from the illustrative example

In conclusion, GM functions pose a significant challenge for optimization algorithms due to the presence of plateau regions in the search space, as shown in 2.8. These regions correspond to areas where the magnitude of the gradient of the objective function becomes very small. Even when algorithm parameters are tuned to address these regions, there is no guarantee of good performance in regions near the maximum where the gradient increases.

The results of the illustrative example clearly emphasize these facts, highlighting the need to employ an algorithm that can adapt its behavior while optimizing a GM objective function. It is evident that such a tool is essential for overcoming the challenges posed by plateau and non-plateau regions.

Taking into account these considerations, Section 2.4 presents a hybrid algorithm capable of effectively converging to local maximums of GM functions, even when the functions contain multiple Gaussians and the initial estimate resides in a plateau region.

2.4 Proposed solution

This section presents a solution to address the problem defined in Section 2.3. Taking into account the challenges identified during the maximization of GM functions discussed in the previous section, we propose a hybrid algorithm that effectively tackles these issues. 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

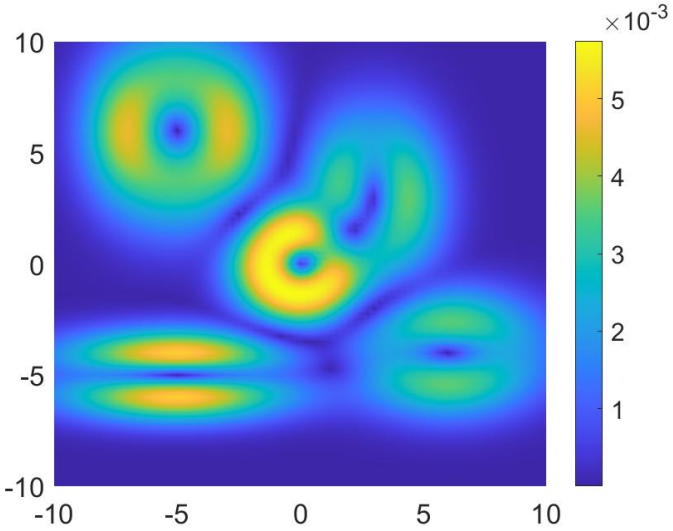


Figure 2.8: Gradient norm of $h(x)$ example shown in Figure 2.2

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 explained on Section 2.2.4. 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.

In Section 2.4.1, we propose an oracle capable of providing these parameters to the hybrid algorithm. However, it is worth noting that the hybrid algorithm can be used with any other source of information that can supply these parameters. In Section 2.4.2, we provide a detailed description of the proposed algorithm.

2.4.1 Gaussian mixture Oracle design

Oracles are a popular tool in optimization as they allow access to information about the objective function, representing an idealized resource for solving problems. However, oracles can sometimes be computationally expensive.

This work develops an oracle for estimating the L-Lipschitz constant of a given GM. Additionally, it also estimates the m-convexity constant of the non-plateau regions around the local maxima. To do so, it employs an accuracy versus complexity trade-off approach to avoid the excessive computational burden.

In this chapter, without loss of generality, we go through the design assuming diagonal covariance matrices in the form

$$\Sigma_k = \begin{bmatrix} \sigma_1^2 & 0 \\ 0 & \sigma_2^2 \end{bmatrix}. \tag{2.30}$$

In doing so, the Gaussian probability density on Equation (2.26) turns into

$$\mathcal{N}_k(x) = \frac{1}{2\pi\sigma_1\sigma_2} e^{\left(\frac{-(x_1-\mu_1)^2}{2\sigma_1} + \frac{-(x_2-\mu_2)^2}{2\sigma_2}\right)}. \tag{2.31}$$

Our oracle employs a second-order Taylor approximation for each Gaussian in the GM function. Each expansion is centered in a Gaussian mean as:

$$\begin{aligned} \mathcal{N}(x) \approx \mathcal{N}(\mu) + (x_1-\mu_1)\frac{\partial \mathcal{N}}{\partial x_1}(\mu) + (x_2-\mu_2)\frac{\partial \mathcal{N}}{\partial x_2}(\mu) + \frac{(x_1-\mu_1)^2}{2}\frac{\partial^2 \mathcal{N}}{\partial x_1^2}(\mu) \\ + (x_1-\mu_1)(x_2-\mu_2)\frac{\partial^2 \mathcal{N}}{\partial x_1 x_2}(\mu) + \frac{(x_2-\mu_2)^2}{2}\frac{\partial^2 \mathcal{N}}{\partial x_2^2}(\mu). \end{aligned} \tag{2.32}$$

Since the Gaussian distribution has a null gradient at $x = \mu$, and given the diagonal covariance matrix assumed in Equation (2.30), the following terms vanish:

$$\begin{aligned} \frac{\partial \mathcal{N}}{\partial x_1}(\mu) = 0, \\ \frac{\partial \mathcal{N}}{\partial x_2}(\mu) = 0, \\ \frac{\partial^2 \mathcal{N}}{\partial x_1 x_2}(\mu) = 0. \end{aligned} \tag{2.33}$$

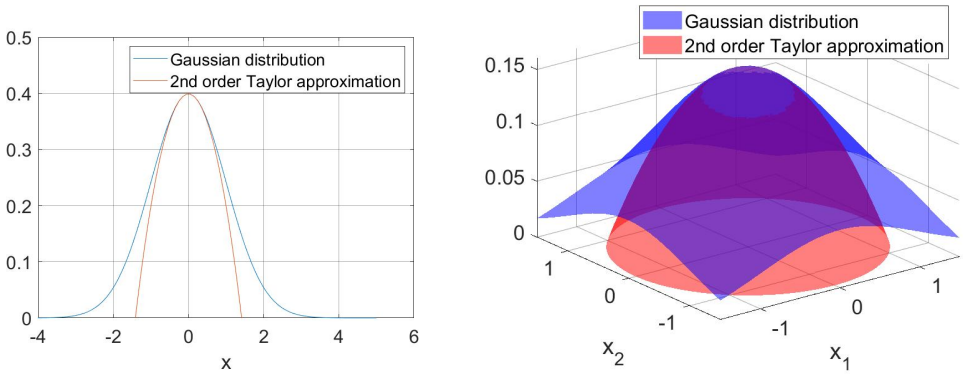
This results in a function that reasonably approximates the Gaussian distribution around its mean point, as shown in Figure 2.9.

At this point, our oracle applies the definition of the constants L and m, as given in Equation (2.23) 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:

$$H = \frac{1}{2\pi\sigma_1\sigma_2} \begin{bmatrix} 1/\sigma_1^2 & 0 \\ 0 & 1/\sigma_2^2 \end{bmatrix}, \tag{2.34}$$

associated with each quadratic term. Since Equation (2.34) is a 2×2 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. Another option would be to choose the output as the biggest L_k and the corresponding m_k value. However, the tests conducted in this work did not demonstrate a considerable advantage in



(a) 1 dimensional Gaussian **(b)** 2 dimensional Gaussian

Figure 2.9: Example of Gaussian functions and respective second-order Taylor's approximations, as Equation (2.32)

using this latter option.

Algorithm 2.3 provides a summary of the described algorithm. By using the output of this oracle, it is possible to use the optimal parameters expression, explained in Section 2.2.4, to run the gradient descent and its momentum versions without requiring human intervention for tuning the algorithms to a specific objective function. However, the user must consider two important points:

- The L and m parameters computed by this oracle do not necessarily correspond to the true L-Lipschitz and m-convexity constants of the objective function. In fact, the m-convexity constant of a Gaussian function is 0, due to the flatness of the plateau region;
- The computations used in this oracle only take into account the regions near the center of each individual Gaussian, so the usage of these L and m parameters may only be appropriate near or inside the non-plateau regions of the complete function.

Despite these remarks, this oracle remains a useful tool for finding local maxima of non-convex functions, as will be demonstrated later. To accomplish this, we will use a hybrid algorithm based on the one from [40], which is explained in the following section.

2.4.2 Hybrid Algorithm for Gaussian mixture functions

As concluded from Section 2.3.3, finding local maximums of GM functions requires an algorithm capable of adapting its parameters to overcome the challenges posed by plateau regions. This is because the suitable parameters for these regions are not suitable for the rest of the function near the maxima, where the algorithm should prevent oscillations. This section describes such an algorithm, inspired by the work from [40].

Algorithm 2.3: GM Oracle

begin
  Set $m \leftarrow 0$, $L \leftarrow 0$
  for *all K Gaussians* **do**
    $\Sigma \leftarrow \Sigma_k$
    $secondGradient \leftarrow \frac{\omega_k}{2\pi}\left[(\sigma_{xx}^3 \cdot \sigma_{yy})^{-1}, (\sigma_{yy}^3 \cdot \sigma_{xx})^{-1}\right]$
    $L_k \leftarrow \max(secondGradient)$
    $m_k \leftarrow \min(secondGradient)$
    if $L_k > L$ **then**
      $L = L_k$
    if $m_k < m$ **then**
      $m = m_k$

The intuitive idea behind the algorithm is to divide its behavior based on the proximity to a maximum or 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:

$$\widetilde{\nabla}_1^{k^{2nd}} = \frac{f(x_1^k + \delta x) - 2f(x_1^k) + f(x_1^k - \delta_x)}{\delta_x^2}, \tag{2.35}$$

$$\widetilde{\nabla}_2^{k^{2nd}} = \frac{f(x_2^k + \delta_x) - 2f(x_2^k) + f(x_2^k - \delta_x)}{\delta_x^2}, \tag{2.36}$$

where δ_x represents the step size for the finite difference approximation. The step size can be specified by the user if the optimization algorithm has access to a continuous objective function. However, if only a discrete version is available (see Section 2.3.2), the better option is to use a step size equal to the domain resolution, i.e, $\delta_x = \Delta$.

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).

Figure 2.10 formalizes the state machine used.

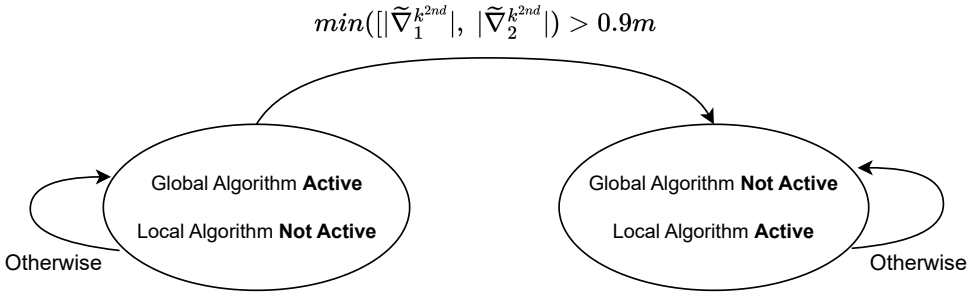


Figure 2.10: Hybrid algorithm supervisor decision

The introduction steps of the algorithm have left out the selection of both the local and global algorithms. Before advancing to that selection, we introduce an extra step required for the case of discrete functions.

2.4.2.A Supervisor's extra step for discrete functions

For discrete functions, the supervisor has to perform extra computations since the current estimate may not lie in a discrete point of the domain. Rounding the estimate to the nearest existing domain position can be a quick solution, but it may lead to large roundings and introduce errors in the estimation.

To address this situation, our algorithm applies a bilinear interpolation method. This method assumes that the values in the discrete domain vary linearly across each row and column. Although this is a rough approximation of what happens with the second gradient of a GM function, if a sufficiently fine discretization is used, bilinear interpolation can be a useful technique for estimating the second gradient of a GM function using finite difference equations.

To apply bilinear interpolation to estimate the second gradient of a GM function, we start by computing the finite difference approximation at each of the four nearest points in the discrete domain. In Figure 2.11, these points are denoted as p_{00}, p_{01}, p_{10}, and p_{11}. The bilinear interpolation process starts by applying linear interpolation to the values at p_{00} and p_{10} to create an intermediate point Q_1, and linear interpolation to the values at p_{01} and p_{11} to create another intermediate point Q_2. That is:

$$\widetilde{\nabla}^2(Q_0) = \frac{x_1^k - n_1\Delta}{(n_1+1)\Delta - n_1\Delta}\widetilde{\nabla}^2(p_{10}) + \frac{(n_1+1)\Delta - x_1^k}{(n_1+1)\Delta - n_1\Delta}\widetilde{\nabla}^2(p_{00}), \tag{2.37}$$

$$\widetilde{\nabla}^2(Q_1) = \frac{x_1^k - n_1\Delta}{(n_1+1)\Delta - n_1\Delta}\widetilde{\nabla}^2(p_{11}) + \frac{(n_1+1)\Delta - x_1^k}{(n_1+1)\Delta - n_1\Delta}\widetilde{\nabla}^2(p_{01}). \tag{2.38}$$

Finally, it applies linear interpolation to the values at Q_1 and Q_2 to estimate the value of the second gradient at the desired point, x^k:

$$\widetilde{\nabla}^2(x^k) = \frac{x_2^k - n_2\Delta}{(n_2+1)\Delta - n_2\Delta}\widetilde{\nabla}^2(Q_1) + \frac{(n2+1)\Delta - x_2^k}{(n_2+1)\Delta - n_2\Delta}\widetilde{\nabla}^2(Q_0). \tag{2.39}$$

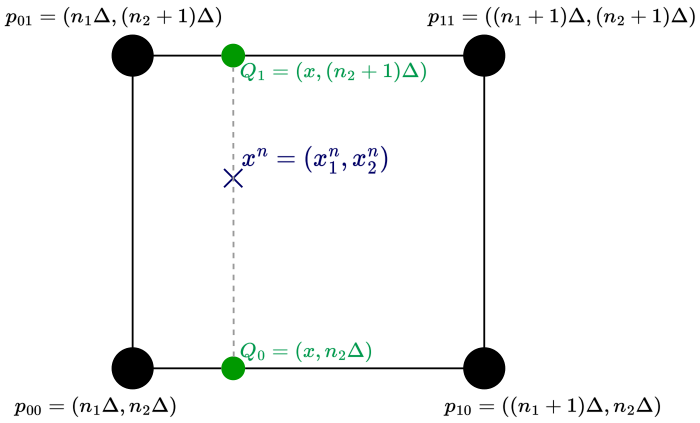


Figure 2.11: Bilinear interpolation

2.5 Results

As seen in Section 2.3.3, among the tested non-hybrid algorithms, only the adaptive algorithm was able to converge to a local maximum regardless of whether the initial estimate is in a plateau or non-plateau region. Therefore, 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 GM function. The following options were considered:

- Adaptive step as the global algorithm and gradient descent with parameters defined using the oracle output and Table 2.1 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 Table 2.1 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 Table 2.1 as the local algorithm - Identified as ”A+TM”.
- Adaptive step as the global algorithm and Triple Momentum algorithm with parameters defined using the oracle output and Table 2.1 as the local algorithm - Identified as ”Adaptive”.

2.5.1 Test 1: GM with $K = 1$

The first set of tests is similar to the one presented in Section 2.3.3, i.e, with $K = 1$, $\mu = \begin{bmatrix} 0 & 0 \end{bmatrix}^T$ and $\Sigma = \begin{bmatrix} 1 & 0 \\ 0 & 5 \end{bmatrix}$, as shown in Figure 2.3. 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 algorithm parameters for each optimization method are listed in Table 2.3

Table 2.3: Test 1 - Parameters generated using the oracle L and m constants and Table 2.1

Method	Parameters
Gradient Descent (GD)	$\alpha = 20.1062$
Nesterov's (NE)	$\alpha = 15.4663$ $\beta = 0.2864$
Heavy Ball (HB)	$\alpha = 20.1062$ $\beta = 0.1111$
Triple Momentum (TM)	$\alpha = 18.8496$ $\beta = 0.1667$ $\gamma = 0.1111$ $\delta = 0.3333$
Adaptive	$\alpha^0 = 0.5$ $u = 1.05$ $d = 0.8$ Normalized gradient

For the first initial point, $x^0 = \begin{bmatrix} -1.5 & 2 \end{bmatrix}^T$, the performance can be seen in Figure 2.12. As in the preliminary results from Section 2.3.3, all algorithms were able to converge to a local maximum. However, the gradient descent algorithm and its momentum versions required fewer iterations to meet the stopping criteria compared to the adaptive algorithm. It is noteworthy that, when compared to the similar test in Figure 2.14, the hybrid algorithms required fewer iterations (around 25 iterations compared to around 40 iterations) than their non-hybrid counterparts. This improvement is due to the hybrid algorithm's ability to use the initial faster convergence rates from the adaptive algorithm, as well as the automatic adjustment of algorithm parameters using the oracle information. In fact, one of the objectives of designing the hybrid algorithm was to transition from the global to the local algorithm (i.e., from the adaptive - "A" - to a gradient descent algorithm - "GD", "HB", "NE", "TM") when the adaptive algorithm started oscillating around the local maximum. The trajectory of the gradient descent algorithm throughout its iterations is illustrated in Figure 2.14(a), which is similar for all the tested algorithms. It is important to highlight that the transition from the global to the local algorithm occurs very close to the local maximum of the Gaussian function.

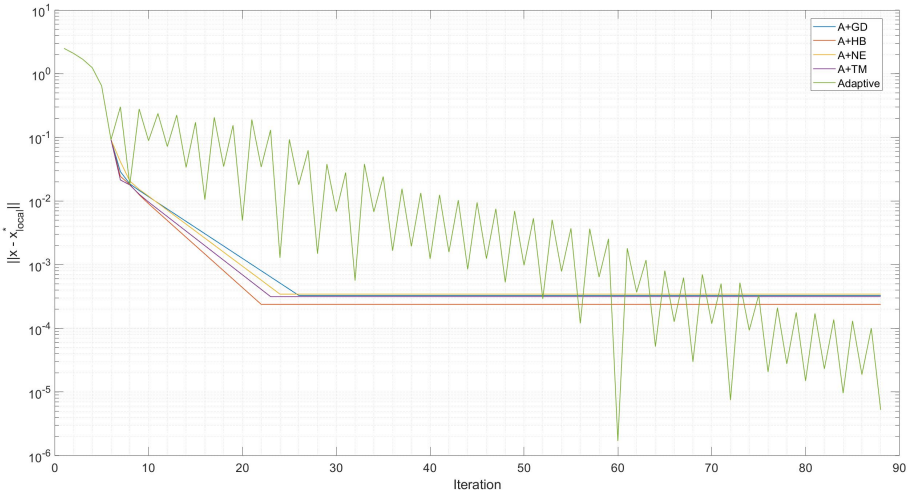


Figure 2.12: Test 1 - Distance to the maximum in each iteration using initialization 1

Regarding initialization 2, $x^0 = \begin{bmatrix} -5 & -5 \end{bmatrix}^T$, an important improvement can be observed in Figure 2.13. The gradient descent algorithm and its momentum versions are now able to converge to a local maximum when used as local strategies of the hybrid algorithm. This improvement can be attributed to the use of the adaptive step to overcome the plateau region and the oracle parameters to properly converge to the local maximum when in close proximity.

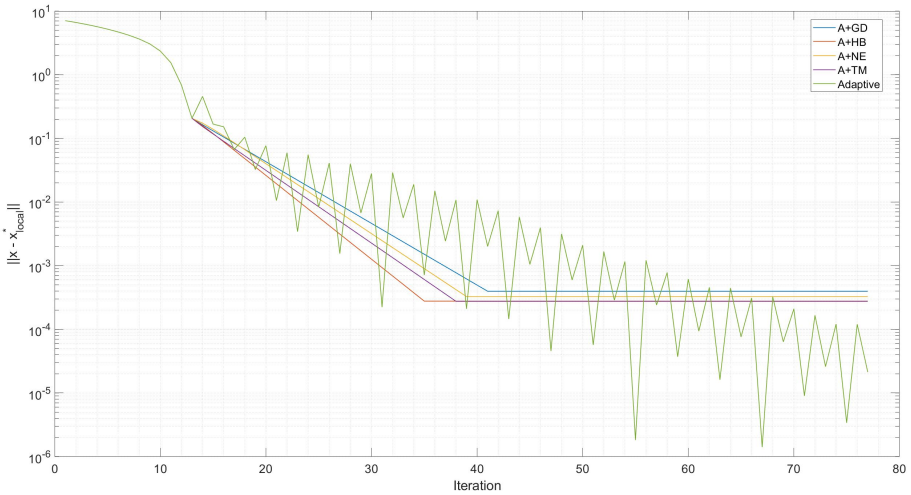
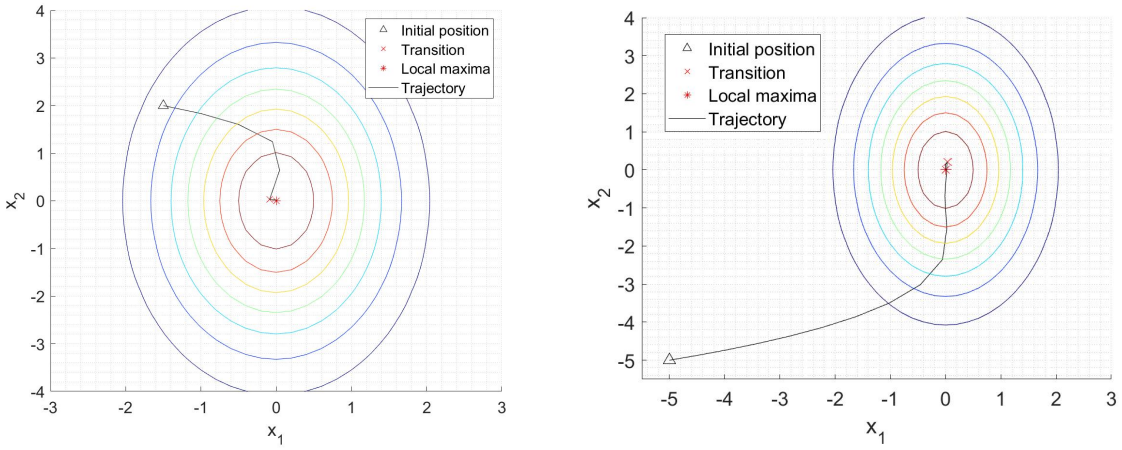


Figure 2.13: Test 1 - Distance to the maximum in each iteration using initialization 2

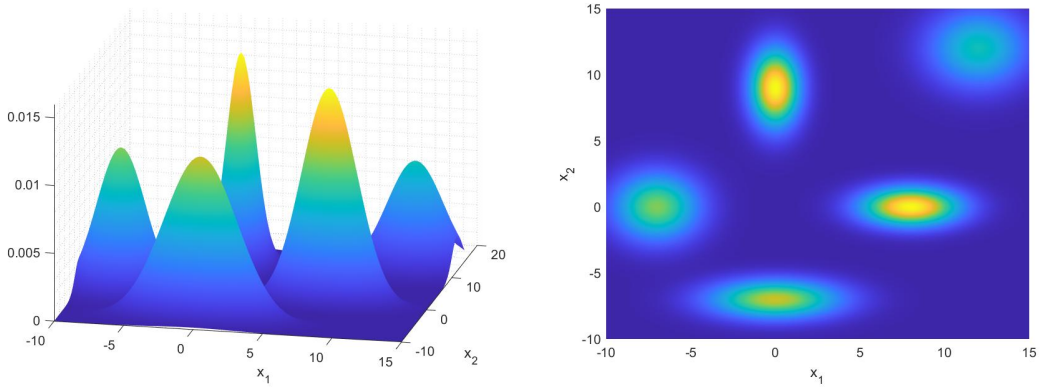


(a) Initialization 1 **(b)** Initialization 2

Figure 2.14: Test 1 - Trajectory created by the hybrid algorithm with gradient descent with initialization 1 and initialization 2

2.5.2 Test 2: Gaussian mixture with $K = 5$ far Gaussians

In this experiment, we evaluate the performance of each algorithm using a GM function with five non-overlapping Gaussians. As shown in Figure 2.15, the four local maximums of this function are separated and its non-plateau regions do not overlap. We use this function to assess whether our oracle can suggest appropriate L and m constants for each algorithm to adapt and converge regardless of the initial position.



(a) 3 dimensional view **(b)** 2 dimensional view

Figure 2.15: GM function used in Test 2

To achieve this, we employ four different initial positions, illustrated in Figure 2.16. The first three

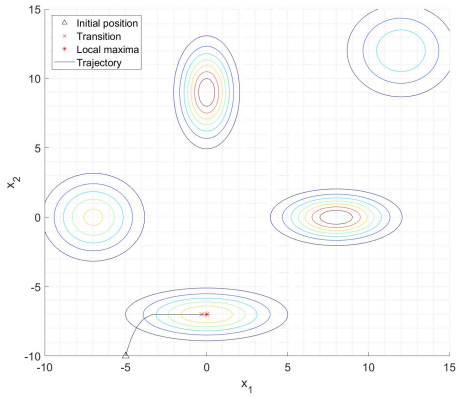
initializations are used to observe how the iteration typically converges to local maxima, which usually is in the direction of the Gaussian valley. The fourth initialization is used to force the iteration to approach a local maximum via the most challenging path, which is perpendicular to the direction of the valley where the gradients have the highest magnitude.

The algorithm parameters shown in Table 2.4 were obtained using the oracle, which outputted the L constant from the Gaussian centered at $\mu = \begin{bmatrix} 8 & 0 \end{bmatrix}^T$ and the m constant from the Gaussian centered at $\mu = \begin{bmatrix} 0 & -7 \end{bmatrix}^T$. Since the values of L and m for the other Gaussians fall within this range, the algorithms can successfully converge to all local maxima without diverging, as opposed to the preliminary results discussed in Section 2.3.3.

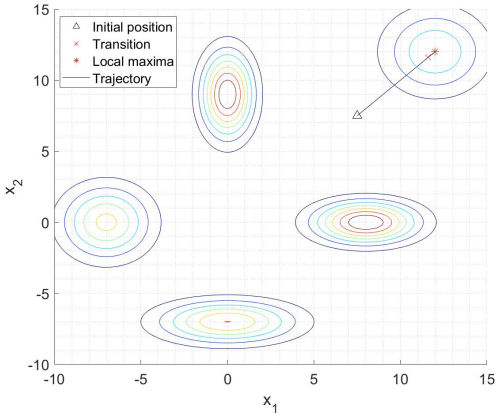
Table 2.4: Test 2 - Parameters generated using the oracle L and m constants and Table 2.1

Method	Parameters	
Gradient Descent (GD)	$\alpha = 113.4159$	
Nesterov's (NE)	$\alpha = 80.8649$	$\beta = 0.4569$
Heavy Ball (HB)	$\alpha = 142.3769$	$\beta = 0.2554$
Triple Momentum (TM)	$\alpha = 105.0160$	$\beta = 0.3393$
	$\gamma = 0.2030$	$\delta = 0.8207$
Adaptive	$\alpha^0 = 0.5$	$u = 1.05$
	$d = 0.8$	Normalized gradient

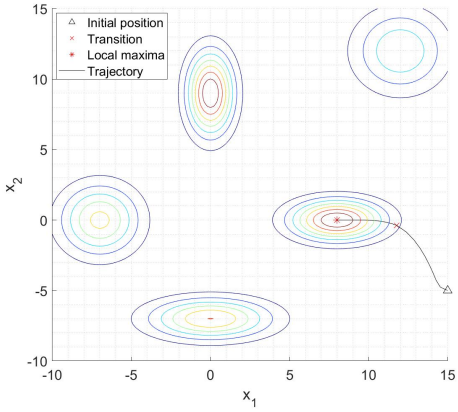
Figures 2.17 and 2.18 show the results for initialization 2 and initialization 4, respectively. In both cases, the momentum versions of the gradient descent algorithm (i.e, "A+HB", "A+NE", and "A+TM") outperform the gradient descent (i.e, "A+GD") and the adaptive algorithm (i.e., "Adaptive"). For initialization 4, some algorithms exhibit oscillation around the local maximum, which could be due to the parameters not being fully adjusted to this particular Gaussian.



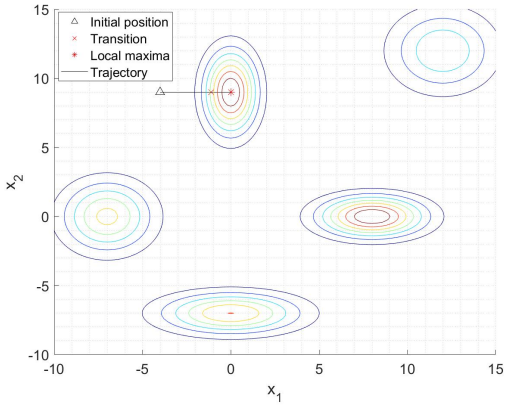
(a) Initialization 1



(b) Initialization 2



(c) Initialization 3



(d) Initialization 4

Figure 2.16: Test 2 - Trajectory created by the hybrid algorithm with gradient descent with four initializations

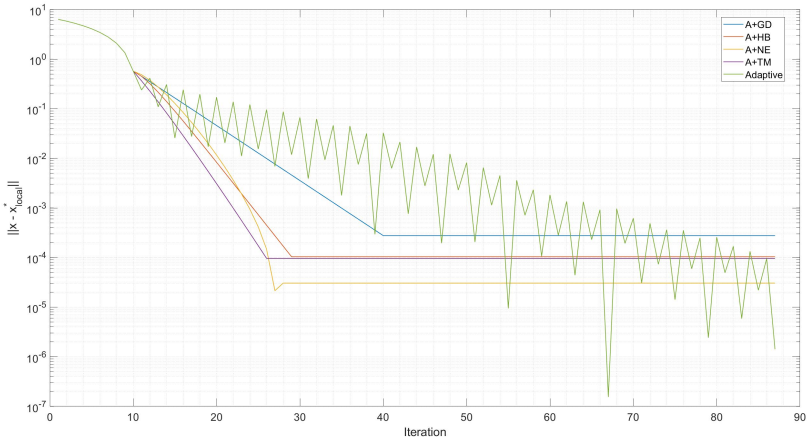


Figure 2.17: Test 2 - Distance to the maximum in each iteration using initialization 2

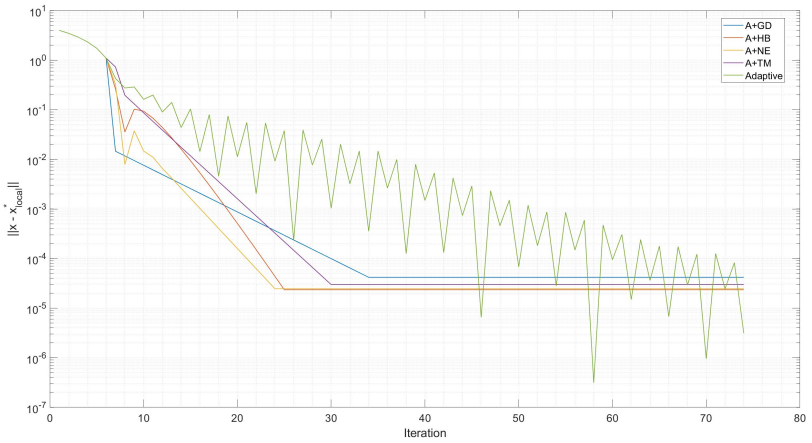


Figure 2.18: Test 2 - Distance to the maximum in each iteration using initialization 4

2.5.3 Test 3: Gaussian mixture with $K = 6$ near Gaussians

In this experiment, we evaluate the performance of each algorithm using a GM function with 6 overlapping Gaussians, which has already been introduced in Figures 2.2 and 2.8. 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 parameters for each algorithm are provided in Table 2.5.

Table 2.5: Test 3 - Parameters generated using the oracle L and m constants and Table 2.1

Method	Parameters	
Gradient Descent (GD)	$\alpha = 216.7547$	
Nesterov's (NE)	$\alpha = 153.8259$	$\beta = 0.4714$
Heavy Ball (HB)	$\alpha = 275.2511$	$\beta = 0.2699$
Triple Momentum (TM)	$\alpha = 200.7310$	$\beta = 0.3552$
	$\gamma = 0.2110$	$\delta = 0.8781$
Adaptive	$\alpha^0 = 0.5$	$u = 1.05$
	$d = 0.8$	Normalized gradient

In this set of experiments, we evaluated the performance of each algorithm using 50 random initializations, as shown in Figure 2.20. 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 Figure 2.19. In addition to successfully converging to a local maximum in each initialization, the results indicate that the hybrid algorithm with gradient descent and its momentum versions outperform the adaptive algorithm. Among the gradient descent algorithms, the momentum versions also outperform the gradient descent. On average, the Heavy Ball algorithm appears to meet the stopping criteria earlier than the Nesterov and Triple Momentum versions, although more tests are needed for a definitive conclusion.

Overall, these tests confirm the effectiveness of our hybrid algorithm in converging to a local maximum, even when multiple Gaussians in the GM function overlap.

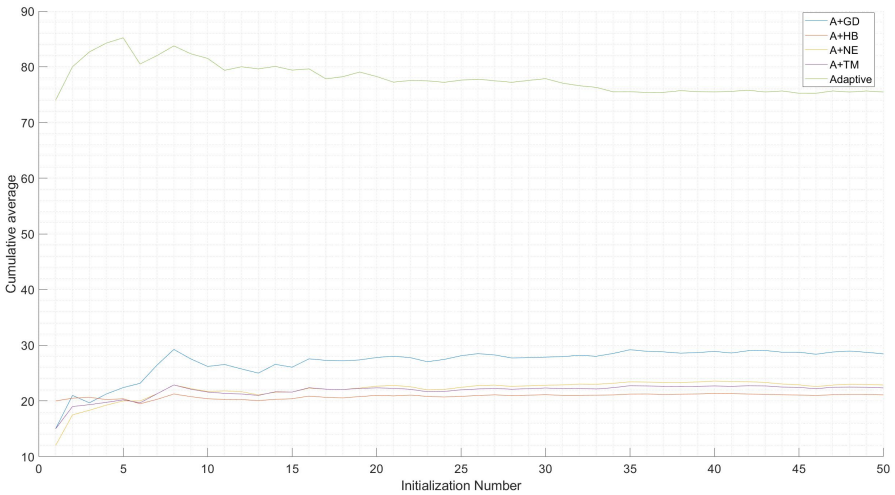


Figure 2.19: Test 3 - Cumulative average of iterations required by each algorithm to meet the stopping criteria

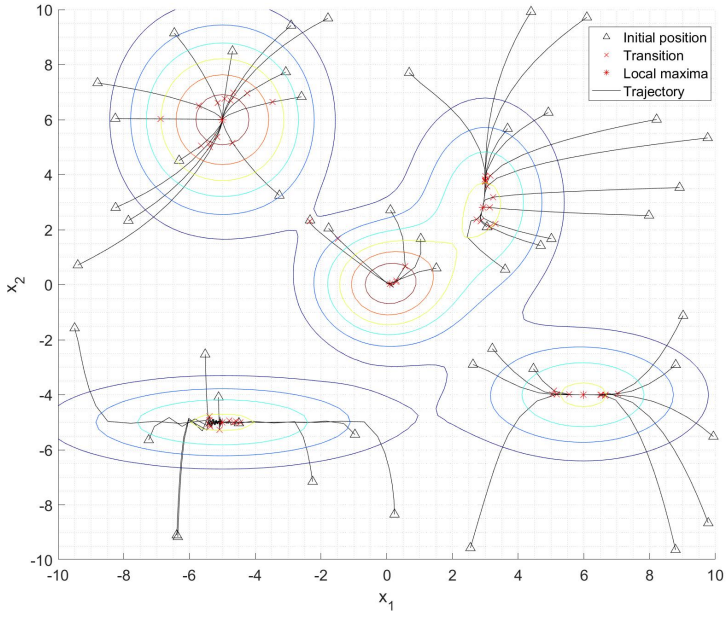


Figure 2.20: Test 3 - Initial estimate and hybrid algorithm with gradient descent trajectories for each one of the 50 random initializations

2.5.4 Test 4: Random Gaussian mixture with $K = 15$ Gaussians

In this experiment, the performance of each algorithm is evaluated using 10 randomly generated GM functions. Each map contains 15 random Gaussians with centers generated using $\mu_k = \begin{bmatrix}\mu_{x_1} & \mu_{x_2}\end{bmatrix}^T$, where $\mu_{x_1}, \mu_{x_2} \in [-10, 10]$. The covariance matrices for the Gaussians are generated using $\Sigma_k = \begin{bmatrix}\sigma_1^2 & 0 \\ 0 & \sigma_2^2\end{bmatrix}$, where $\sigma_1, \sigma_2 \in [1, 21]$, with an example of 2 of the 10 generated maps being shown in Figure 2.21 and Figure 2.22.

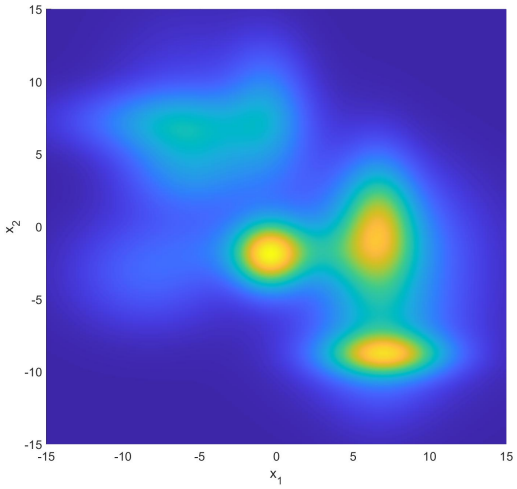
For each of the 10 generated maps, we generated 25 random initial conditions satisfying $\|x^0\|_\infty \leq 15$ and ran each algorithm, counting the number of iterations required to meet the stopping criteria. We report in Table 2.6 the average number of iterations for each algorithm on each map, as well as the overall average across all maps.

From this simulation, hybrid algorithm with momentum terms is a better solution since they can cope with almost plateau regions in some maps when several Gaussians accumulate within a given area, as is evident in map 6. This particular map has its Gaussians grouped in 4 groups that when mixed, create 4 main areas with local maximums located at $x = \begin{bmatrix}6.9 & 8.9\end{bmatrix}^T$, $x = \begin{bmatrix}6.1 & -0.9\end{bmatrix}^T$, $x = \begin{bmatrix}-0.5 & -1.8\end{bmatrix}^T$, $x = \begin{bmatrix}-2.8 & 7\end{bmatrix}^T$ and $x = \begin{bmatrix}-6 & 5.8\end{bmatrix}^T$ (refer to Figures 2.21(a) and 2.21(b)). The addition of an isolated Gaussian centered at $\mu_k = \begin{bmatrix}-8.4 & -3.1\end{bmatrix}^T$ between these non-plateau areas with higher values leads to the formation of the almost plateau region or a saddle point. In a situation like this, momentum algorithms are able to use the momentum generated for previous iterations, for example, the momentum

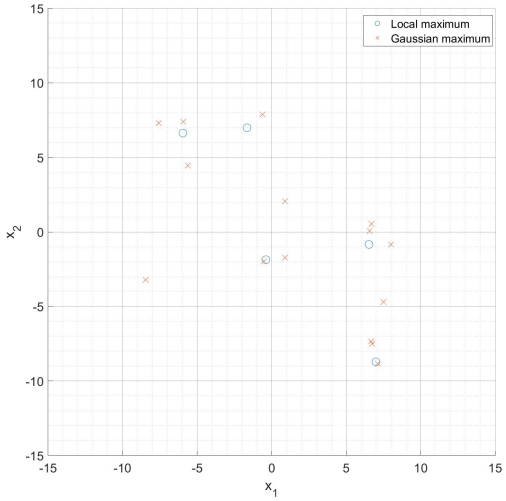
acquired during the transition from the plateau to the almost plateau region, to overcome the decreased gradients.

Table 2.6: Test 4 - Average iterations to meet stopping criteria for the hybrid algorithm using different local methods. The last column shows the overall algorithm average

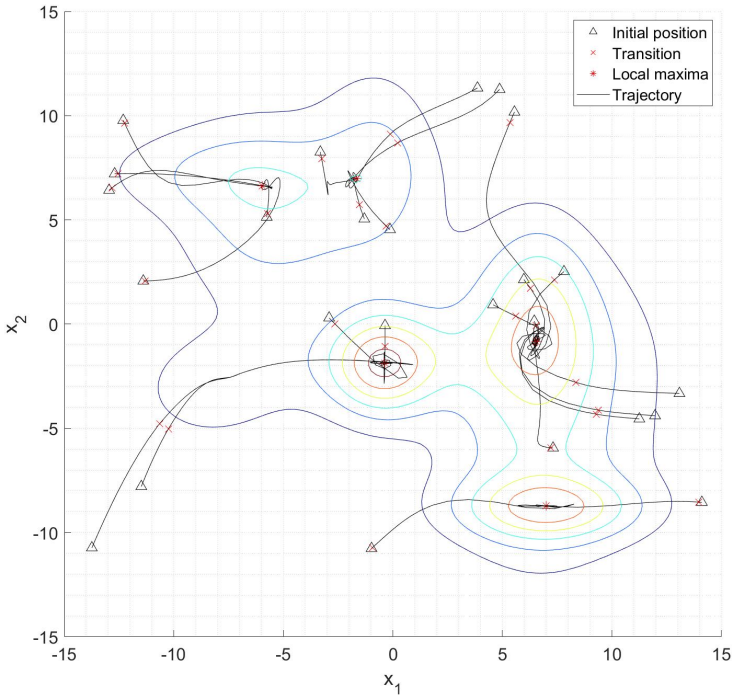
Method	Map 1	Map 2	Map 3	Map 4	Map 5	Map 6	Map 7	Map 8	Map 9	Map 10	Average
Gradient Descent	181	85	97	97	191	125	71	175	66	49	113.7
Nesterov's	62	53	54	59	60	60	39	58	30	34	50.9
Heavy Ball	81	57	61	64	75	67	40	74	37	35	59.1
Triple Momentum	74	55	58	64	67	64	39	66	33	33	55.3
Adaptive	71	81	75	85	72	74	75	71	78	77	75.9



(a) 2 dimensional upper view

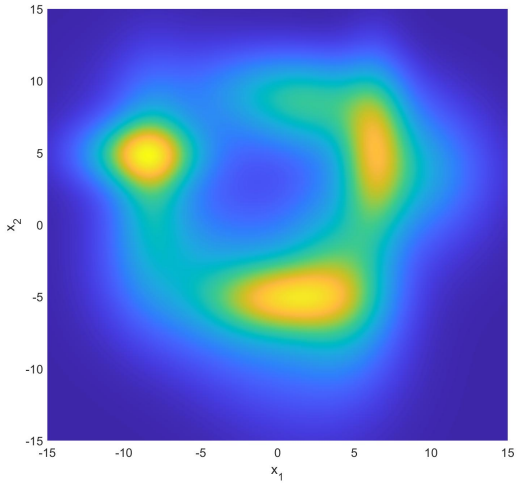


(b) Locations of the 15 random Gaussians' means and location of the local maximums of the GM function

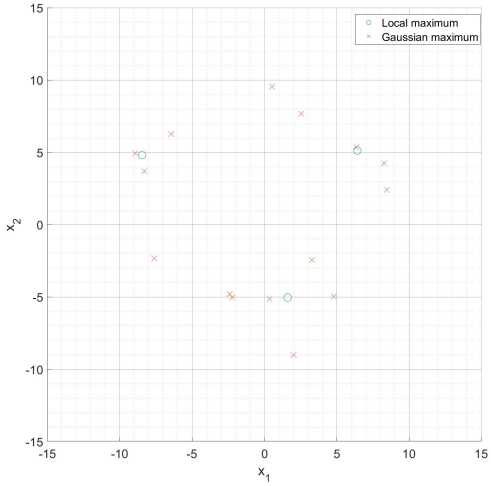


(c) Initial estimate and hybrid algorithm with Heavy Ball trajectories for each one of the 25 random initializations

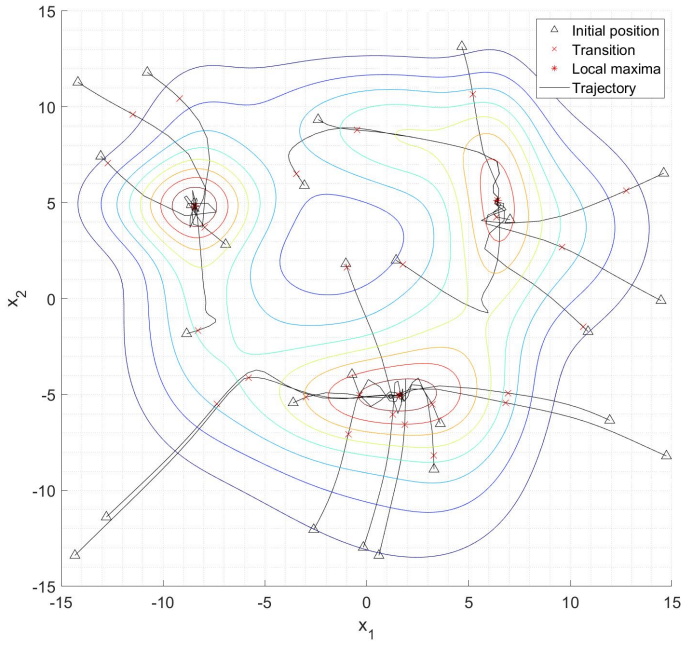
Figure 2.21: Test 4 - Random map 6



(a) 2 dimensional upper view



(b) Locations of the 15 random Gaussians' means and location of the local maximums of the GM function



(c) Initial estimate and hybrid algorithm with Heavy Ball trajectories for each one of the 25 random initializations

Figure 2.22: Test 4 - Random map 7

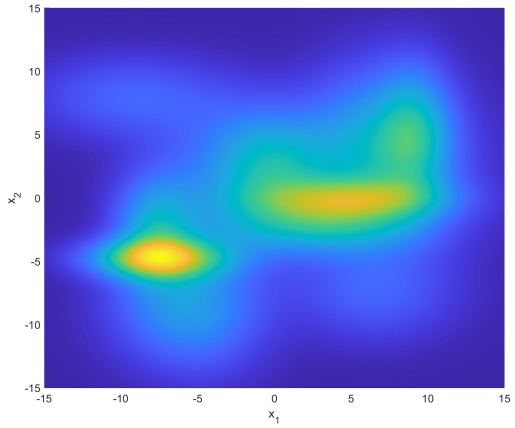
2.5.5 Test 5: Hybrid algorithm with adaptive parameters

To fully explore the possibilities for the proposed Hybrid algorithm in the selection of a local method, we follow a strategy suggested by Almeida and Silva in their work [33]. In this experiment, we use the same protocol described in the previous section, where we generated 10 random maps and used 25 initial estimations for each map. However, we now incorporate adaptive steps and automatically tuned algorithms simultaneously into the algorithm. To achieve this, we apply an updated term to the α parameter after each iteration, as defined in Equation (2.13), for both the gradient descent and the three momentum versions. The initial value of α is still the one obtained using the oracle's output and Table 2.1. By incorporating this strategy, the local algorithms can easily overcome almost plateau regions, which are areas where the gradient always points in the same direction, leading to increasing steps. However, this approach also has a disadvantage. The increased step size may result in oscillations when the estimate approaches the local maximum, thereby requiring more iterations to meet the stopping criteria. The red trajectory in Figure 2.23 represents a pathological case of this behavior. In fact, for all 10 randomly generated maps, this trajectory had the highest oscillations recorded.

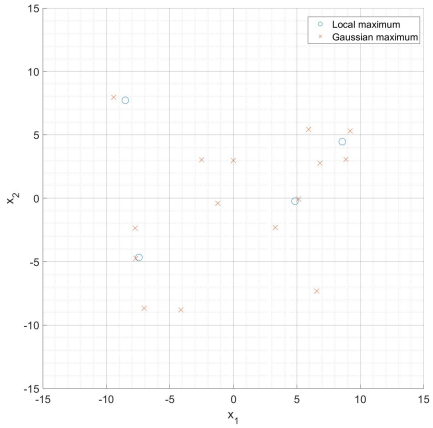
The results presented in Table 2.7 demonstrate an overall improvement in the performance of all hybrid algorithms using "GD", "HB", "NE" and "TM" as local algorithms, particularly for the gradient descent "GD". Moreover, the momentum algorithms, particularly the Heavy Ball algorithm, exhibited a significant improvement in the average number of iterations required for convergence, despite the expected oscillation effects. Although the decrease in the number of iterations observed for the Nesterov algorithm may be due to statistical variations, we can conclude that the introduction of the adaptive α in this chapter does not negatively impact the performance of the hybrid algorithms.

Table 2.7: Test 5 - Average iterations to meet stopping criteria for the hybrid algorithm using different local methods. The last column shows the overall algorithm average

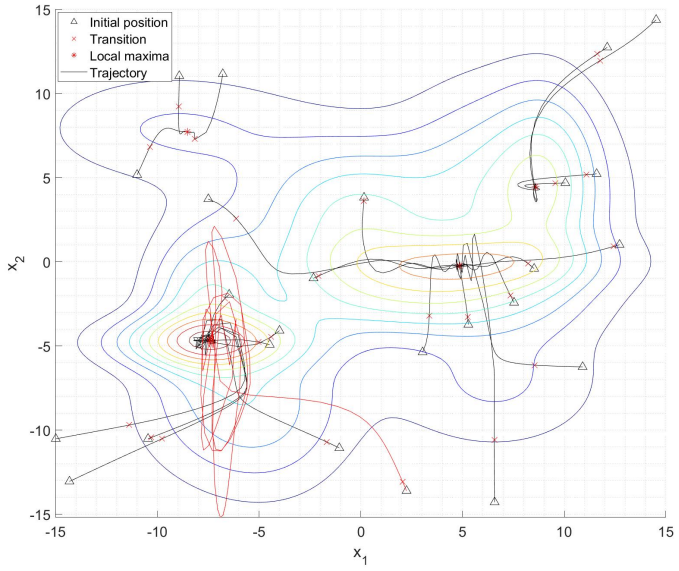
Local method	Map 1	Map 2	Map 3	Map 4	Map 5	Map 6	Map 7	Map 8	Map 9	Map 10	Average
Gradient Descent	49	48	47	54	40	42	60	63	46	44	49.3
Nesterov's	45	55	49	53	42	45	57	58	57	42	50.3
Heavy Ball	48	52	51	55	44	46	57	59	54	49	51.5
Triple Momentum	48	57	50	56	44	47	59	61	58	48	52.8
Adaptive	79	78	82	78	81	79	73	78	76	78	78.2



(a) 2 dimensional upper view



(b) Locations of the 15 random Gaussians' means and location of the local maximums of the GM function



(c) Initial estimate and hybrid algorithm with Heavy Ball trajectories for each one of the 25 random initializations

Figure 2.23: Test 1 - Random map 7

2.6 Conclusion

In conclusion, this chapter provided a comprehensive analysis and investigation of optimization techniques for non-convex functions modeled by a GM function. The main contribution of this study was the

proposal of a hybrid algorithm that effectively combines the strengths of two distinct optimization algorithms. By leveraging the unique characteristics of each algorithm, the hybrid approach offers improved performance and adaptability throughout the optimization process.

In Section 2.3 we defined the optimization problem and the objective function, and in Section 2.3.3 we provided an illustrative example to demonstrate the challenges posed by a GM function. Particularly, this example revealed:

- **Presence of plateau regions**: The search space for GM functions often contains plateau regions, which present a significant obstacle for optimization algorithms with static parameters. These plateaus create a flat landscape where traditional optimization methods struggle to progress. Overcoming plateau regions require higher parameters then the ones selectede for good convergence.
- **Difficulty of manual tuning**: Manual parameter tuning is a laborious and time-consuming process. The presence of plateau regions and the wide range of Gaussian functions that can be incorporated into a GM model, pose significant challenges in achieving effective manual tuning of algorithm parameters. Consequently, there is a need for a systematic approach that can select the parameters.

Our hybrid algorithm, proposed in Section 2.4 is able to overcome these challenges. To assess its performance, we conducted a series of five tests, comparing four variations of the hybrid algorithm against the gradient descent with adaptive step, since it was the only capable of converging with some hybrid behavior.

In the first test (Section 2.5.1), we replicated the setup used in the illustrative example. All four versions of the hybrid algorithm successfully converged to a local maximum, demonstrating the effectiveness of our oracle in providing valid L-Lipschitz and m-strongly convex constants for a function with a single Gaussian. Furthermore, the hybrid algorithm was able to switch from the global to the local algorithm at the right time to avoid oscillation or divergence.

In the second test (Section 2.5.2) we created a function with 5 Gaussian with different shapes. The objective of this test was to illustrate that our hybrid algorithm is able to converge by switching to the local algorithm even with several Gaussians. This example highlighted that when the oracle provides valid L and m constants for the function being optimized, our algorithm is able to adapt its parameters effectively, regardless of the presence of multiple Gaussians.

In the third test (Section 2.5.3) we designed a function composed of 6 Gaussians whith overlaping non-plateau regions. The overlapping regions could pose difficulties as the parameters have been computed assuming separation of the Gaussians. Despite this challenge, our algorithm successfully switched to the local algorithm and converged to a local maximum of the function when initialized at 50 distinct random locations. Moreover, all the hybrid algorithms versions required fewer than 50% of the iterations compared to the adaptive gradient descent.

The fourth and fifth tests (Sections 2.5.4 and 2.5.5) serve to present the behavior for random setups and used 10 random GM functions, each consisting of 15 Gaussians, and initialized the algorithms in 25 random location. Regarding test 4, the hybrid algorithm ”A+GD” performed the worst, followed by the ”Adaptive” algorithm. The hybrid algorithms ”A+HB”, ”A+NE”, and ”A*TM” showed the best performance, with similar results among them. After identifying the possible reasons for the poor performance of the ”A+GD” approach, we conducted test number 5, where the local algorithm used the adaptive step principle. This strategy proved to be advantageous for the gradient descent algorithm and did not worsen the performance of the momentum term version. However, further tests should be performed to assess the oscillation effect introduced by the adaptive step principle in the local algorithm with momentum terms.

3

UAV Path Planning and Path Following

Contents

In this chapter, we explore an approach for solving the problem of optimal path generation from multiple uncertainty maps, as discussed in Section 1.3. Due to the significant computational resources and time required to solve this complex problem, we propose an alternative sub-optimal solution that is based on an empirical intuition. The system is divided into two distinct but interconnected component to drive the autonomous vehicle trough the most uncertainty areas of the map to conduct a surveillance mission in order to reduce the uncertainty regarding a possible wildfire in that region.

Additionally, this chapter focuses on path planning and path following strategies for a quadrotor Unmanned Aerial Vehicle (UAV), which are essential components for effective surveillance and data collection. We evaluate these strategies in a simulated environment provided by Gazebo and Robot Operating System (ROS) tools.

For that, Section 3.1 provides a review of relevant background theory to facilitate understanding the described solution in following sections. The crucial role of aerial vehicle and the growing interest in the development of flight arenas for testing such vehicle will be discussed. Section 3.2 outlines our specific problem statement as a particular case of the original problem, while Section 3.3 proposes several approaches that enables out system to reach its goal. Finnaly, Section 3.4 analyze our complete system, proving that the autonomous vehicle is able to follow the desired path with a low magnitude tracking error, while flying trough the most pertinent areas.

3.1 Background Theory

This section provides essential background theory that is useful for understanding both the problem statement and the proposed solution. We will start by discussing the fundamental role of UAVs and the increasing interest in creating flight arenas for UAV testing purposes. Next, we explore the concept of UAV control, emphasizing the common challenges faced and the strategies employed, with a particular focus on the problem of path following. Lastly, we study the parametrization of curves using B-splines and spiral curves, which serves as a key component in guiding the UAV along its intended path. We draw special attention on uniform cubic B-splines and Constant Linear Velocity spirals.

3.1.1 Unmanned Aerial Vehicles

UAVs are aircraft that can operate without a human pilot on board. They can be controlled remotely or through autonomous algorithms, making them useful for a wide range of applications, including surveillance, reconnaissance, and delivery. There are various types of UAVs available, such as fixed-wing aircraft, rotary-wing aircraft, and hybrid aircraft. However, in recent years, rotary-wing quadrotors have become increasingly popular among researchers due to their agility, versatility, and ease of maintenance. For further details on UAVs classificaton refer to [41].

One of the most significant advantages of quadrotors is their maneuverability. Unlike fixed-wing aircraft, which require a runway for takeoff and landing, quadrotors can take off and land vertically and hover in place. This makes them ideal for operations in confined spaces or areas where a runway is not available.

Despite their benefits, quadrotors do have some limitations, including their limited flight time and limited payload capacity. Due to their small size and the power requirements of the rotors, the flight time of a quadrotor is usually limited to a few minutes. This can make them unsuitable for applications that require long-duration flights, such as forest fire surveillance.

The use of quadrotor UAVs in this work was due to their simplicity and the availability of existing tools already developed for the Institute for Systems and Robotics (ISR)'s flight arena. While long-duration flights may be necessary for forest fire surveillance, the agility and ease of maintenance offered by quadrotors can be beneficial in other aspects of the project, mainly during the prototype phase. Moreover, utilizing the ISR's flight arena tools can facilitate the simulation and real testing of the developed algorithms.

3.1.2 Flight Arena Technologies

Recently, there has been an increasing interest in developing flight arenas for UAV) that provide a controlled environment for testing UAVs and their algorithms. The ISR has designed its own flight arena that is tailored for quadrotor UAVs, which includes an optical motion capture system to provide accurate vehicle position and attitude ground-truth, and a set of offboard computers to manage communication between systems and run user programs. Oliveira's work in [1, 42] details the complete design and implementation of the IST flight arena architecture, which follows a modular approach combining the PX4 autopilot, a vehicle, a motion capture system, and a ground computer module.

The figures shown in Figures 3.1 and 3.2 illustrate the architecture of the system for both real and simulated flight scenarios, respectively. This architecture adopts a modular approach that combines the following components:

- PX4 Autopilot: This is an open-source autopilot system specifically designed for UAVs. It provides flight control algorithms, sensor drivers, and estimation algorithms for different types of UAVs, and acts as an interface between the drone and higher level algorithms. PX4 can run on a real drone using its Hardware In The Loop capabilities or communicate with a vehicle simulator using the Software In The Loop mode.

- Vehicle: Depending on the simulation type, the vehicle can either be a real quadrotor or a simulation. To simulate the vehicle's behavior, the system uses the Gazebo simulator, an open-source 3D robot simulation software that allows developers to create and test complex robotic systems in a

virtual environment, including its dynamic and kinematics. Gazebo models the behavior of sensors and actuators using plugins. Furthermore, the MAVROS package is used to interface Gazebo with PX4, acting as a bridge between the two systems by translating messages between the lightweight MAVLink protocol used by PX4 and the ROS messaging format used by Gazebo.

- Motion capture system - In the fligth arena, the motion capture is done by an Optitrack sysyem, equiped with 8 infrared cameras to track the position and orientation of objects. For the simulation environment, Almeida developed a software program using the capabilities of the ROS middleware capable of listening to the Gazebo simulation's outputs and mimic the Optitrack system, deliveryng the pose of the vehicle to the user (in the ground computer model) and to the PX4.

- Ground computer module: This module consists of three models, two of which were developed in the base architecture of the flight arena using the MAVROS and MAVSDK libraries. The work presented in this chapter is included in the third module, referred to as "User code" in the figures. The input/telemetry module subscribes to the information published by the PX4 and the motion capture system, making this information available to the user in the ROS framework. The output/offboard module offers several methods that the user can call to send offboard commands and control references to the PX4.

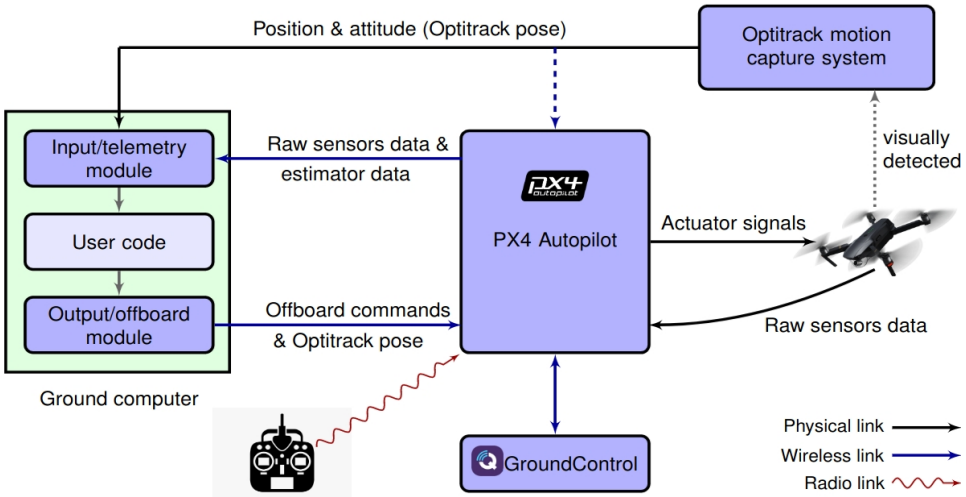


Figure 3.1: Architecture for the real environment of the ISR Flying Arena (Oliveira [1], 2021, Figure 2.3)

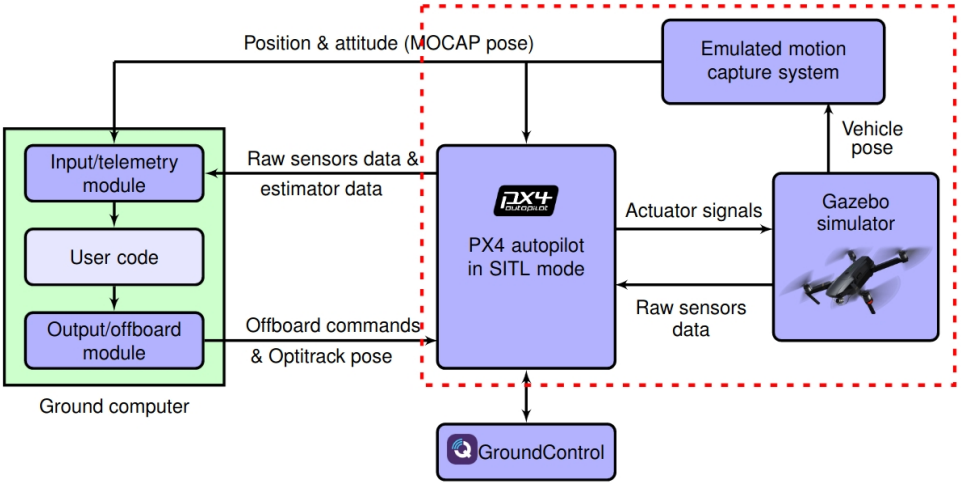


Figure 3.2: Architecture for the simulation environment of the ISR Flying Arena (Oliveira [1], 2021, Figure 2.4)

Almeida based the fundamental design and operational mechanism of its architecture in other testbeds, as the MIT RAVEN [43] and the ETH Zurich's Flying Machine Arena [44].

3.1.3 Quadrotor Control

Controlling a quadrotor requires using feedback control algorithms that adjust the vehicle's orientation, position, and velocity based on sensor measurements and control commands to achieve the desired actions. However, before designing the control algorithm, it is essential to create a mathematical model that accurately describes the quadrotor's dynamics and kinematics, using physical properties such as mass, moments of inertia, and aerodynamic coefficients. This model predicts how the vehicle behaves under different conditions and provides a basis for designing control algorithms that can stabilize it in various flight regimes such as hovering, altitude control, or trajectory tracking. Accurately modeling the quadrotor's dynamics may be crucial for effective control design, and many works have led to models that capture the vehicle's behavior accurately. For example, the models presented in [45, 46] define the quadrotor as a rigid body with six degrees of freedom and four control inputs. However these models can precisely describe the quadrotor's dynamics, they can be mathematically challenging and expensive to develop control laws. In practice, they are primarily useful for state estimation, where sensor data from devices like inertial measurement units or cameras is used to estimate the quadrotor's real position and attitude, and for inner-loops controllers, responsible for increasing the drone's stability while acting on the rotors of the vehicle and generating a set of forces and torques. Controllers for attitude and positioning are usually implemented in these inner-loops.

However, even these important tasks of estimation and low-level control can be designed using linearized models. These models involve approximating the nonlinear equations of motion using linear equations valid for small deviations from a nominal operating point, usually chosen to be the hover. Many

works have demonstrated that it is possible to control the quadrotor using linear control techniques such as Proportional Integrative Derivative controller (PID) [47] or Linear Quadratic Regulator (LQR) (LQR) strategies [48]. These linear control techniques are easier to design and implement compared to nonlinear controllers and have been shown to achieve satisfactory control performance for quadrotors.

One of the key advantages of PX4 is that it already comes equipped with a set of low-level controllers and estimators, including attitude and velocity controllers, as well as state estimators based on sensor fusion algorithms. These controllers and estimators have been developed and tested extensively, and they have been shown to work well in a wide range of conditions. As a result, this work focus its research in higher level controllers strategies and on creating outer loops for the quadrotor. Specifically, we aim to create a path following strategy that can run after the path planning algorithm, ensuring that the drone follows the desired route. As an outer-loop controller, our algorithm will generate references for the PX4 inner-loop controllers.

3.1.4 Path following

Path following can be summarized as a set of techniques that allow a vehicle to navigate along a predefined path with accuracy and smoothness. Unlike trajectory tracking techniques, which rely on time parameterization of the path, path following does not impose any specific timing constraints [49]. Instead, it focuses on accurately following a path defined by a series of waypoints, curves, or even complex geometric shapes without any time restriction.

Designing path following controllers encompasses various approaches that differ in complexity and the level of detail in the considered vehicle model. One approach involves utilizing a detailed model of the vehicle. By incorporating these specific characteristics, the algorithm can achieve precise control. Techniques following this approach often leverage concepts from Lyapunov theory [50] or employ model predictive control (MPC) [51]. Lyapunov-based approaches utilize the Lyapunov stability concept, aiming to establish the stability of the system by constructing a Lyapunov function that guarantees convergence to the desired path. While Lyapunov-based methods offer theoretical guarantees and robustness, they can involve complex control theory and advanced mathematical computations. Model predictive control (MPC) is another approach that utilizes the dynamic model of the vehicle to optimize future control inputs based on a cost function. By considering the vehicle's detailed model, MPC can generate control commands that explicitly account for the system's dynamics and constraints. However, MPC may also pose computational challenges due to the need for extensive numerical computations and real-time optimization.

In the context of this work, we develop within a framework that employs PX4 as an autopilot for inner-loop controllers. PX4 utilizes sophisticated control algorithms to stabilize the quadrotor. Therefore, for the design of the outer-loop controller, we will adopt a linear model. Linearized solutions approximate the

quadrotor's nonlinear dynamics by linearizing the system around a specific operating point or trajectory. One commonly used linearized model is the double integrator model, which considers the quadrotor's position and velocity as the only relevant states. By focusing on position and velocity control, the design process becomes more straightforward and computationally efficient.

3.1.5 Path parametrization

In the literature there are two main strategies to define and parameterize a curve or path:

- Implicit equations describe a relationship between variables without explicitly expressing one variable in terms of the others. They express this relationship between variables through an equation. For example, a curve $\mathbf{C}$ lying on xy plane has the form $f(x, y) = 0$. Implicit equations can represent a wide range of curves, surfaces, or higher-dimensional shapes. However, they do not provide explicit representations of the variables in terms of a parameter.

- Parametric equations define variables as functions of a parameter or set of parameters. Each variable is expressed independently in terms of a parameterizing variable γ. For example, in the context of a curve lying on the xy plane, a parametric equation has the form

$$\mathbf{C} = \begin{bmatrix} x(\gamma) \\ y(\gamma) \end{bmatrix}, \text{ with } \gamma \in [a, b], \tag{3.1}$$

  where $a, b \in \Re$, and $\mathbf{C} \in \Re^2$.

The use of parametric equations is particularly beneficial when generating a path for a quadrotor or any other robotic system. Parametric equations enable the generation of smooth and continuous paths, which are crucial for quadrotor motion planning. Additionally, they allow for the expression of bounds on the curve segments by defining bounds on the path parameter interval. Furthermore, extending a parametric curve to an N-dimensional space is easy, requiring only the addition of an extra set of coordinates that depend on the path parameter γ.

3.1.5.A B-Splines Curves

This work utilizes Bézier curves to generate a smooth and continuous path for the quadrotor. Bézier curves are a widely used type of parametric curve in computer graphics, computer-aided design (CAD), and other fields. They are defined by a set of control points that determine their shape and degree. By combining multiple Bézier curves, a B-spline can be formed. This approach allows for the creation of complex paths by joining together multiple low-degree Bézier curves instead of using a single high-degree curve.

A unidimensional order $k+1$ B-spline is a piecewise polynomial function, formed by joining several pieces of polynomials of degree k. A general B-spline curve of order $k+1$ and defined by $n+1$ control points is formed by joining several pieces of polynomials of degree k and consists of $n-k+1$ Bézier curves. It can be interpreted as a linear combination of each control by:

$$\mathbf{C}(\gamma) = \sum_{i=0}^{n} B_{i,k}(\gamma) P_i \tag{3.2}$$

where $P_i, i = 0, 1, ..., n$ are the set of control points and $B_{i,k}(\gamma)$ the basis functions of fixed degree.

The basis functions are generated recursively using

$$B_{i,0} = \begin{cases} 1 & \text{if } \gamma_i \leq \gamma \leq \gamma_{i+1} \\ 0 & \text{otherwise} \end{cases} \tag{3.3}$$

where

$$B_{i,k}(\gamma) = \frac{\gamma - \gamma_i}{\gamma_{i+j} - \gamma_i} B_{i,j-1}(\gamma) + \frac{\gamma_{i+j+1} - \gamma}{\gamma_{i+j+1} - \gamma_{i+1}} B_{i+1,j-1}(\gamma) \tag{3.4}$$

The values γ_i belong to a knot vector defined as $\Gamma = [\gamma_0, ..., \gamma_m]$, such that $\gamma \in [\gamma_0, \gamma_m]$. The number of knots depends on the degree of the curve and the number of control points, such that $m = k + n + 1$. Apart from the control points, the knot vector plays a crucial role in defining the shape and behavior of a B-spline curve or surface. For that, one can define its multiplicity and spacing:

- The multiplicity of a knot refers to the number of times it is repeated in the knot vector. Higher multiplicities indicate that the corresponding control points have a stronger influence on the shape of the B-spline in that region. Increasing the multiplicity of a knot reduces the continuity of the curve at that knot. Specifically, the curve is $(k-p)$ times continuously differentiable (has $\mathcal{C}^{k-p}$ continuity) at a knot with multiplicity p, with $(p \leq (k+1))$ [52].

  A B-spline is said to be clamped if the multiplicity of the first and last values of the knot vector is $(k+1)$. That is:

  $$\Gamma = [\underbrace{\gamma_0, \gamma_1, ..., \gamma_k}_{k+1 \text{ repeated knots}}, \underbrace{\gamma_{k+1}, ..., \gamma_n}_{n-k \text{ internal knots}}, \underbrace{\gamma_{n+1}, ..., \gamma_{n+k+1}}_{k+1 \text{ repeated knots}}] \tag{3.5}$$

  A first and last positions of a clamped B-spline coincide with its first and last control points.

- The spacing between adjacent knots determines the influence of the corresponding control points. Closer knots result in a more localized effect, while widely spaced knots lead to a smoother, more gradual influence. A B-spline is said to be uniform if its knots are equidistant, i.e, the knots are given by

  $$\gamma_i = (i-1)\Delta, i = 1, ..., m \tag{3.6}$$

3.1.5.B Uniform Cubic B-Spline Curves

In this section we explore the application of the previous section to a specific case: a uniform cubic B-spline, i.e., with $k = 3$. Every point of a uniform B-spline has multiplicity 1, therefore, the curve has $\mathcal{C}^2$ continuity. Specifically:

- Each segment (i.e., each Bezier curve) is continuous. Furthermore, the final point of the segment i has the same coordinates as the first point of the segment $i + 1$ - C^0 continuity
- The first derivative of each segment is continuous. Furthermore, the first derivative at the end of segment i is the same as the first derivative at the first point of segment $i+1$. In other words, there is no abrupt change in slope - $\mathcal{C}^1$ continuity
- The final point of segment i has the same coordinates as the first point of segment $i + 1$. In other words, there are no abrupt changes of polarity - $\mathcal{C}^2$ continuity

Let's now consider that the uniform cubic B-spline is composed only of 4 control points (n=3), therefore with only 1 segment. The knot vector is therefore defined as:

$$\Gamma = [0, 1, 2, 3, 4, 5, 6, 7]^T \tag{3.7}$$

Using Equations (3.2) to (3.4) we can now define a unidimensional, uniform cubic spline as:

$$\begin{aligned}
\mathbf{C}(\gamma) &= B_{0,3}(\gamma)P_0 + B_{1,3}(\gamma)P_1 + B_{2,3}(\gamma)P_2 + B_{3,3}(\gamma)P_3 && (3.8)\\
&= \frac{1}{6}[(1-\gamma)^3 P_0 + (3\gamma^3 - 6\gamma^2 + 4)P_1 + (-3\gamma^3 + 3\gamma^2 + 3\gamma + 1)P_2 + \gamma^3 P_3] && (3.9)\\
&= \frac{1}{6}[(-\gamma^3 + 3\gamma^2 - 3\gamma + 1)P_0 + (3\gamma^3 - 6\gamma^2 + 4)P_1 + (-3\gamma^3 + 3\gamma^2 + 3\gamma + 1)P_2 + \gamma^3 P_3] && (3.10)
\end{aligned}$$

where $\gamma \in [0, 1[$. Equation (3.10) can be rearranged in the following matrix form:

$$\mathbf{C}(\gamma) = \frac{1}{6}\begin{bmatrix}\gamma^3 & \gamma^2 & \gamma & 1\end{bmatrix}\begin{bmatrix}-1 & 3 & -3 & 1\\ 3 & -6 & 3 & 0\\ -3 & 0 & 3 & 0\\ 1 & 4 & 1 & 0\end{bmatrix}\begin{bmatrix}P_0\\ P_1\\ P_2\\ P_3\end{bmatrix} \tag{3.11}$$

Using this notation, it is possible to extend it to a general case of $n+1$ points and $n-k+1$ segments. For that, let's assume $\gamma \in [0, n-k+1]$, and define $i \coloneqq \lfloor x \rfloor$, such that $\gamma - i \in [0, 1[$. To compute the current curve position:

$$\mathbf{C}(\gamma) = \frac{1}{6}\begin{bmatrix}(\gamma-i)^3 & (\gamma-i)^2 & (\gamma-i) & 1\end{bmatrix}\begin{bmatrix}-1 & 3 & -3 & 1\\ 3 & -6 & 3 & 0\\ -3 & 0 & 3 & 0\\ 1 & 4 & 1 & 0\end{bmatrix}\begin{bmatrix}P_i\\ P_{i+1}\\ P_{i+2}\\ P_{i+3}\end{bmatrix} \tag{3.12}$$

This representation offers significant advantages by allowing us to store only the essential information required for computing the current segment. It eliminates the need to expand the knot vector to include $n + k + 2$ knot values and removes the need to retain the previous control points that are no longer needed. Moreover, it facilitates the pre-computation of the required basis functions, eliminating the need for recursive computation when drawing a new segment. Appendix A provides a detailed explanation of the computations presented above.

When dealing with 2-dimensional paths, like the ones in this work,Equation (3.12) can be applied to each dimension independently.

Equation (3.12) is also a useful approach to compute the first derivative of the path in order to the parametrized variable. The only part of the definition of the B-spline that depends on γ is the vector $\begin{bmatrix}(\gamma-i)^3 & (\gamma-i)^2 & (\gamma-i) & 1\end{bmatrix}$ which has trivial derivatives.

$$\frac{\partial \mathbf{C}}{\partial \gamma}(\gamma) = \frac{1}{6}\begin{bmatrix}3(\gamma-i)^2 & 2(\gamma-i) & 1 & 0\end{bmatrix}\begin{bmatrix}-1 & 3 & -3 & 1\\ 3 & -6 & 3 & 0\\ -3 & 0 & 3 & 0\\ 1 & 4 & 1 & 0\end{bmatrix}\begin{bmatrix}P_i\\ P_{i+1}\\ P_{i+2}\\ P_{i+3}\end{bmatrix} \tag{3.13}$$

The same process can lead us to the second derivative of the path in order to the parametrized variable:

$$\frac{\partial^2 \mathbf{C}}{\partial \gamma^2}(\gamma) = \frac{1}{6}\begin{bmatrix}6(\gamma-i) & 2 & 0 & 0\end{bmatrix}\begin{bmatrix}-1 & 3 & -3 & 1\\ 3 & -6 & 3 & 0\\ -3 & 0 & 3 & 0\\ 1 & 4 & 1 & 0\end{bmatrix}\begin{bmatrix}P_i\\ P_{i+1}\\ P_{i+2}\\ P_{i+3}\end{bmatrix} \tag{3.14}$$

Figure 3.3 presents an example of a bi-dimensional uniform cubic B-spline, with 13 control points. A uniform B-spline is not a clamped B-spline, i.e., the first and last positions of the curve do not necessarily coincide with its first and last control points. Although, in this example, we have repeated the first and last control points 4 times, forcing the curve to be clamped. Figure 3.4 shows the first and second derivatives in each dimension, providing additional insights into the behavior of the curve.

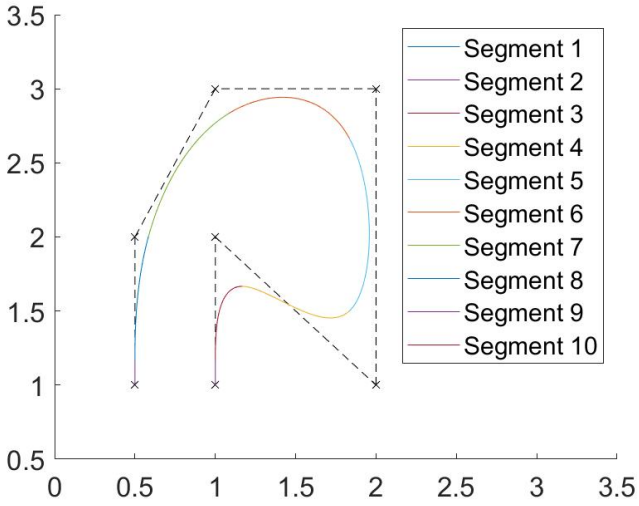
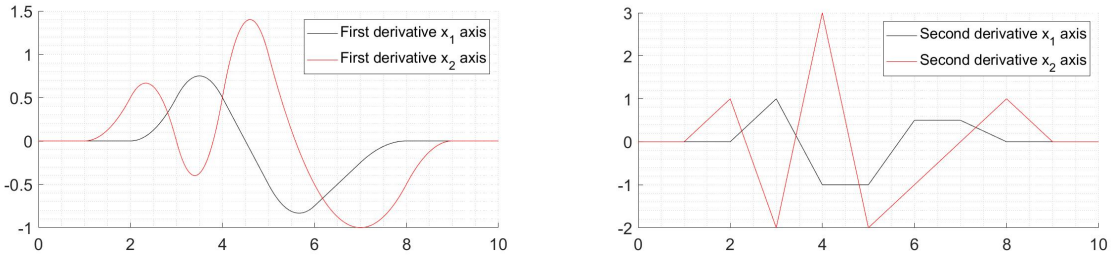


Figure 3.3: Example of a bi-dimensional uniform cubic B-spline



(a) First derivative

(b) Second derivative

Figure 3.4: First and second derivatives of the bi-dimensional uniform cubic B-spline from Figure 3.3

3.1.5.C Spiral curves

To enhance the quadrotor's surveillance mission, our proposed solution involves guiding it along a two-dimensional spiral trajectory. This trajectory is generated through a discretization process of a spiral curve. Therefore, it is crucial to establish a parametrization for the spiral curve. One possible approach is to utilize parametric equations, which can be expressed as follows:

$$\begin{aligned} x &= r_x(\gamma)\cos(\theta(\gamma)), \\ y &= r_y(\gamma)\sin(\theta(\gamma)). \end{aligned} \tag{3.15}$$

Here, γ represents the parametric variable, $\theta(\gamma)$ determines the angular variation of the curve, while the functions $r_{x,y}(\gamma)$ can be selected to control the rate at which the spiral's radius grows.

Let's consider a simple case from Equation (3.15):

$$\begin{aligned} r_x(\gamma) &= r_y(\gamma) = \gamma, \\ \theta(\gamma) &= \gamma. \end{aligned} \tag{3.16}$$

To calculate the angular velocity, is defined by:

$$\omega = \frac{\partial \theta(\gamma)}{\partial \gamma} = 1. \tag{3.17}$$

To find the linear speed, v, we differentiate x and y with respect to γ and then compute the magnitude of the resulting derivative vector. For the given example, it becomes:

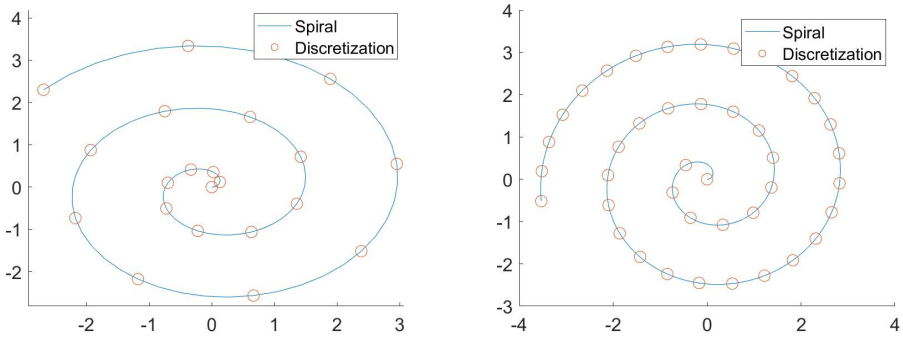
$$v(\gamma) = \left|\left| \frac{\partial \mathbf{C}(\gamma)}{\partial \gamma} \right|\right| = \sqrt{\left(\frac{dx}{d\gamma}\right)^2 + \left(\frac{dy}{d\gamma}\right)^2} = \omega(\gamma) r(\gamma) = \gamma. \tag{3.18}$$

From this example we can conclude that Equation (3.16) would lead to an incremental linear speed of the discretization process. That is, if we increment the variable γ in regular intervals, the distance between two consecutive discretized points increases with each step. This pattern is not desirable for our application, where a consistent spacing between the discretized points is advantageous.

We follow the result from [53] which suggest a valid choice for $r_x(\gamma)$, $r_y(\gamma)$ and $\theta(\gamma)$ that lead to a constant linear velocity (CLV) spiral. Defining V_{CLV} as the linear velocity along the spiral and Δr as the spiral pitch or sampling pitch along the radial dimension (a measurement for the growth rate of the radius), the chosen parametrization becomes:

$$\begin{aligned} r_x(\gamma) &= r_y(\gamma) = \sqrt{\frac{\Delta r V_{CLV}}{\pi}}\gamma, \\ \theta(\gamma) &= \sqrt{\frac{V_{CLV} 4\pi}{\Delta r}}\gamma. \end{aligned} \tag{3.19}$$

Figure 3.5 illustrate spiral curves generated from Equation (3.16) and Equation (3.19). In both cases, the discretization was obtained by assigning uniformly spaced values to γ. However, there is a notable difference between the two discretizations. With Equation (3.16), the discretization results in irregular intervals between each discretized point. The distance between consecutive waypoints starts at a very small value and grows to infinity. On the other hand, Equation (3.19) yields positions that are equally distributed along the spiral, maintaining a consistent distance between waypoints.



(a) Using Equation (3.16) **(b)** Using Equation (3.19)

Figure 3.5: Two examples of spiral parametrization and respective discretization

3.2 Problem statement

We aim to develop a real-time solution for early detection of wildfire events. To achieve this, we will utilize an uncertainty map that captures the importance of specific positions for drone inspection at different times. The proposed algorithm should be able to maximize the efficiency of the drone's inspection by directing it to the most relevant positions in the surveillance area. By utilizing sensors such as cameras or smoke detectors, the algorithm will gather new information to infer the presence of a fire and to update the uncertainty map.

We will also assume that the quadrotor has limited access to the map. This limitation may arise from communication constraints between the vehicle and the ground computer responsible for generating the complete map. It can also arise from onboard resource limitations, such as limited computational resources or memory capacity, which affect the quadrotor's ability to store and process extensive uncertainty maps or perform complex computations in real-time.

A complete formulation for the ideal surveillance algorithm was already introduced in Section 1.3. In this chapter, we will focus on a specific case of that original problem, presented in Equation 1.2. This 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 φ that will be utilized by the path planner, instead of directly addressing the sub-problem outlined in Equation 1.3
- Consider that the uncertainty map $h(x)$ is modeled as a discrete GM function, as explained in Section 2.3.1. Furthermore, we assume that a previously developed surveillance system is responsible for maintaining the uncertainty map.

We have opted to focus on the case where $K = 1$ because the sub-problems exhibit a high degree of independence from each other. By narrowing our attention to a single map, we can dedicate our research efforts to a specific instance and develop an algorithm that can be consistently applied to the remaining sub-problems.

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.

3.3 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.

To guide the vehicle towards this target location, we employ the hybrid algorithm described in Section 2.4.2. This algorithm has demonstrated efficient convergence to local maxima of GM functions within a limited number of iterations. Once it arrives at the target location, it carries out a surveillance mission using a spiral path to systematically scan the region.

Figure 3.6 provides a visual representation of the entire process. The path that must be followed by the quadrotor is parametrized as a two-dimensional uniform cubic B-spline. The vehicle is controlled by a path following contoller designed to ensure that the quadrotor's position $p(t)$ converges to a tube around the desired position $p_d(\gamma)$ (corresponding to the variable $\mathbf{C}(\gamma$ of the B-spline computations). This convergence can be made arbitrarily small, effectively making $||p(t) - p_d(\gamma)||$ approach a neighborhood of the origin. Our path following algorithm generates a set of angular references, which then need to be converted into thrust and angular references to be tracked by the quadrotor's inner loop controllers.

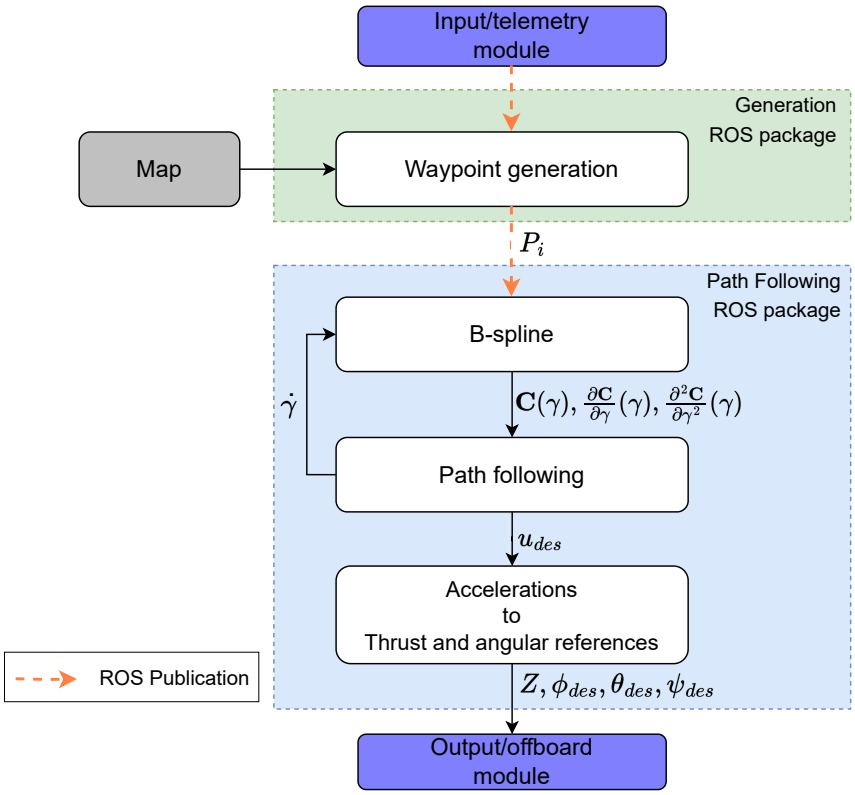


Figure 3.6: Diagram of the proposed solution. Each module is further explored in the following section

We will be developing our algorithms within the existing framework of the ISR's flight arena. Therefore, the purple input and output modules from Figure 3.6 should be interpreted as the correspondent modules from Figures 3.1 and 3.2, which are responsible for communication with the motion capture system and the PX4 autopilot. The gray module representing the generation of the uncertainty map is assumed to be already developed, and we will consider the map to be generated as a GM function. The white modules are the ones to be developed and will be explained in the following sections.

All modules have been developed using ROS and C++. Additionally, the waypoint generation has been implemented as a separate ROS package. This design decision was made to enhance the independence of the proposed strategies. By decoupling the waypoint generation from the path following algorithm, it becomes possible to change any of them for improvements in the future.

3.3.1 Waypoint generation

The generation of waypoints could follow a technique based on flocking with a gradient term as in [54,55]. However, given the development of an optimization algorithm in Definition 1, our proposal resorts to using the sequence of estimations for the maximum in tandem with the following state machine depicted in Figure 3.7. 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.

3.3.1.A States 1 and 2: Hybrid algorithm

In the first part, corresponding to states 1 and 2, the state machine employs the hybrid algorithm proposed in Chapter 2. The state machine initiates by computing the first waypoint using the global strategy (state 1). Based on the results from Chapter 2, we adopt the gradient descent with adaptive step as the global strategy. This state suggests a series of waypoints that guide the quadrotor towards a non-plateau region of the map, where the uncertainty values increase and the drone needs to conduct detailed surveillance.

In each iteration, the state machine computes an estimation of the second-order derivative with respect to each dimension of the map to determine if a transition to the local strategy is required. This approach is similar to the one employed by the supervisor in the hybrid algorithm. For further details on this computation, please refer to Section 2.4.2.

The local strategy state can be implemented using the gradient descent algorithm or any of its variants with momentum terms. The parameters for the local strategy are computed using the L-Lipschitz and m-strongly convex constants.

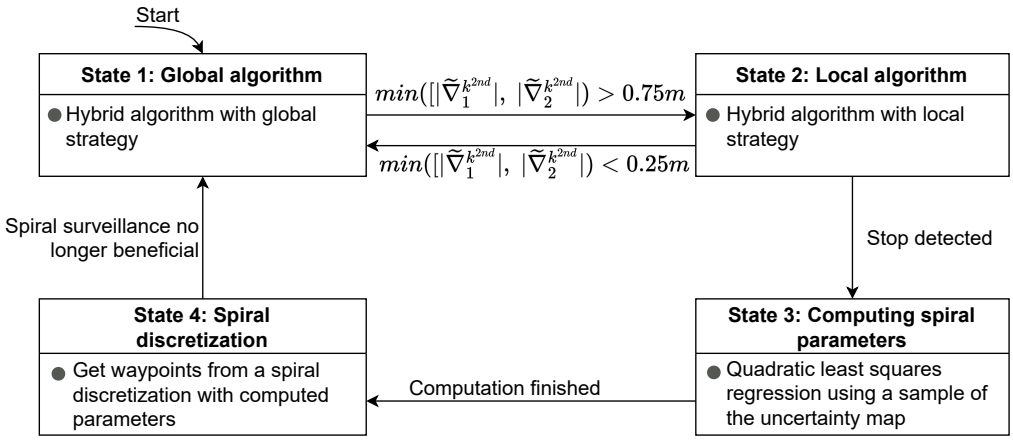


Figure 3.7: State machine for waypoint generation

3.3.1.B States 3 and 4: Spiral trajectory

The second part, corresponding to states 3 and 4, is activated when the state machine is in state 2 and detects that the waypoints are not improving its estimates for a local maximum. This condition can be identified, for instance, by setting a minimum threshold that two consecutive iterations must be distanced one from each other.

States 3 and 4 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. It is important to keep in mind that the uncertainty map comprises distinct Gaussian functions, each with different covariance values in each dimension. In this regard, we must consider the following aspects:

1. Path Following Performance: We aim to generate waypoints in a way that ensures a consistent distance between consecutive waypoints, thereby improving path following performance.
2. Surveillance Coverage Performance: We aim to generate a spiral path that is adapted to the expected region/Gaussian, maximizing the surveillance coverage performance.

To address the first consideration, we adopt a parameterization approach for the spiral path based on the suggestion from [53], as discussed in Section 3.1.5.C. 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 and meet the second requirement, 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 quadratic regression using the least squares principle. This approach involves solving the following optimization problem:

$$\underset{\hat{\beta}}{\text{minimize}} \quad \sum_{i=0}^{N} e_i^2(\hat{\beta}). \tag{3.20}$$

Here, N represents the number of data points to be fitted, $\hat{\beta}$ denotes the regression parameters for the model $\hat{h}$, which can be expressed as:

$$\hat{h} = \begin{bmatrix} x & y \end{bmatrix} \underbrace{\begin{bmatrix} a & b/2 \\ b/2 & c \end{bmatrix}}_{Q} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} d & e & f \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}. \tag{3.21}$$

Therefore, $\hat{\beta} = \begin{bmatrix} a & b & c & d & e & f \end{bmatrix}^T$, and the residual e_i is defined as the difference between the observed value h and the value predicted by the model using $\hat{h}$. To solve the least square problem, we rely on a closed form solution provided by the normal equations.

To simplify the computations, we can assume that $d \approx e \approx 0$ and, therefore, ignore them. We can also ignore the value of f since it does not affect the quadratic shape. By neglecting these variables we simplify the equation and focus on the main components of the quadratic form. Furthermore, to adapt the spiral radius to the desired shape we can find the major and minor axis of the ellipse defined by

$$\begin{bmatrix} x & y \end{bmatrix} Q \begin{bmatrix} x \\ y \end{bmatrix} = 1 \tag{3.22}$$

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 major and minor axis, and rotation of the ellipse as in Figure 3.8, the correspondent values are

$$a = \frac{1}{\sqrt{\lambda_1}}, \quad b = \frac{1}{\sqrt{\lambda_2}}, \quad \alpha = \arctan\left(\frac{v_1^{(2)}}{v_1^{(1)}}\right), \tag{3.23}$$

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 λ_i.

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 $||a + b|| = 1$, ensuring that their magnitudes do not influence the shape of the spiral.

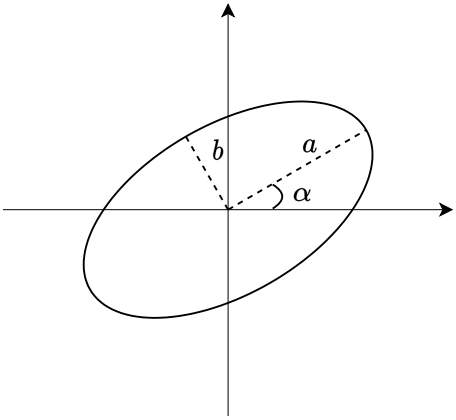


Figure 3.8: Characterization of an ellipse for the spiral path

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

$$\mathbf{C}(\gamma) = P_{center} + \begin{bmatrix} \cos(\alpha) & -\sin(\alpha) \\ \sin(\alpha) & \cos(\alpha) \end{bmatrix} \begin{bmatrix} a \cdot r(t)\cos(\theta(\gamma)) \\ b \cdot r(t)\sin(\theta(\gamma)) \end{bmatrix}, \tag{3.24}$$

where

$$\begin{aligned} r(\gamma) &= \sqrt{\frac{\Delta r V_{CLV}}{\pi}}\gamma, \\ \theta(\gamma) &= \sqrt{\frac{V_{CLV} 4\pi}{\Delta r}}\gamma. \end{aligned} \tag{3.25}$$

The parametrization variable γ is incremented by one unit after each discretized point. By adopting this strategy, the interpretation of V_{CLV} becomes simpler: it represents the distance that the parametriza-

tion variable travels along the spiral before the next discretization. The value of Δr can be adjusted for each vehicle based on its sensor's range.

The spiral curve is discretized regularly by the waypoint generation algorithm, and each waypoint is then sent to the path following ROS package, which is responsible for creating the final path to be followed by the quadrotor.

The state machine switches from state 4 to state 1 if one of the following conditions is met:

- A new map was received and the quadrotor is at a position where the estimation of the second-order derivative falls below a defined threshold
- The angle between the gradient at the current waypoint location and a vector pointing to the center o the spiral, defined as the difference between the current waypoint and the center of the spiral exceeds a defined threshold. This threshold, such as $90°$, indicates that the next waypoint computed by the hybrid algorithm is likely to lead the vehicle to a new region with higher uncertainty.

The second condition is designed to drive the vehicle to a new non-plateau region with a distinct local maximum, even if the next map has not been received. However, since a complete cycle of the state machine is assumed to take longer than the refreshing period of the uncertainty map filter, the state machine can be interrupted at any time. This allows for flexibility in responding to new information or changes in the environment.

- If the new map is received while the state machine is in state 2: it can continue in this state if the current location of the drone at the new map is considered near a local maximum. In the other case, the state machine will transit to state 1, and a global search is performed.
- If the new map is received while the state machine is in state 4: the spiral path can be interrupted if the current location of the drone no longer justified a spiral surveillance mission.

3.3.2 B-spline path

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. It is important to note two aspects:

- The waypoints are used as control points to construct the B-spline. Therefore, the desired path does not necessarily pass directly over the waypoints. However, this is not a concern because the main objective of the waypoints is to guide the quadrotor to a maximum of the uncertainty map. Additionally, the quadrotor will still traverse through high uncertainty zones where it can gather measurements using its sensors. This is possible because the B-spline path lies within the convex hull created by the control points.

- The path following ROS package is unaware of whether the current waypoints are generated by states 1, 2, or 4 of the state machine shown in Figure 3.7. Consequently, the final path to be tracked by the vehicle is generated in the same manner, regardless of whether the control points are obtained from the hybrid optimization algorithm or the discretization of the spiral curve.

To ensure a smoother initial movement of the drone the first waypoint is repeated four times, like in the example from Section 3.1.5.B. This results in the parameterized path coinciding with the first waypoint. By doing so, the drone's initial movement becomes less aggressive, allowing for a more gradual transition onto the planned path.

This module also takes in the time derivative of the parametrization variable $\dot{\gamma}$. This value will be interpreted in the following section as the speed of a virtual target moving along the desired path generated by the B-spline module. However, the module adjusts this value based on the current position error between the vehicle and the last computed curve position. Let us define this error as:

$$e_p = p(t) - p_d(\gamma). \tag{3.26}$$

The idea is to slow down the progression of γ when the position error increases. This allows the vehicle to converge to a neighborhood of the curve before the parametrized variable γ starts to follow the desired speed profile. To achieve this, a sigmoid-like function is used as a scaling factor:

$$\dot{\gamma}^* = \sigma(e_p)\dot{\gamma} \tag{3.27}$$

where

$$\sigma(e_p) = 1 + \frac{1}{1 + e^{e_p^2 - c}} - \frac{1}{1 + e^{-c}} \quad \in]0, 1]. \tag{3.28}$$

By tuning the value of the parameter c, the user can control the level of impact that the increase in position error has on $\dot{\gamma}$. Figure 3.9 demonstrated an example of this function for $c = 2$.

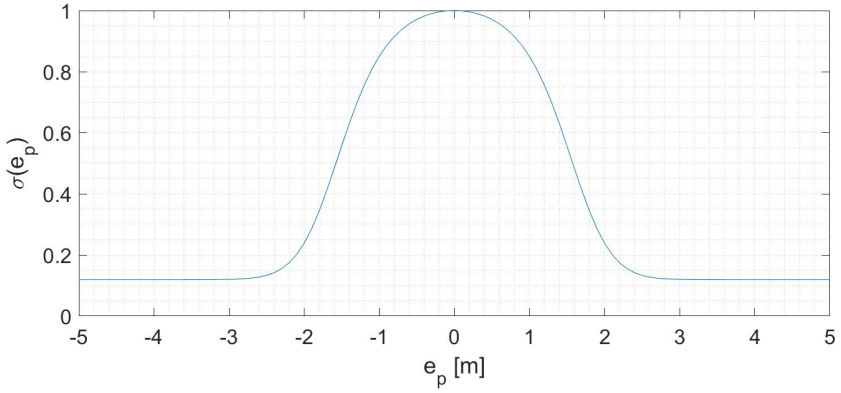


Figure 3.9: Example of the function $\sigma(e_p)$, for $c = 2$, used as scaling factor for the update of γ

To determine the current value of the parametrization variable, the module integrates the received time derivative over the elapsed time since the previous iteration:

$$\gamma \leftarrow \gamma + \Delta t \dot{\gamma}, \tag{3.29}$$

where δt represents the time interval since the B-spline module's previous iteration.

Another precaution that must be taken is to ensure that the value of γ is saturated based on the total number of waypoints received. This saturation guarantees that the algorithm does not try to generate more segments than the number of control points available. If the last segment is completely generated and no further control points are received, γ is saturated to its maximum value, and this module continues outputting the last point of that segment. Let N denote the number of waypoints received up to a given instant. This means that:

$$\gamma \leftarrow \max(\gamma + \Delta t \dot{\gamma}, N). \tag{3.30}$$

This module outputs the curve parametrization and it's firsts and second derivative. These values will be used by the path following algorithm to compute a position and velocity error that must be driven to zero using a Proportional Derivative controller.

3.3.3 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 $p_d(\gamma) \in \Re^2$. Figure 3.10 illustrates this approach.

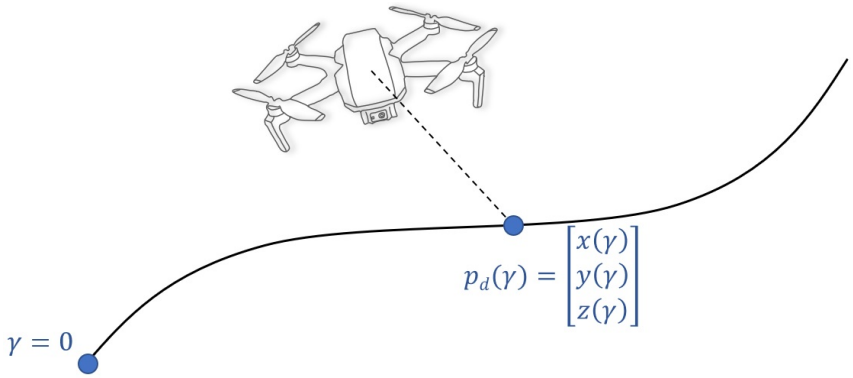


Figure 3.10: Virtual target for path following representation

The concept of a virtual target is usually used to guide the motion of a vehicle along a specified path by assigning a dynamic to this virtual target. Using this approach, it is possible to control the position that the vehicle should track at each instant. For that, let's consider that our virtual target is characterized by

a desired speed profile given by:

$$v_d(\gamma) = \frac{V_{des}}{||p'_d(\gamma)|| \cdot (1 + K||p''_d(\gamma)||)}, \tag{3.31}$$

where $p'_d(\gamma)$ and $p''_d(\gamma)$ correspond to the first and second derivatives of the path with respect to γ, respectively. The constant V_{des} represents the desired constant speed for the vehicle.

This profile was suggested by Romulo et al. [56], allowing the virtual target, and consequently, the vehicle, to reduce its velocity during sharper turns and approach the desired constant speed V in straight lines. Furthermore, the term $|p'_d(\gamma)|$ enables the virtual target to slow down in larger path segments where $|p'_d(\gamma)|$ increases.

Our path following algorithm aims to generate a set of desired accelerations, denoted as $u_{des} \in \Re^3$, which latter will be converted by the next module into thrust and angular references for the PX4 autopilot's inner loop (see Figure 3.6). Our path following rule 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 the desired speed profile. 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:

$$e_p \quad := p(t) - p_d(\gamma) \tag{3.32}$$

$$e_v \quad := \dot{p} - \tfrac{\partial p_d}{\partial \gamma} v_d(\gamma) \tag{3.33}$$

In Jacinto et al. [57], the author considered the quadrotor as a double integrator and proposes a control law based on the Lyapunov stability principle. The control law utilizes a Proportional-Derivative (PD) controller for the vehicle's position, along with a feed-forward term for the acceleration. The controller, as proposed by Jacinto, can be expressed as follows:

$$u_{des} = h(\gamma)v_d(\gamma) - K_p e_p - K_v e_v, \tag{3.34}$$

where u_{des} represent the desired acceleration that must be verified by the vehicle, while K_p and K_v are the control gains that need to be adjusted by the user. 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

$$\begin{aligned}\frac{d^2p_d}{dt^2}(\gamma) = \frac{\partial}{\partial t}\left(\frac{\partial p_d}{\partial \gamma}v_d(\gamma)\right) &= \frac{d}{dt}\left(\frac{\partial p_d}{\partial \gamma}\right)v_d(\gamma) + \frac{\partial p_d}{\partial \gamma}\frac{dv_d}{dt}(\gamma) \\ &= \frac{\partial^2 p_d}{\partial \gamma^2}\dot{\gamma}v_d(\gamma) + \frac{\partial p_d}{\partial \gamma}\left(\frac{\partial v_d}{\partial \gamma}\dot{\gamma}\right) \\ &= \underbrace{\left[\frac{\partial^2 p_d}{\partial \gamma^2}v_d(\gamma) + \frac{\partial p_d}{\partial \gamma}\frac{\partial v_d}{\partial \gamma}\right]}_{h(\gamma)}\dot{\gamma} \\ &= h(\gamma)\dot{\gamma}.\end{aligned} \tag{3.35}$$

For our work, we extended the proposed control law by Jacinto and incorporated an Integral term to further enhance the control performance. The inclusion of the Integral term allows for the integration of the accumulated position errors over time, which helps to address steady-state errors and improve the controller's response to long-term deviations.

The extended controller can be expressed as follows:

$$u_{des} = h(\gamma)v_d(\gamma) - K_p e_p - K_v e_v - K_i \int e_p, \tag{3.36}$$

where $\int e_p$ represents a discrete integration process of e_p, and K_i the correspondent control gain.

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

$$\dot{\gamma} = v_d(\gamma). \tag{3.37}$$

At this point, we have the capability to generate a smooth path using the waypoints provided by the waypoint generator. Additionally, we have developed a following controller that generates a set of accelerations for the quadrotor to follow in order to successfully converge to the desired path. In the following section, we will introduce a module that is capable of converting these accelerations along each axis of the vehicle into thrust and angular references. These references can then be utilized by the inner-loop controller of the PX4 autopilot.

3.3.4 Accelerations to Trust 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:

- $\eta_1 = [x, y, z]^T$: Position of the origin of frame $\{B\}$ measured in frame $\{U\}$.

- $\eta_2 = [\phi, \theta, \psi]^T$: Orientation of frame $\{B\}$ with respect to frame $\{U\}$, expressed in Euler angles.

- $F_{RB} = [X, Y, Z]^T$: External forces measured in frame $\{B\}$.

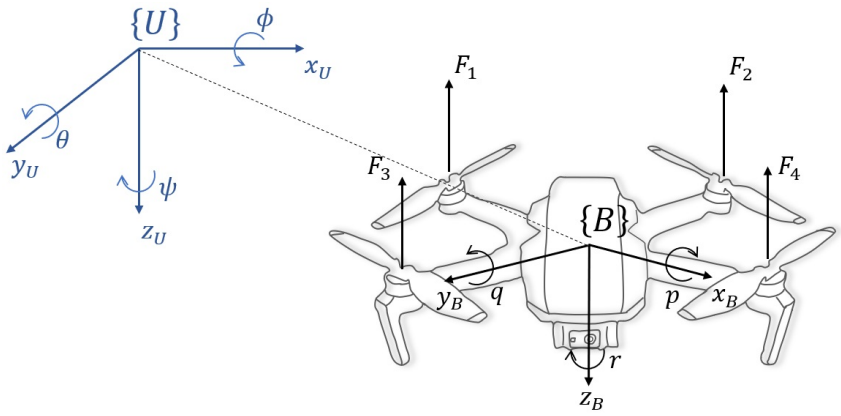


Figure 3.11: Adopted reference frames

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

$$\ddot{p} \coloneqq \ddot{\eta}_1 = -\frac{Z}{m} {}^{U}_{B}R(\eta_2)e_3 + ge_3, \tag{3.38}$$

where ${}^{U}_{B}R(\eta_2)$ represents the rotation matrix from the body reference frame (see Figure 3.11) to the inertial frame and can be defined as

$${}^{U}_{B}R(\eta_2) = R_z(\psi)R_y(\theta)R_x(\phi) = \begin{bmatrix} c\theta c\psi & s\phi s\theta c\psi - c\phi s\psi & c\phi s\theta c\psi + s\phi s\psi \\ c\theta s\psi & s\phi s\theta s\psi + c\phi c\psi & c\phi s\theta s\psi - s\phi c\psi \\ -s\theta & s\phi c\theta & c\phi c\theta \end{bmatrix} \tag{3.39}$$

with c and s denoting 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. Our outer loop is responsible for producing desired accelerations u_{des}. Hence, it is crucial to develop a subsystem capable of computing the total thrust Z and the desired attitude using the desired accelerations. In this work, we will consider the yaw angle reference, ψ_{des}, as a free variable defined by the outer-loop controller. Specifically, we will not impose any specific rules on how the yaw angle should vary. Instead, we will set ψ_{des} to a constant value of zero.

Expanding Equation (3.38), we get

$$u_{des} = \ddot{\eta}_1 = -\frac{1}{m} R_z(\psi_{des})[R_y(\theta)R_x(\phi)Z]e_3 + ge_3 \tag{3.40}$$

which can be further simplified by defining the auxiliary variable u^* as

$$u^* \coloneqq R_y(\theta)R_x(\phi)Z. \tag{3.41}$$

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 Equation (3.41) in Equation (3.40):

$$u^* = -mR_z^T(\psi_{des})(u_{des} - ge_3) \tag{3.42}$$

which has a relationship with the required thrust by

$$||u^*|| = Z. \tag{3.43}$$

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

$$\frac{u^*}{||u^*||} = \begin{bmatrix} \cos(\phi)\sin(\theta) \\ \sin(\theta) \\ \cos(\phi)\cos(\theta) \end{bmatrix}. \tag{3.44}$$

By solving Equation (3.44) with respect to the attitude angles and considering that $u^* = [u_1^*, u_2^*, u_3^*]^T$, we obtain the following expressions:

$$\phi_{des} = \arcsin\left(-\frac{u_2^*}{Z}\right) \tag{3.45}$$

$$\theta_{des} = \arccos\left(\frac{u_1^*}{u_3^*}\right). \tag{3.46}$$

We can now transmit the thrust and attitude references to the inner loop of the PX4 autopilot. The PX4 autopilot then calculates the precise control signals to be sent to the quadrotor's propellers, enabling it to accurately follow the desired path.

3.4 Simulation results

In this section, the previously proposed solution is evaluated using a highly reliable and detailed simulation environment provided by Gazebo. This powerful tool is integrated with the framework described in Section 3.1.2, offering a simulation environment that closely resembles real-world flight arenas. The integration ensures that all the developed modules in C++ and ROS are fully compatible and functional within the Institute for System and Robotics' flight arena framework. For a more comprehensive understanding of the framework's technical details, please refer to Section 3.1.2.

From the solution description, some variables and constants can be tailored for the specific vehicle

and usage case. Therefore, for the results presented in this section we defined the following:

- Global algorithm for state 1 of the state machine: Gradient descent with adaptive step
- Local algorithm for state 2 of the state machine: Heavy Ball algorithm with adaptive step
- The sigmoid-like function uses the parameter $c = 2$
- The spiral is discretized using $V_{CLV} = 1$
- The spiral is discretized using $\Delta_r = 2$
- The desired speed profiled for the virtual target uses $V_{des} = 0.7\ m/s$
- The PID controller uses $K_p = [5.5, 5.5, 4.5],\ K_i = [0.5, 0.5, 0.75],\ K_d = [2.0, 2.0, 1.0]$
- The desired yaw angle is not controlled by the outer-loop. Therefore, along all fligth $\psi_{dis} = 0$
- The quadrotor's altitude reference does not vary along its flight. The drone is pretended to maintain a constant altitude of 1 meter
- The output acceleration computed by the path following algorithm was saturated to avoid agressive manovour by the vehicle. Thefore $||u_{dis}|| \leq 3m/s^2$.

The tests are structured in a constructive manner, starting with tests 1 and 2, where we evaluate the path following ROS package using a predefined set of waypoints. This selection enables us to specifically assess the performance of the B-spline generator, the path following controller, and the module responsible for transforming the output of the outer loop controller into thrust and angular references.

Regarding tests 3 and 4, the uncertainty maps are discretized to simulate scenarios where a real estimation filter may not be able to compute a continuous map accurately. To address this, the waypoint generation algorithms must consider the necessary adaptations, as explained in Sections 2.3.2 and 2.4.2.A.

For each test, the analysis is conducted using the estimated real position provided by the Extended Kalman Filter of the vehicle.

3.4.1 Test 1: Path following performance

For the first test, the drone was initialized at $(x, y) = (-4, -5)$. However, the first waypoint was defined at $(x, y) = (-3, -4)$. As a result, the vehicle had to overcome the initial error and converge to the desired path. The last 5 waypoints were repeated 2 times to create an almost circular path of radius one meter. This allowed us to test if the vehicle is able to adapt to a circular path.

Figure 3.12 shows a top view of the desired path, defined by the B-spline module, and the actual path of the vehicle. The figure also displays the location of each waypoint, which serves as control points for the B-spline module, and the initial and final positions of the quadrotor.

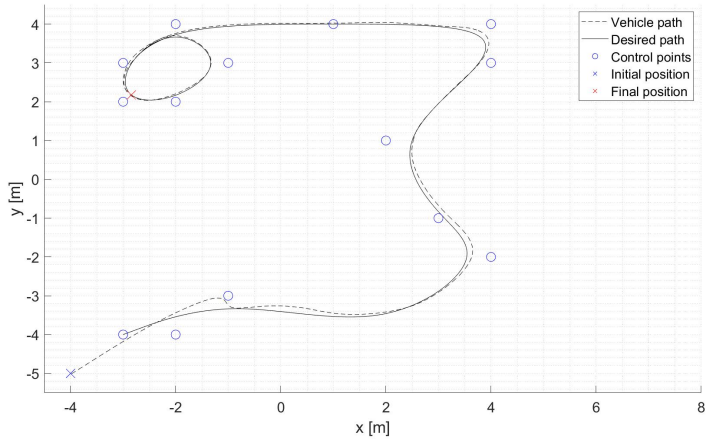
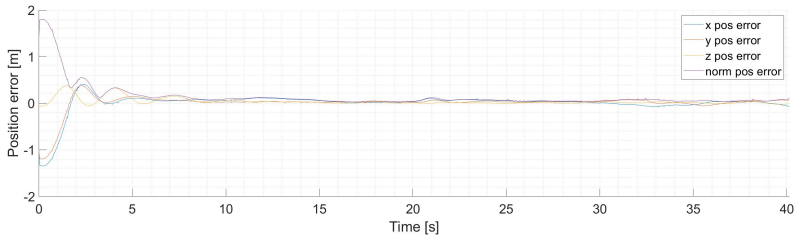
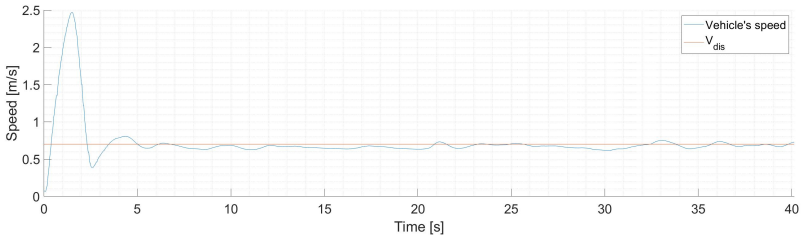


Figure 3.12: Quadrotor test 1: Top view of the desired and performed path

From Figure 3.13(a) it can be concluded that 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, despite the initial error. Furthermore, Figure 3.13(b) demonstrates that the vehicle also achieved convergence to the desired speed of $V_{des} = 0.7\ m/s$. This speed is included in the algorithm trough the desired speed profile of the virtual target $v_d(\gamma)$.



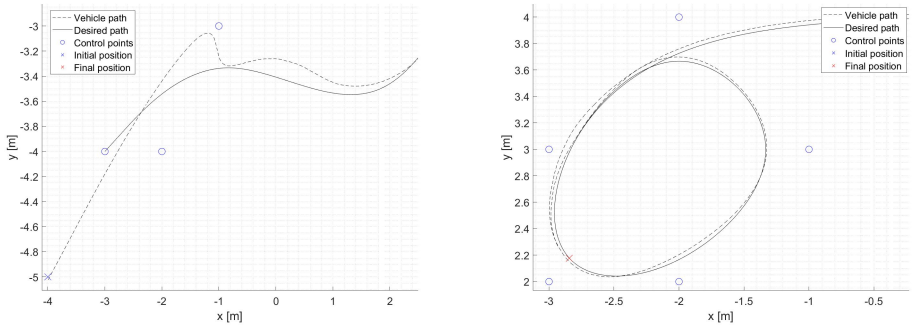
(a) Position error



(b) Speed and desired speed

Figure 3.13: Quadrotor test 1: Performance of the path following algorithm

Figure 3.14 highlights the beginning of the quadrotor's trajectory, showing the convergence of its position towards the desired path. It also showcases the final part of the trajectory, where the last 5 waypoints were repeated 2 times to create an almost circular curve with a radius of one meter. These figures provide visual evidence supporting the confidence in the performance of the system.



(a) Position error **(b)** Speed and desired speed

Figure 3.14: Quadrotor test 1: Trajectory highlights

3.4.2 Test 2: Path following performance for spiral curve

The second test was conducted to evaluate the performance of the following strategy when using waypoints computed from a discretization process of a spiral curve. To initiate the test, we parameterized a

spiral with shaping parameters $a = 2$ and $b = 1$. The first 40 waypoints were then computed from this spiral and utilized by the B-spline module to generate the desired path. Figure 3.15 shows a top view of the desired path, defined by the B-spline module, and the actual path of the vehicle. The figure also displays the location of each waypoint, which serves as control points for the B-spline module, and the initial and final positions of the quadrotor.

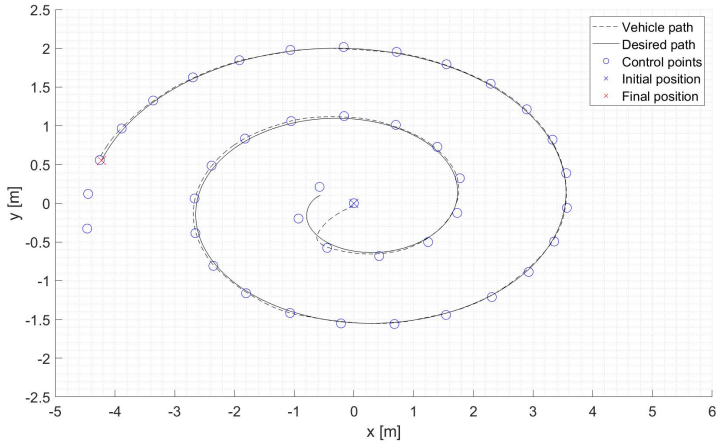
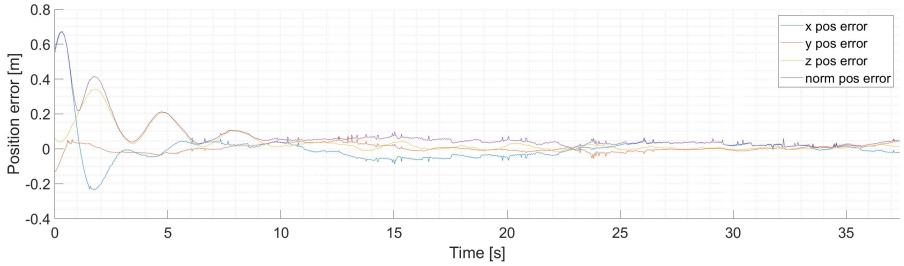
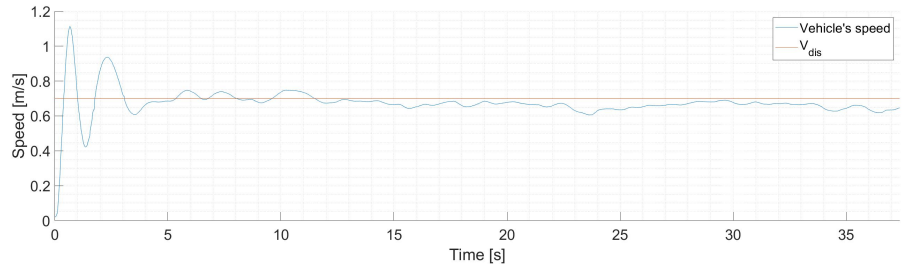


Figure 3.15: Quadrotor test 2: Top view of the desired and performed path

Figure 3.16 demonstrates the successful convergence of the vehicle to the desired path, as evidenced by the tracking error approaching zero and the vehicle's speed aligning with the desired velocity of $V_{des} = 0.7\ m/s$. These results reaffirm the effectiveness of the proposed strategies in enabling accurate path following.



(a) Position error



(b) Speed and desired speed

Figure 3.16: Quadrotor test 2: Performance of the path following algorithm

Figure 3.17 shows a comparison between the parametrized spiral and the generated cubic B-spline. Although they are not perfectly aligned, the B-spline effectively captures the shape and characteristics of the spiral curve. It is noteworthy that while we could directly utilize the spiral's parametrization for the path following controller, this approach would deviate from the modular approach aimed for this work. By using the B-spline representation, the path following ROS package remains agnostic to the computation process of the waypoints, allowing the user to use another waypoint generation ROS package.

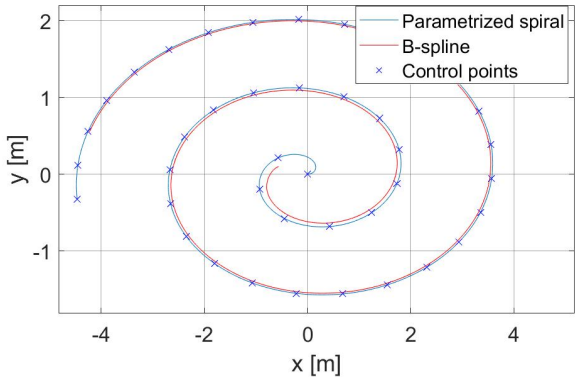


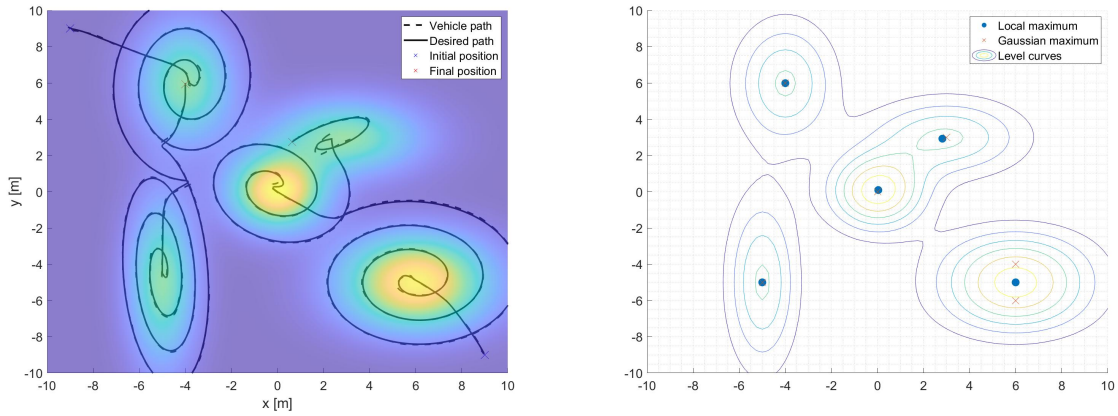
Figure 3.17: Quadrotor test 2: Comparison between the original spiral curve and the cubic B-spline generated through a discretization process of the spiral.

3.4.3 Test 3: Waypoint generation and path following with an uncertainty map with 6 Gaussians

The third test aimed to assess the performance of the overall proposed algorithm, which includes the waypoint generation module.

For this test, a discretized version of the previously introduced map consisting of 6 Gaussians, as presented in the third test of the previous chapter (see Section 2.5.3), was utilized. 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.

Figure 3.18 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.



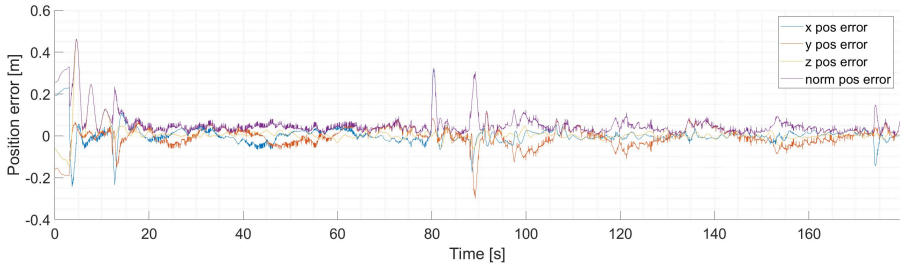
(a) Top view of the desired and performed path

(b) Composition of the uncertainty map with the location of each of the 6 Gaussians and location of the local maximums

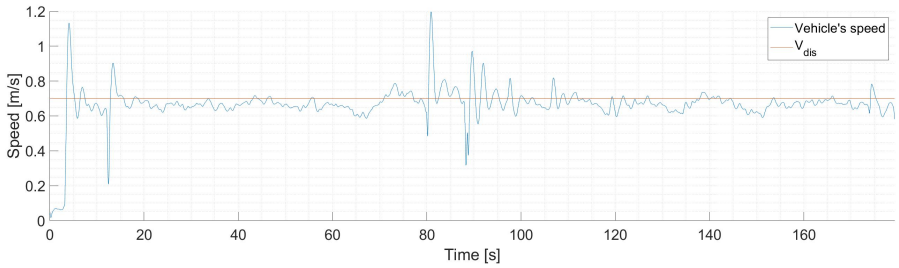
Figure 3.18: Quadrotor test 3: Performance of the overall algorithm for two intialiatizations

Figure 3.19 illustrates the position tracking error and the speed of the vehicle during the first run of the test, with an initial position at $(-9, 9)$. From the graph, we can conclude that the tracking error remained close to zero for most of the flight, with occasional peaks of around 0.5 meters in the more

aggressive parts of the trajectory. These peaks are observed at the start of the flight when the vehicle transitions from its initial location to the parametrized path, 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 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. The system's ability to make the drone track the desired path is demonstrated by achieving a root mean square error of the tracking position of 0.0802 meters for the first run and 0.0977 meters for the second run.



(a) Position error



(b) Speed and desired speed

Figure 3.19: Quadrotor test 3: Performance of the path following algorithm for the first run with initial location $(-9, 9)$

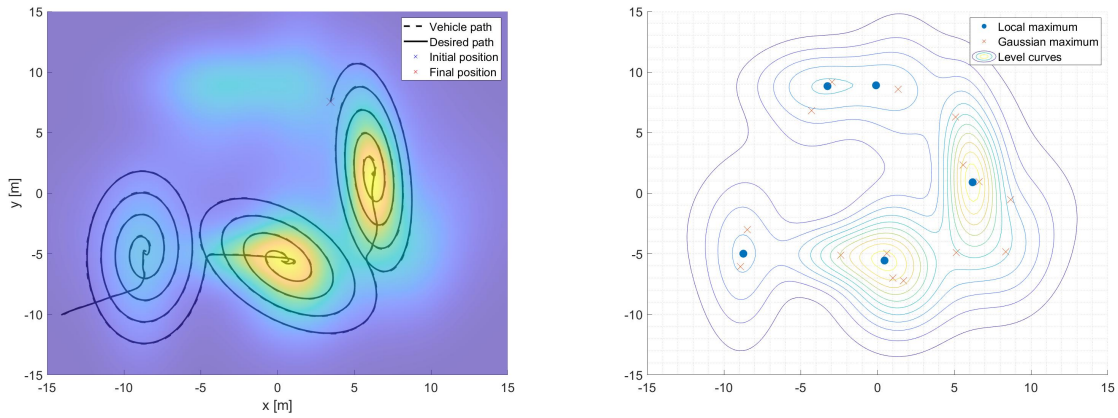
3.4.4 Test 4: Waypoint generation and path following with a random uncertainty map

The procedure for the fourth test is similar to the previous one. However, in this case, we used a randomly generated map consisting of 15 random Gaussians. The centers of these Gaussians were generated using $\mu_k = \begin{bmatrix}\mu_{x_1} & \mu_{x_2}\end{bmatrix}^T$, where $\mu_{x_1}, \mu_{x_2} \in [-10, 10]$. The covariance matrices for the Gaussians are generated using $\Sigma_k = \begin{bmatrix}\sigma_1^2 & 0 \\ 0 & \sigma_2^2\end{bmatrix}$, where $\sigma_1, \sigma_2 \in [1, 21]$.

Figure 3.20 we illustrate one run of the complete algorithm with the drone starting at $(-14, -10)$. The waypoint algorithm successfully suggested waypoints that guided the vehicle to three distinct local

maxima. For each maximum, the quadratic regression using the least squares principle was able to determine a valid parametrization for the spiral curve, considering the length of the major and minor axes and the correct inclination angle.

This particular map poses a significant challenge due to the highly overlapping Gaussian functions. For instance, the non-plateau region around $(0, -5)$ has a shape that deviates significantly from a Gaussian shape. Despite this, our algorithm was able to extract certain properties that directed the vehicle to survey the area with the highest uncertainty first. In fact, during the first lap, the spiral curve successfully traversed the entire yellow area.

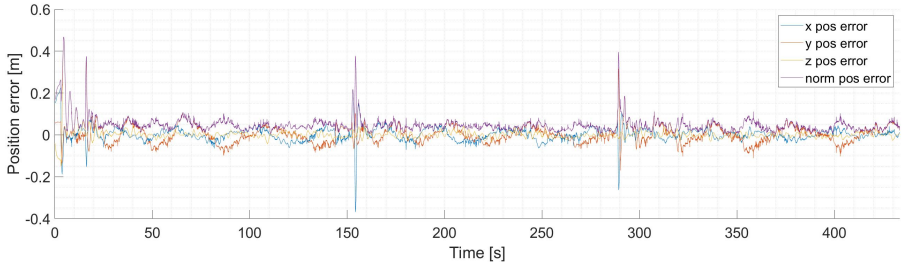


(a) Top view of the desired and performed path

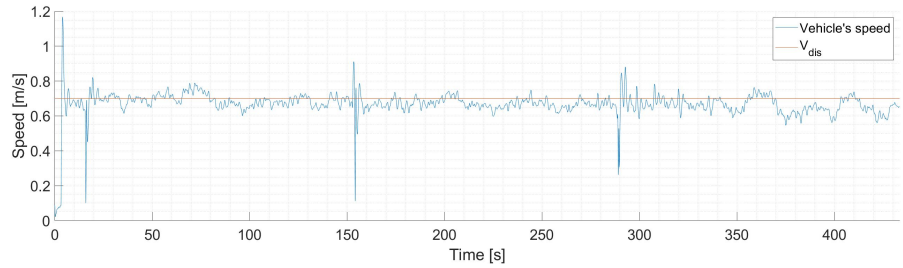
(b) Composition of the uncertainty map with the location of each of the 15 Gaussians and location of the local maximums

Figure 3.20: Quadrotor test 4: Performance of the overall algorithm

Figure 3.21 depicts the position tracking error and speed of the vehicle, exhibiting a similar performance to the previous tests. Notably, there are four prominent peaks in both the tracking error and the vehicle's speed. These peaks occur during the initial movement of the vehicle and at the beginning of each spiral path. The system's ability to make the drone track the desired path is demonstrated by achieving a root mean square error of the tracking position of 0.062 meters.



(a) Position error



(b) Speed and desired speed

Figure 3.21: Quadrotor test 4: Performance of the path following algorithm

3.5 Conclusion

In this chapter, we proposed a real-time surveillance system inspired by the optimization problem formulation in Chapter 1. We relied on empirical intuition to develop a system capable of analyzing the uncertainty map and suggesting a set of waypoints to navigate the vehicle through the most uncertain areas.

After discussing relevant background theory of aerial vehicles, we progress to the control of a quadrotor using feedback control algorithms and the challenges of path following controllers that enable precise and smooth navigation along predetermined paths. To explore the parametrization of these paths, we conducted a detailed review of B-spline and spiral curve parametrization, with a particular focus on uniform cubic B-splines and Constant Linear Velocity spirals.

Our proposed solution consists of a system divided into two distinct but interconnected components: i) a waypoint generation package that directly maximizes the value of the uncertainty function; ii) a path following package that incorporates smooth cubic B-spline parametrization and an outer-loop path following PID controller. This modular design allows users to test alternative algorithms for analyzing uncertainty and suggesting waypoints, or to integrate our approach with different path following methods.

Finally, our complete system was analyzed using a highly reliable and detailed simulation environment provided by Gazebo, integrated with ISR flight arena's framework. This allowed our tests to repli-

cate flight conditions and closely resemble real-world scenarios. The tests were structured in a constructive manner, starting by the validation of our path following package using predefined waypoints. We then proceeding for the evaluation of the complete approach, using two uncertainty maps. The results showed a path that drove the vehicle through the most uncertain areas, with near straight lines guiding the vehicle to local maxima and spiral curves that successfully shaped its evolution as a function of the area where the quadrotor was located.

4

Conclusion

Contents

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.

Chapter 2 was dedicated to the creation of a hybrid algorithm capable of converging to local maximums of non-convex functions. To achieve this, a comprehensive literature review on first-order optimization algorithms was conducted, highlighting their respective strengths and limitations. By modeling non-convex functions as GMs, 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, thereby enhancing surveillance coverage. 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. We provide a detailed description of B-spline parametrizations and explain how we can adapt them to our problem. 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 highly reliable and 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, observed in the second run of the third test. Furthermore, the trajectory is successfully tailored to address the shape of the uncertainty region in which the drone

was conducting surveillance. This showcases the precision and robustness of our proposed solution.

4.1 Future work

The work presented in this thesis can be further developed and tested. Namelly:

- Test the proposed hybrid algorithm in Chapter 2 with different oracles that provide estimations for the L-Lipschitz and m-strongly convex constants. As the main contribution of the second chapter is the hybrid algorithm, it is important to assess its independence from our oracle. Another oracle could use, for example, evaluation of the objective function instead of evaluation of the individual Gaussians;
- Test and adapt the behavior of the proposed hybrid algorithm to non-convex functions when they are not modeled as GMs. Our approach may assume certain conditions verified by GM functions but not verified by other non-convex functions. For example, it would be pertinent to evaluate the performance in functions with several local minimizers;
- Test the surveillance algorithm from Chapter 3 with more than one map, incorporating consequent maps generated using measurements gathered by the deployed vehicle. Our approach is designed to react to updates in the uncertainty map. However, there could be unforeseen dynamics due to changing conditions in the information sources used by the estimation filter that need to be considered by our algorithms;
- Introduce sensor models of the onboard sensor in the path parametrization and following. Different sensors may impose operational constraints, such as maximum velocities or specific attitudes (especially for the yaw angle), in order to maximize their performance. Therefore, it would be important to assess these characteristics in the path planning phase;
- Test the proposed architecture from Chapter 3 in a real flight arena. Although simulation tools like Gazebo are capable of accurately emulating aerial vehicle dynamics, it is important to test the algorithms in real flight. Real conditions introduce nonlinearities, disturbances, and noises that cannot be predicted in simulation tools;
- The performance of our algorithm can be compared with the performance of solutions generated using optimizers that can directly solve the optimization problem formulated in the first chapter, such as Model Predictive Control.

Bibliography

[1] T. A. da Silva Oliveira, “Rapid development and prototyping environment for testing of unmanned aerial vehicles,” Master’s thesis, Instituto Superior Técnico, 2021.

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

[3] Resolução do Conselho de Ministros n.º 159/2017, “Diário da república, 1.ª série-n.º 209-30 de outubro de 2017,” 2017. [Online]. Available: www.dre.pt

[4] “Consolidated version of the treaty establishing the european atomic energy community,” https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX:12012A/TXT, accessed: 2022-08-17.

[5] R. Armitage, “To CCTV or not to CCTV?: A review of current research into the effectiveness of CCTV systems in reducing crime,” *Nacro Briefing Note*, 2002.

[6] R. Surette, “The thinking eye: Pros and cons of second generation CCTV surveillance systems,” *Policing: An International Journal of Police Strategies & Management*, 2005.

[7] M. Lopes, “Desde 2013, videovigilância na rua passou de 38 para mais de 850 câmaras autorizadas,” *Publico*. [Online]. Available: https://www.publico.pt/2021/09/20/politica/noticia/desde-2013-videovigilancia-rua-passou-38-850-camaras-autorizadas-1978004

[8] “U.S. Border Patrol deploys Autonomous Surveillance Towers for West Texas and New Mexico,” *U.S. Custom and Border Protection*. [Online]. Available: https://www.cbp.gov/newsroom/local-media-release/us-border-patrol-deploys-autonomous-surveillance-towers-west-texas-and

[9] V. Machi, “Half-autonomous ships and space systems top european defence fund’s 2023 wish list,” *Defense News*. [Online]. Available: https://www.defensenews.com/global/europe/2022/06/15/half-autonomous-ships-and-space-systems-top-european-defence-funds-2023-wish-list

[10] S. Bao, N. Xiao, Z. Lai, H. Zhang, and C. Kim, “Optimizing watchtower locations for forest fire monitoring using location models,” *Fire safety journal*, vol. 71, pp. 100–109, 2015.

[11] J. Lloret, M. Garcia, D. Bri, and S. Sendra, "A wireless sensor network deployment for rural and forest fire detection and verification," *Sensors*, vol. 9, no. 11, pp. 8722–8747, 2009.

[12] M. Hefeeda and M. Bagheri, "Wireless sensor networks for early detection of forest fires," in *IEEE International Conference on Mobile Adhoc and Sensor Systems Conference*. IEEE Computer Society, 2007, pp. 1–6.

[13] F. X. Catry, F. C. Rego, F. L. Bação, and F. Moreira, "Modeling and mapping wildfire ignition risk in portugal," *International Journal of Wildland Fire*, vol. 18, no. 8, pp. 921–931, 2009.

[14] J. Verde and J. Zêzere, "Assessment and validation of wildfire susceptibility and hazard in portugal," *Natural Hazards and Earth System Sciences*, vol. 10, no. 3, pp. 485–497, 2010.

[15] IPMA, "Cálculo do Índice de Risco de Incêndio Rural - Risco Conjuntural e Meteorológico (RCM)," Instituto Português do Mar e da Atmosfera, I.P, Tech. Rep., 2020.

[16] C. Van Wagner *et al.*, *Structure of the Canadian forest fire weather index*. Environment Canada, Forestry Service Ontario, 1974, vol. 1333.

[17] Z. Xu, L. Mei, K.-K. R. Choo, Z. Lv, C. Hu, X. Luo, and Y. Liu, "Mobile crowd sensing of human-like intelligence using social sensors: A survey," *Neurocomputing*, vol. 279, pp. 3–10, 2018.

[18] H. Zheng, Y. Hong, D. Long, and H. Jing, "Monitoring surface water quality using social media in the context of citizen science," *Hydrology and Earth System Sciences*, vol. 21, no. 2, pp. 949–961, 2017.

[19] W.-T. Chen, P.-Y. Chen, W.-S. Lee, and C.-F. Huang, "Design and implementation of a real time video surveillance system with wireless sensor networks," in *IEEE Vehicular Technology Conference*. IEEE, 2008, pp. 218–222.

[20] X. Song, L. Sun, J. Lei, D. Tao, G. Yuan, and M. Song, "Event-based large scale surveillance video summarization," *Neurocomputing*, vol. 187, pp. 66–74, 2016.

[21] "Nasa website about modis instrument," https://modis.gsfc.nasa.gov/data/, accessed: 2023-05-31.

[22] G. Rowlands, J. Brown, B. Soule, P. T. Boluda, and A. D. Rogers, "Satellite surveillance of fishing vessel activity in the ascension island exclusive economic zone and marine protected area," *Marine Policy*, vol. 101, pp. 39–50, 2019.

[23] M. Q. Ngo, P. D. Haghighi, and F. Burstein, "A crowd monitoring framework using emotion analysis of social media for emergency management in mass gatherings," *arXiv preprint arXiv:1606.00751*, 2016.

[24] D. Di Paola, A. Milella, G. Cicirelli, and A. Distante, “An autonomous mobile robotic system for surveillance of indoor environments,” *International Journal of Advanced Robotic Systems*, vol. 7, no. 1, p. 8, 2010.

[25] K. Kim, S. Bae, and K. Huh, “Intelligent surveillance and security robot systems,” in *IEEE Workshop on Advanced Robotics and its Social Impacts*. IEEE, 2010, pp. 70–73.

[26] G. Ferri, A. Munafò, A. Tesei, P. Braca, F. Meyer, K. Pelekanakis, R. Petroccia, J. Alves, C. Strode, and K. LePage, “Cooperative robotic networks for underwater surveillance: an overview,” *IET Radar, Sonar & Navigation*, vol. 11, no. 12, pp. 1740–1761, 2017.

[27] R. E. Kalman, “A new approach to linear filtering and prediction problems,” *Journal of Basic Engineering*, 1960.

[28] L. Shi, K. H. Johansson, and R. M. Murray, “Kalman filtering with uncertain process and measurement noise covariances with application to state estimation in sensor networks,” in *IEEE International Conference on Control Applications*, 2007, pp. 1031–1036.

[29] C. L. V. Rodeiro and A. B. Lawson, “Online updating of space-time disease surveillance models via particle filters,” *Statistical methods in medical research*, vol. 15, no. 5, pp. 423–444, 2006.

[30] J. Nocedal and S. J. Wright, *Numerical Optimization*, 2nd ed. New York, NY, USA: Springer, 2006.

[31] S. Du, J. Lee, H. Li, L. Wang, and X. Zhai, “Gradient descent finds global minima of deep neural networks,” in *International conference on machine learning*. PMLR, 2019, pp. 1675–1685.

[32] W. Li and C. G. Cassandras, “Distributed cooperative coverage control of sensor networks,” in *Proceedings of the 44th IEEE Conference on Decision and Control*. IEEE, 2005, pp. 2542–2547.

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

[34] B. T. Polyak, “Some methods of speeding up the convergence of iteration methods,” *USSR Computational Mathematics and Mathematical Physics*, vol. 4, no. 5, pp. 1–17, 1964.

[35] 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.

[36] 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.

[37] D. Silvestre, “Optool—an optimization toolbox for iterative algorithms,” *SoftwareX*, vol. 11, p. 100371, 2020.

[38] D. Silvestre, J. Hespanha, and C. Silvestre, “Fast desynchronization algorithms for decentralized medium access control based on iterative linear equation solvers,” *IEEE Transactions on Automatic Control*, vol. 67, no. 11, pp. 6219–6226, 2022.

[39] ——, “Desynchronization for decentralized medium access control based on gauss-seidel iterations,” in *2019 American Control Conference (ACC)*, 2019, pp. 4049–4054.

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

[41] N. Elmeseiry, N. Alshaer, and T. Ismail, “A detailed survey and future directions of unmanned aerial vehicles (uavs) with potential applications,” *Aerospace*, vol. 8, no. 12, 2021.

[42] T. Oliveira, P. Trindade, D. Cabecinhas, P. Batista, and R. Cunha, “Rapid development and prototyping environment for testing of unmanned aerial vehicles,” in *IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC)*, 2021, pp. 191–196.

[43] J. P. How, B. Behihke, A. Frank, D. Dale, and J. Vian, “Real-time indoor autonomous vehicle test environment,” *IEEE Control Systems Magazine*, vol. 28, no. 2, pp. 51–64, 2008.

[44] S. Lupashin, M. Hehn, M. Mueller, A. Schoellig, M. Sherback, and R. D’Andrea, “A platform for aerial robotics research and demonstration: The flying machine arena,” *Mechatronics*, vol. 24, 02 2014.

[45] R. Mahony, V. Kumar, and P. Corke, “Multirotor aerial vehicles: Modeling, estimation, and control of quadrotor,” *IEEE Robotics & Automation Magazine*, vol. 19, no. 3, pp. 20–32, 2012.

[46] B. Erginer and E. Altug, “Modeling and PD Control of a Quadrotor VTOL Vehicle,” in *IEEE Intelligent Vehicles Symposium*, 2007, pp. 894–899.

[47] J. Li and Y. Li, “Dynamic analysis and PID control for a quadrotor,” in *IEEE International Conference on Mechatronics and Automation*. IEEE, 2011, pp. 573–578.

[48] L. M. Argentim, W. C. Rezende, P. E. Santos, and R. A. Aguiar, “PID, LQR and LQR-PID on a quadcopter platform,” in *International Conference on Informatics, Electronics and Vision (ICIEV)*, 2013, pp. 1–6.

[49] A. P. Aguiar and J. P. Hespanha, “Trajectory-tracking and path-following of underactuated autonomous vehicles with parametric modeling uncertainty,” *IEEE Transactions on Automatic Control*, vol. 52, no. 8, pp. 1362–1379, 2007.

[50] L. Lapierre and D. Soetanto, “Nonlinear path-following control of an AUV,” *Ocean engineering*, vol. 34, no. 11-12, pp. 1734–1744, 2007.

[51] A. Alessandretti, A. P. Aguiar, and C. N. Jones, “Trajectory-tracking and path-following controllers for constrained underactuated vehicles using model predictive control,” in *European Control Conference (ECC)*, 2013, pp. 1371–1376.

[52] “Shape interrogation for computer aided design and manufacturing,” https://web.mit.edu/hyperbook/Patrikalakis-Maekawa-Cho/, accessed: 2023-05-21.

[53] O. M. Carrasco-Zevallos, C. Viehland, B. Keller, R. P. McNabb, A. N. Kuo, and J. A. Izatt, “Constant linear velocity spiral scanning for near video rate 4D OCT ophthalmic and surgical imaging with isotropic transverse sampling,” *Biomedical optics express*, vol. 9, no. 10, pp. 5052–5070, 2018.

[54] R. Ribeiro, D. Silvestre, and C. Silvestre, “Decentralized control for multi-agent missions based on flocking rules,” in *CONTROLO 2020*, J. A. Gonçalves, M. Braz-César, and J. P. Coelho, Eds. Cham: Springer International Publishing, 2021, pp. 445–454.

[55] ——, “A rendezvous algorithm for multi-agent systems in disconnected network topologies,” in *2020 28th Mediterranean Conference on Control and Automation (MED)*, 2020, pp. 592–597.

[56] R. T. Rodrigues, A. P. Aguiar, and A. Pascoal, “A coverage planner for AUVs using B-splines,” in *IEEE/OES Autonomous Underwater Vehicle Workshop (AUV)*, 2018, pp. 1–6.

[57] M. F. Jacinto, “Cooperative motion control of aerial and marine vehicles for environmental applications,” Master’s thesis, Instituto Superior Técnico, 2021.

A

Uniform Cubic B-Splines

This appendix provides a detailed explanation of obtaining Equation 3.12. The following properties are important in this process:

- **Non-negativity:** $B_{i,j}$ is always non-negative.
- **Local Support:** The basis function $B_{i,j}$ is a non-zero polynomial for $\gamma \in [\gamma_i, \gamma_{i+j+1}]$.
- **Partition of unity:** From the recursive relation defined in Equations (3.3) and (3.4), for any valid γ, the non-null B-spline functions are positive and add up to 1.

Based on these properties, it can be concluded that a uniform cubic B-spline with 4 control points ($n = 3$) is correctly defined only for $\gamma \in [4, 5[$. Beyond this range, the partition of unity property is not satisfied.

Figure A.1 illustrates an auxiliary scheme for computing the basis functions of our B-spline. The green functions B_{i_j} represent the functions that are non-zero within the interval $\gamma \in [3, 4[$, as defined in Equation 3.3. Computing the curve as in Equation (3.2):

$$B_{0,3} = \frac{1}{6}(4-\gamma)^3 \tag{A.1}$$

$$B_{1,3} = \frac{1}{6}(\gamma-1)(4-\gamma)^2 + \frac{1}{6}(5-\gamma)(\gamma-2)(4-\gamma) + \frac{1}{6}(5-\gamma)^2(\gamma-3) \tag{A.2}$$

$$B_{2,3} = \frac{1}{6}(\gamma-2)^2(4-\gamma) + \frac{1}{6}(\gamma-2)(5-\gamma)(\gamma-3) + \frac{1}{6}(6-\gamma)(\gamma-3)^2 \tag{A.3}$$

$$B_{3,3} = \frac{1}{6}(\gamma-3)^3 \tag{A.4}$$

Let us now introduce a change of variable, $\gamma' = \gamma - 3$:

$$B_{0,3} = \frac{1}{6}(1-\gamma')^3 \tag{A.5}$$

$$B_{1,3} = \frac{1}{6}(\gamma'+2)(1-\gamma')^2 + \frac{1}{6}(2-\gamma')(\gamma'+1)(1-\gamma') + \frac{1}{6}\gamma(2-\gamma')^2 \tag{A.6}$$

$$B_{2,3} = \frac{1}{6}(\gamma'+1)^2(1-\gamma') + \frac{1}{6}\gamma'(\gamma'+1)(2-\gamma') + \frac{1}{6}\gamma^2(3-\gamma') \tag{A.7}$$

$$B_{3,3} = \frac{1}{6}\gamma'^3 \tag{A.8}$$

where $\gamma' \in [0, 1[$. By performing some algebraic manipulation, we can obtain an equivalent definition to the one given in Equation 3.10.

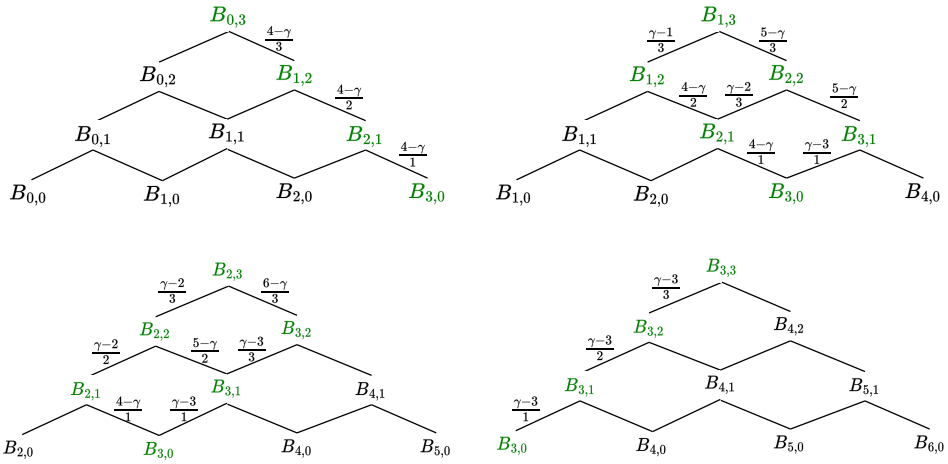


Figure A.1: Auxiliary scheme to compute the basis functions of a uniform cubic B-spline with 4 control points

Let's consider the addition of a new control point, P_4, which introduces a new segment in the B-spline. This segment is defined by the points P_1, P_2, P_3, P_4, and the B-spline curve can be expressed as:

$$\mathbf{C}(\gamma) = B_{1,3}(\gamma)P_1 + B_{2,3}(\gamma)P_2 + B_{3,3}(\gamma)P_3 + B_{4,3}(\gamma)P_4, \tag{A.9}$$

where $\gamma \in [4, 5[$. In A.2, an auxiliary diagram is presented to visualize the computation of the basis

functions for the second segment. By applying the same process as before and introducing the change of variable $\gamma' = \gamma - 4$, such that $\gamma' \in [0, 1[$, we can observe that all the basis functions used to define each segment remain the same as those defined in 3.10.

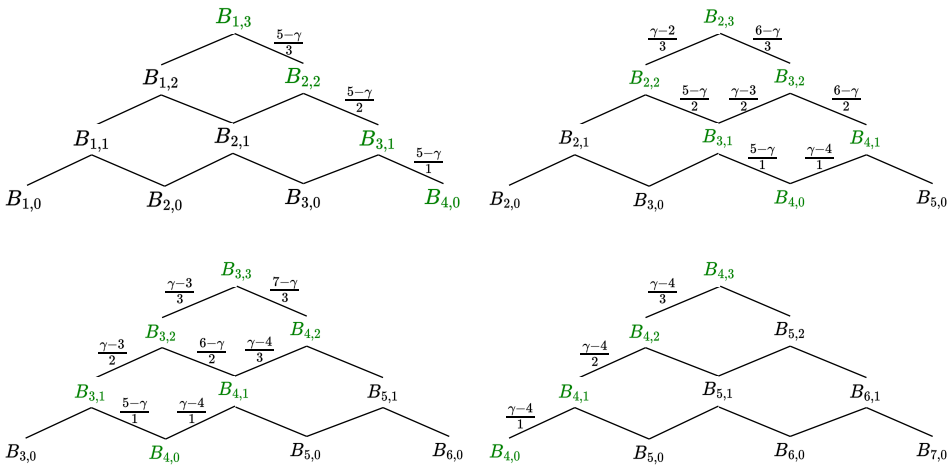


Figure A.2: Auxiliary scheme to compute the basis functions of the second segment of a uniform cubic B-spline with 5 control points

In conclusion, by using a parameterized variable $\gamma' = \gamma - \lfloor x \rfloor$ within the interval $[0, 1[$ and employing the same basis function, we can effectively compute the current segment of the B-spline curve for a generic number of control points. This allows for a consistent and convenient representation of the B-spline, regardless of the number of control points or the specific segment being evaluated.