



POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

Optimisation of UAV trajectories for target detection using a prob- abilistic approach

TESI DI LAUREA MAGISTRALE IN
SPACE ENGINEERING - INGEGNERIA SPAZIALE

Author: **Giovanni Boscolo Boscoletto**

Student ID: 991826

Advisor: Prof. Davide Invernizzi

Co-advisors: Hélène Piet-Lahanier, Michel Kieffer

Academic Year: 2024-2025

Abstract in lingua italiana

Questa tesi ha come scopo il design e l'ottimizzazione delle traiettorie degli UAVs (Unmanned Aerial Vehicles) per problemi di rilevamento di obiettivi. In particolare, si considera un UAV alla ricerca di un obiettivo fisso o mobile. L'area della missione è nota e gli ostacoli sono modellati come cerchi. Le traiettorie sono descritte matematicamente dai polinomi di Bernstein. Le caratteristiche e le proprietà di questi polinomi permettono di creare un framework di ottimizzazione per la generazione di traiettorie. Queste traiettorie, evitando gli ostacoli, minimizzano un criterio come il tempo di missione o la lunghezza della traiettoria. In questo lavoro di tesi la componente per il rilevamento degli obiettivi è aggiunta al criterio dell'ottimizzazione. In particolare, per quanto riguarda l'utilizzo delle informazioni a priori sugli obiettivi e le caratteristiche del processo di rilevamento. La tesi si limita alla costruzione di un metodo di pianificazione offline (globale). In preparazione all'implementazione di un framework per la generazione di traiettorie online (locali) una descrizione probabilistica viene incorporata utilizzando un metodo di aggiornamento bayesiano. Un vantaggio chiave dell'uso dei polinomi di Bernstein è la capacità di ottenere traiettorie "lisce" e dinamicamente realizzabili, al contrario di molti metodi di path planning che producono un insieme di waypoints non direttamente utilizzabili come traiettorie di riferimento. Tuttavia, la scelta del grado del polinomio di Bernstein non è banale e dipende fortemente dall'ambiente e dalla traiettoria scelta come guess iniziale. I risultati ottenuti mostrano come la traiettoria sia influenzata dalla conoscenza preliminare della probabile posizione di un obiettivo. La stima della posizione dell'obiettivo mostra una precisione promettente, riflettendo le prestazioni del sensore.

Parole chiave: pianificazione di traiettoria, UAV, ottimizzazione, polinomi di Bernstein, conoscenza a priori, evitamento degli ostacoli, stima bayesiana

Abstract

The aim of this thesis is the design and optimisation of Unmanned Aerial Vehicle (UAV) trajectories for target detection problems. In particular, the scenario involves a UAV searching for either a stationary or moving target. The mission area is known, and obstacles are modelled as circles. The trajectories are mathematically represented using Bernstein polynomials. The properties and characteristics of these polynomials enable the formulation of an optimisation framework for trajectory generation. These trajectories avoid obstacles while minimising a given cost function, such as mission time or path length. In this work, the target detection component is incorporated into the optimisation framework. Specifically, this includes the use of prior information on targets and the characteristics of the detection process. The proposed method is limited to offline (global) planning. In preparation for the implementation of an online (local) trajectory generation framework, a probabilistic description is introduced using a Bayesian update method. A key advantage of employing Bernstein polynomials lies in their ability to generate smooth and dynamically feasible trajectories, in contrast to many path planning methods that produce sequences of waypoints which are not directly usable as reference trajectories. However, the selection of the polynomial degree is non-trivial and heavily dependent on both the environment and the initial guess trajectory. The results demonstrate how prior knowledge of the likely target location influences the resulting trajectory. The estimated target position exhibits promising accuracy, reflecting the performance of the sensor.

Keywords: trajectory planning, UAV, optimisation, Bernstein polynomials, prior knowledge, obstacle avoidance, Bayesian estimation

Contents

Abstract in lingua italiana	i
Abstract	iii
Contents	v
Introduction	1
1 Path Planning State of the Art	5
1.1 Graphed-Based Search	6
1.1.1 Deterministic Search	6
1.1.2 Stochastic Search	8
1.1.3 Graph-Based Search Limitations	10
1.2 Path Planning without Graph Description	10
1.2.1 Artificial Potential Field Approach	10
1.2.2 Particle Swarm Optimisation	11
1.3 Search for Trajectory using Optimal Control	11
1.3.1 Optimal Search Problem	12
1.4 Accounting for Information Uncertainty	14
1.5 Multi-Agent Path Finding (MAPF)	16
2 Trajectory Representation	19
2.1 Bernstein Polynomials	19
2.2 Properties	20
2.3 Algorithms	24
2.4 Advantages of Trajectory Representation using BP	26
3 Problem Description	27
3.1 UAV	27
3.1.1 Dynamics	27

3.1.2	Sensor Modelling	31
3.2	Targets	35
3.2.1	Known Fixed Target	35
3.2.2	Partially Known Fixed Target	36
3.2.3	Unknown Moving Target	37
3.3	Environment and Obstacles	38
4	Estimation and Trajectory Optimisation Methods	41
4.1	Estimation of Target Location	41
4.1.1	Correction Step	42
4.1.2	Prediction Step	44
4.2	Trajectory Optimisation	47
4.2.1	Objective Function	48
4.2.2	Initial Guess of the Optimisation	49
5	Results	55
5.1	Optimal Problem Formulation	55
5.2	Parameters and variables choice	56
5.2.1	Bernstein Polynomials Degree	56
5.2.2	Temporal and Spatial discretisation	58
5.3	Simulations	59
5.3.1	Obstacle avoidance	60
5.3.2	Target detection	65
5.4	Limitations	77
6	Conclusions	79
6.1	Conclusions	79
6.2	Future Works	81
	Bibliography	85
A	Appendix A	91
B	Appendix B	93
C	Appendix C	97

Introduction

Research as an activity is something humans have been doing for millennia. Since antiquity, nomadic populations have been looking for the best migration paths. Primitive men also developed hunting techniques, whether hunting alone or in groups, using their ability to perceive or animals with advanced physical, hearing, and smelling abilities. As the first human communities formed, one of the oldest activities, war, began. There has always been the necessity for better organisation in terms of strategies to deal with new enemy tactics and technologies. The strategy has always been closely linked to, and sometimes ahead of, advancements in weapon technology. But it was only with modern warfare, and in particular anti-submarine warfare, that target search has been rigorously studied from a scientific and mathematical point of view. During World War II, the foundation for the field of *search theory* was laid by the U.S. Navy's Antisubmarine Warfare Operations Research Group (ASWORG) in 1942 in response to the German submarine threat in the Atlantic. Koopman's work [36] presents the basics of the optimal search problem. On one side, there is the *searcher* with its capabilities, detection instrument characteristics, position, and motion. On the other side, the *target* with its position and motion. For the nature of a real situation, probability has to come into play to model the problem. Three key points are reported in the first chapter of Koopman's work and are the main concepts that are studied in the whole technical report:

1. *Contact* or *detection* problem: determining the probability of successfully detecting the target.
2. *Track* problem: analysing the path and motion of the searcher relative to the estimated location and movement of the target.
3. *"Force" requirement*: optimizing search effectiveness within the constraints of limited available resources.

In [19], a review of the search theory is presented. After a summary of Koopman's works and main contributions, the author concentrates the report on the application of the Search and Rescue (SAR) problem. It is defined as the search for and provision of aid to people who are in distress or imminent danger. SAR operations benefit significantly

from advancements in search theory, especially in time-critical scenarios involving environments that are hazardous or inaccessible to humans, where an optimized strategy can be crucial. Autonomous vehicles, particularly Unmanned Aerial Vehicles (UAVs), present a promising solution to these challenges. With advancements in technology, UAVs have become increasingly capable, offering the potential for coordinated fleet operations that can communicate and cover extensive areas. [41] reports an extensive survey on the UAV applications for SAR. Some of the advantages of the exploitation of drones for these situations are the ease of deployment, high mobility, low maintenance cost, and the ability to hover in areas where human intervention is really difficult and dangerous. In addition, thanks to the advancement in different technological fields, from high-precision sensors to efficient machine learning algorithms, a large amount of data can be collected by drones and analysed on board or by a decentralised computer. The authors, after a classification of different UAVs based on the design, focus their attention on the operations themselves. Task assignment, path planning, and collision avoidance are analysed.

The search and track problem can be viewed as a generalization of Search and Rescue operations. In this broader context, the target may be either cooperative or uncooperative. As a result, once the target is detected, it becomes necessary to analyse the searcher's path and motion relative to the estimated location and movement of the target. In the most general formulation, this problem is not trivial to solve. Because of its complex nature, it is convenient to subdivide it into sub-problems, such as task assignment, trajectory planning, control techniques, hardware and software design, and implementation.

This thesis addresses UAV trajectory design and optimisation for target detection problems. The primary objective is to generate trajectories that integrate various essential mission aspects, including UAV dynamics, sensor characteristics, control constraints, obstacle avoidance, and information about the zone of interest. The latter may be derived from prior knowledge or through communication with other UAV fleet members. Kragelund et al. [37] incorporate the sensor detection rate and potential prior information about the target position into the optimisation design. The problem they address, known as the Mine Countermeasures Mission (MCM), focuses solely on trajectory planning and does not incorporate probabilistic updates.

Our study aims to develop the basis of a new framework that will support online trajectory generation. The main idea is to optimise the trajectory over a reduced time horizon and use measurements to re-plan the mission and to update information through Bayesian estimation, as done in [23]. In particular, this work focuses on building the first fundamental block of trajectory design over a well-defined horizon.

Formulating an optimal control problem is especially effective for motion planning in applications where the trajectory must minimise a cost function while adhering to a complex set of vehicle and problem constraints. However, deriving a closed-form solution for a nonlinear constrained optimal control problem is challenging. To address this, a wide range of discretisation methods have been developed, [47], [54] and [51], but they present two main drawbacks. Constraints can only be enforced at the discretisation nodes, not at every time instant. Additionally, a high number of discretisation points significantly increases computational time, making online implementation impractical. To address these challenges, we build upon the work of Cichella et al. [8], who propose a direct method using Bernstein approximation. Bernstein approximants and their derivatives converge uniformly to the functions they approximate. Furthermore, these polynomials, also known as Bézier curves, enable computationally efficient algorithms for evaluating constraints such as minimum and maximum velocity, acceleration, and minimum distance between paths. These constraints can be applied across the entire trajectory, rather than just at discretisation points.

In Chapter 1, a survey on trajectory planning methods is conducted. This chapter covers the foundational concepts of the field as well as the most recent developments, analysing the strengths and limitations of each methodological approach.

In Chapter 2, an introduction to the mathematical theory behind the chosen trajectory representation is provided. Bernstein polynomials are the candidates for performing this step. Their main properties and algorithms are provided to facilitate the understanding of the modelling and implementation of the problem later on.

In Chapter 3, the main elements constituting the problem are modelled. This includes the UAV's dynamics and control input constraints and covers the transition from their definitions to their reformulation using Bernstein polynomials. Subsequently, the drone's onboard sensor and the target information are modelled. Different scenarios are presented. Finally, the modelling of the environment is analysed, with a particular focus on the description of obstacles.

In Chapter 4, the estimation of the target's position is addressed. It involves alternating between correction and prediction steps. The first step uses the measurements provided by the sensor when they are available. The second one is applied only under the assumption of a moving target, and it is used to propagate the information over time. The second part of this chapter is dedicated to the definition of two important elements in the trajectory optimisation framework. Firstly, the objective function is analytically derived and rewritten using the elements presented so far and through Bernstein polynomials.

Finally, some considerations on the choice of the initial guess trajectory are reported.

In Chapter 5, a summary of the mathematical formulation of the optimal trajectory generation problem is first provided, both in continuous form and in the discrete form necessary for numerical implementation. The main body of the chapter is dedicated to analysing the simulation results, with a focus on both the effectiveness of obstacle avoidance and the accuracy of target detection based on prior information about the area of interest.

Chapter 6 concludes the thesis by summarising the main points, highlighting both strengths and limitations of the approach. Finally, the main directions for future developments are presented.

1 | Path Planning State of the Art

Literature on the topic of trajectory planning is immense. The complexity of this problem necessitates addressing it by focusing on one or a few aspects at a time. For this reason, each work considers a specific application under various hypotheses. Before going into the details of the different methods proposed by the scientific community, a recall of the principal problem characteristics is necessary for a better understanding of the strengths and limitations.

The primary distinction lies between works that focus on offline trajectory planning and those that enable reactive implementation of new local trajectories, based on searching and tracking strategies as well as avoidance of unknown obstacles. In the first phase of the mission, it is necessary to plan the trajectory with the available prior knowledge of the environment and the targets. As offline planning concerns the definition of the trajectory for the whole mission, it is often referred to as global. The second phase, which concerns variations induced by a priori unknown events, e.g. new obstacles or variations of environmental conditions, is thus called local. As a UAV begins to fly and follow its global trajectory, this trajectory dynamically evolves in response to the environment, particularly for target detection and based on information received through external communications. According to this new information, an updated trajectory is computed over a small horizon of time. This process is repeated till the mission is completed in terms of objectives or maximum time. In both situations, the goal is to generate a reference trajectory, in terms of states and control inputs, that the UAV can follow, respecting a series of constraints. The first constraint to satisfy is the one linked to the definition of the dynamics of the UAV and the saturation of control inputs. It is desirable to respect them to obtain a feasible trajectory with expected performance characteristics. The determination should also take into account a priori information, such as available knowledge about the potential locations or absences of obstacles and targets, as well as the level of confidence associated with this information. For this reason, uncertainty and probability concepts have been introduced into trajectory design. Finally, the study of organised fleets of drones or vehicles in general is gaining more and more attention.

There are various classifications of all existing methods to date, and the classification presented below can be considered simplified. For a complete and recent review of UAV path planning, it is highly recommended to read the following paper [2].

1.1. Graphed-Based Search

The graph-based search methods use a discretisation of the area of interest. The graph representation is well-suited because of its structure, comprising nodes and edges. Nodes correspond to vehicle potential positions and the edges represent the cost, or weight, to pass from one position to another.

1.1.1. Deterministic Search

The first family can be called deterministic. In fact, given a certain environment, an initial position, a target position, and a set of rules, the behaviour is entirely predictable. These algorithms follow precise rules for exploring paths, evaluating costs, and making decisions at each step of the path-finding process. Generally, in path planning problems for trajectory generation, the zone is uniformly discretised. The nodes are equally spaced and edges are selected according to the kind of movements allowed by the vehicle. Each node has some reachable neighbourhood nodes. The two basic models are with 4 neighbours (2 horizontal and 2 vertical movements) and with 8 neighbours (also the 4 diagonal directions).

The first formulation was done by Dijkstra in his famous paper “*A note on two problems in connexion with graph*” in 1959, [11]. The first problem described is about a tree construction of minimum total length between the n nodes coming from the discretisation. Remember that a tree is a graph with one and only one path between every two nodes. The second problem aimed to find the path of minimum total length between two given nodes. Dijkstra formulated two algorithms to face these problems. It is necessary to give a weight to each edge, or transition. It could reflect the length distance to move from one point to the other or some different criteria. Therefore, the environment, in terms of obstacles, has to be completely known when applying this kind of resolution. The most important feature that characterises these two algorithms is the computation of all possible paths. Once all the possible paths are computed, they are compared looking at the total cost function and the best path is finally found. We call $g(n)$ the path cost from the start node to the node n . This in terms of computational cost is very demanding and increases significantly with the number of nodes considered.

However, knowledge of the environment can help focus the search for the optimum path in a specific direction: informed search. This will decrease the computational cost since a smaller area will be analysed. The idea behind the best-first search is that a node is selected for the expansion of the solution tree based on an evaluation function $f(n)$, [52]. The node with the lowest evaluation is expanded first. In general, the most efficient algorithms of this family include as a component of the evaluation function $f(n)$, a heuristic function $h(n)$, where $h(n)$ provides efficiently an estimate of the cost of the cheapest path from the state at node n to the goal state. The heuristic can be modelled with Euclidean, Manhattan or another kind of distance. A^* is the most common algorithm in this family. It uses as evaluation function $f(n) = g(n) + h(n)$. From a time complexity of $\mathcal{O}(n^2)$ for Dijkstra's it diminishes to $\mathcal{O}(n \log(n))$ for A^* , [53].

A major limit for both of these two algorithms is the fact that they cannot take into account the possibility of encountering obstacles not known before the departure from the starting point. A solution to this problem comes from the so-called Dynamic Algorithms. A classification can be found in [50]. *Fully dynamic* algorithms can handle both insertions and deletions, instead *partially dynamics* ones handle only one at a time: *incremental*, only insertions, and *decremental*, only deletions. In a more general way, it can be said that the edges change their costs during the "crossing". In a real situation, this happens because of the presence of onboard sensors that allow the UAV to collect new information about the environment. *Lifelong planning A^* (LPA^*)* [35] combines the advantages of a incremental search, *DynamicsWSF-FP* [48], and the heuristic search of A^* . The first search is the same as that of a version of A^* . When an obstacle is found, or more in general when edge values change, a complete re-computation of the best plan can be wasteful. For this reason, the algorithm reuses those parts of the previous search tree that are identical to the new one.

Another dynamic algorithm is D^* . The name was chosen because it is based on the same structure of A^* , but it is dynamic; it can handle changes in edge cost. The concept of D^* method is very similar to that one of LPA^* . The main difference is that D^* calculates all the paths from the goal location to all the nodes of the considered area, [60]. When an obstacle is encountered the algorithm repairs the path only from the obstacle back to the robot's position. Koenig et al. propose in [34] another variant of D^* called D^* *Lite*. Built on LPA^* , it implements the same navigation strategy as D^* but is algorithmically different. D^* *Lite* is substantially faster than D^* , it uses only one tie-breaking criterion when comparing priorities. LPA^* repeatedly determines shortest paths between the start vertex and the goal vertex as the edge costs of a graph change. Instead, D^* *Lite* repeatedly determines the shortest path between the current vertex of the robot and the goal vertex

as the edge costs of a graph change while the robot moves towards the goal vertex.

All the approaches presented so far are limited by the discretisation, not only in terms of the distance between nodes but also for the possibility of transitions. If we consider a node with 8 neighbours, the only admitted transitions are along directions multiple of $\pi/4$. The *D* Field algorithm* [16] extends *D** and *D* Lite* to use linear interpolation to efficiently produce low-cost paths that eliminate unnecessary turning. As the resolution of the uniform grid increases, the solutions returned by the algorithm improve, approaching true optimal paths. The algorithm *Multiresolution Field D**, an extension of *Field D** able to plan over non-uniform resolution grids, has also been presented herein.

All these algorithms can be computationally expensive, especially in large-scale environments or scenarios with intricate obstacle configurations. This can limit real-time applicability.

1.1.2. Stochastic Search

The second family of path planning methods is called stochastic. This is because the choice of the graph nodes does not follow a predefined pattern, but is randomly generated. Some of the key advantages are the effectiveness in high-dimensional configuration space and the easy adaptability in complex terrain with irregular obstacles. This adaptability makes them particularly suitable for real-world applications in robotics and autonomous systems. The two best-known methods are the *Probabilistic Road Map (PRM)* and the *Rapidly-exploring Random Tree (RRT)* with its variants.

It is important to point out that *PRM* considers stationary obstacles. However, it is worthwhile to have a proper understanding of how it works, [29]. In the first phase, called *learning* phase, a probabilistic roadmap is built. It is an undirected graph where nodes represent collision-free configurations, and edges represent feasible paths connecting these configurations. The nodes are randomly selected from collision-free states, using either a uniform or other probability distributions. Each selected node then connects to all feasible neighbours within a specified range. In the second phase, *query*, the roadmap is used to solve individual path-planning problems. Given a start and goal configuration in the input environment, the method first tries to connect the two configurations to two nodes of the graph describing the Probability Map. Then, a path from these two last nodes is calculated with a deterministic search in the graph.

RRT appeared first in 1998 in S. M. LaValle's work [39]. "*The unique advantage of RRTs is that they can be directly applied to nonholonomic and kinodynamic planning. This advantage stems from the fact that RRTs do not require any connections to be made*

between pairs of configurations (or states), while probabilistic roadmaps typically require tens of thousands of connections." At the beginning, the tree consists of a single node representing the starting configuration. Then an expansion is done until a predefined number of iterations or until a goal condition is reached, the final configuration or set of configurations. In each iteration, a new random configuration is generated from the state space. The nearest node in the tree is found and a new point is calculated moving from the tree towards the random node by a step parameter. If the new node is collision-free then it is added to the original tree. The update of the tree is done until one of the two above-mentioned conditions is satisfied. To fasten the algorithm every n iteration it is possible to generate as random node the goal location. In this way, the algorithm will be driven easier toward the goal. One big limitation is that the path found may not always be the shortest path. Then, depending on the parameters and environment, *RRT* may require many iterations to find a feasible path.

A lot of variants and extensions of the basic *RRT* have been presented. Some of them are described below with their main characteristics. *RRT-connect* extends the basic algorithm by simultaneously constructing two trees rooted at the start and goal configurations, [38]. These trees explore the configuration space around them and gradually approach each other using a simple heuristic approach. Instead of extending an RRT by a single predefined maximum step, this heuristic iterates the extension step until it reaches the other tree or encounters an obstacle. *RRT** is an optimised version of *RRT*, [28]. Two key additions to the algorithm result in significantly different results. First, *RRT** records the distance from each vertex to its parent vertex. After the closest node is found in the graph, a neighbourhood of vertices in a fixed radius from the new node is examined. If a node with a cheaper cost than the proximal node is found, the cheaper node replaces the proximal node. The second difference *RRT** adds is the rewiring of the tree. After a vertex has been connected to the cheapest neighbour, the neighbours are again examined. Neighbours are checked if being rewired to the newly added vertex will make their cost decrease. If the cost does indeed decrease, the neighbour is rewired to the newly added vertex. These features make the path more smooth. Other two extensions are *RRT[#]* [3] and *RRT^X*[44]. [45] introduces in the planner the presence of uncertainties and designs a robust path planner, assuming that all the uncertain quantities are bounded with known bounds. Starting from some uncertain initial configuration, represented by a set, the planner aims at driving the vehicle to a final configuration set.

1.1.3. Graph-Based Search Limitations

When modelling trajectories, one major limitation is the discretisation process. Typically, trajectories need to be interpolated using polynomials. However, this interpolation can become quite challenging in complex environments. The complexities of the environment often make it difficult to choose and fit the right polynomial functions, so the task is not simple.

1.2. Path Planning without Graph Description

Due to the requirements of defining a discretisation of the potential trajectories in graph-based search, other approaches have been suggested in the literature.

1.2.1. Artificial Potential Field Approach

A family of methods originates from the concept first proposed by [30]. This technique draws inspiration from physics, where particles move under the influence of potential fields generated by attractive and repulsive forces. In robotics, this concept is adapted to guide a robot from its initial position to a goal while avoiding obstacles in its path. An artificial potential field (APF) is added to the gravity potential energy. This APF is composed of two parts. One represents the attractive field to the target position and the other a repulsive field to avoid the obstacles. The robot navigates by following the gradient of the total potential field. The most significant drawback of the APF method is the tendency to get trapped in local minima. This occurs when the combined attractive and repulsive forces balance each other out at a point that is not the goal, preventing the robot from progressing towards its destination. Another drawback is that in narrow passages or near obstacles, the robot may experience oscillations due to the alternating influence of attractive and repulsive forces. This can lead to inefficient or even failed path planning.

Improvements to overcome these situations come from the methods called *Hybrid Approaches*. They combine the advantages of global path planning algorithms with the local navigation efficiency of the APF method. In [1] the presented method exploits the global search capability of the A^* algorithm to generate a rough global path that considers the overall layout of the environment. Then the APF method refines this path locally, helping the robot navigate around obstacles smoothly. Another family is the *Adaptive Artificial Potential Field*. The key improvement is that they dynamically adjust the parameters of the potential field based on the robot's current state and environment, in real-time.

The adaptive nature allows the robot to modify its path planning strategy dynamically to handle unexpected obstacles and changes in the environment, [66].

1.2.2. Particle Swarm Optimisation

A different family of algorithms comes from Particle Swarm Optimisation (PSO), a popular optimisation algorithm inspired by the social behaviour of birds flocking or fish schooling. PSO is effective at finding global optima, reducing the likelihood of getting stuck in local minima, which, as seen, is a problem for the basic implementation of the APF method. An improved PSO algorithm is proposed in [24] applied to three-dimensional path planning for fixed-wing UAVs. The environment considered is static and only 1 UAV is considered. Instead, [61] addresses the path planning for multiple UAVs operating in obstacle-rich environments with both static and dynamic obstacles.

1.3. Search for Trajectory using Optimal Control

Optimal control theory is a mathematical framework used to determine the best possible control actions for a dynamic system over time. In path planning, this theory is applied to devise trajectories that optimise specific performance criteria while satisfying to various constraints. It involves the formulation of an optimisation problem. This problem typically aims to minimise or maximise a cost function, which represents the performance measure of interest. For instance, in a path planning context, the cost function might account for factors such as travel length and duration, energy consumption, safety, and smoothness of the trajectory. The process begins with the definition of the system dynamics, which describes how the state of the system evolves in response to control inputs. In the case of a robot or an unmanned aerial vehicle (UAV), these dynamics are often represented by differential equations that capture the kinematics and dynamics of the vehicle. The next step involves specifying the constraints that the system must satisfy. These constraints can be of various types, including state, control and environmental constraints.

Model Predictive Control (MPC) is a practical implementation of optimal control theory, particularly well-suited for real-time path planning. MPC operates by solving an optimisation problem at each control step to determine the best control inputs that minimise a cost function while adhering to constraints over a time horizon. This approach relies on a system model to predict future states and make informed control decisions, using a dynamic model that captures the UAV's kinematics and dynamics. The quality of the prediction must be consistent with the selected time horizon. MPC enforces

state and input constraints to ensure the generation of feasible trajectories and can also incorporate environmental constraints, such as obstacles, into the optimisation process. By utilising real-time feedback to adjust control inputs based on the UAV's current state and the surrounding environment, MPC facilitates dynamic adaptation to changes and uncertainties during the mission. In [4], a framework for the cooperative guidance of a UAV fleet using the MPC approach is introduced. The paper tackles common challenges in cooperative control, including collision and obstacle avoidance, formation flying, and new area exploration. It also details the cost functions related to each aspect of these missions. In particular, three distinct mission scenarios are presented. The first scenario involves guiding a fleet of UAVs towards specified objectives, represented by waypoints while avoiding collisions with obstacles and other vehicles. The second scenario focuses on an exploration mission, utilising both a grid allocation method and a cost-oriented approach. Lastly, the third scenario features formation flying, where the UAVs follow a virtual geometrical structure. Regarding this thesis, the most relevant problem is the exploration using a cost-oriented approach. Each vehicle defines its trajectory to achieve the cooperative mission objectives, which include maximising the cumulative area covered and reaching exit points by the end of the mission.

In [7], a new method for path planning is proposed. It is based on the Artificial Potential Field method but with the addition of a control force. The problem is then reformulated from constrained to unconstrained, and through functional optimisation, an optimal control problem is obtained.

1.3.1. Optimal Search Problem

Some researchers have recently focused on studying the optimal search problem. This problem, first formalised by Koopman as we have seen, has become interesting again due to newly available numerical computation methods. Especially, in the search for objectives that involve motion uncertainty characterised by unknown or partially known parameters [65]. The problem is to optimise the probability of detection of a target with uncertain parameters, given the detection characteristics of the sensors and some constraints in terms of seeker trajectories. In the next reference works the detection model used is the one proposed and characterised for the first time by Koopman in [36], the exponential detection model. This model is the one that is taken as a reference and applied in this thesis. A more detailed description is done in Chapter 4. In [65], the situation considered is a multi-agent optimal search problem. Four searchers have to survey a channel where a target is moving from right to left with constant velocity. The target follows a conditionally deterministic motion. This motion is characterised by the definition of

one parameter. Once this parameter is known, the entire motion becomes deterministic. [46] analyses and studies the same kind of optimal control problem, where the objective function evaluation implies an integration over stochastic parameters and also integration over the time domain. Another important paper that was studied and that contributed to the present work is [37]. The application of the proposed framework is designing trajectories for autonomous underwater vehicles to conduct Mine Countermeasure Missions (MCM). Even if the application field is quite different it is possible to see some common aspects to the UAV search problem. Especially, the two main phases of an MCM are detection, classification and localisation (DCL) and re-acquisition and identification (RID). The first, when there is no prior information about the target locations, is considered as a coverage problem. Instead, in the second phase the trajectories are planned by exploiting prior knowledge about the expected target location. To go deeper into the details of [37] see [17] for the formulation and theory and [18] for the algorithms and computations.

This thesis takes the just mentioned works and expands them under three main aspects. The addition of obstacles, the parametric description of the optimal control problem and, the propagation of uncertainties regarding the knowledge of a target's presence. For this last point see Section 1.4. For what concerns the parametrisation of the problem the Bernstein polynomials (BPs) are introduced. The optimisation parameters are the control points of the BPs. The theory behind this and the all mathematical framework is reported in Chapter 2. The paper [8] is taken as a reference to describe the UAV motion dynamics and to set up the control and environmental constraints, utilising the properties of BPs. In [31] the toolkit BeBOT (Bernstein/Bézier Optimal Trajectories) is presented. It is a toolkit designed for the generation of trajectories for autonomous systems using BPs. [10] and [33] take up and expand with more detail [8]. In [32] the target monitoring task is added to the framework previously introduced by the other works. Especially, the problem can be formulated as: "Consider a mission involving m UAVs. Find m trajectories to be tracked by the UAVs over the time interval $[t_0, t_f]$, that minimise a cost function, $J()$, subject to boundary conditions, feasible flight constraints, and mission-specific constraints.". The formulation takes into account the possibility of re-planning during the mission. For this reason, the problem is defined over a generic time interval $[t_0, t_f]$ and then sequential planning is considered and formulated. Instead, [64] offers a motion planning solution for target localisation that utilises BP basis functions to estimate the probability distribution of the target's trajectory. Also, this solution involves re-planning events and uses all the prior information to update the UAV's trajectory. To evaluate the performance of target localisation the determinant of the Fisher information matrix (FIM) is considered. The target's location is unknown, with only estimates available at

specific discrete times.

1.4. Accounting for Information Uncertainty

In a trajectory generation problem, uncertainty can affect various aspects. For example, this includes incorrect knowledge about the area in terms of obstacles and inaccuracies in the drone's dynamics. A significant source of uncertainty comes from sensor measurement noise. It is particularly this type of uncertainty that will be considered further. This work aims to create a framework that is also capable of updating the measurement uncertainties as the mission unfolds. In the literature, there are two ways to represent uncertainties: as acceptable bounds on variations or using a probability distribution. The first approach led to bounded error estimation or membership set estimation, [55], while the second is particularly used in Bayesian estimation. For the continuation of the work, the second will be used. Nevertheless, it is of general interest to have the guidelines of both in mind.

Set-membership estimation techniques consider noises and uncertainties in the measurements and dynamic systems belonging to bounded sets. The evaluated state estimate comprises sets rather than points. No assumptions are made regarding the probability density functions of the noises and uncertainties. The set estimate contains the current states of the system with certainty if the assumption of bounded noises and measurements is verified and each point within the set estimate is a possible candidate for the actual system state. In [26], [25] and [49] a distributed set-membership estimation and control scheme is applied for a fleet of UAVs that want to search and track mobile targets in a zone of interest. Multiple sets are initialised and updated based on the measurements and information received from neighbours. These include a set for each detected and identified true target, a set for detected but not yet identified targets, and a set for targets that have not yet been detected. Model predictive control is employed to calculate the UAVs' trajectories in terms of control input. The optimisation function aims to reduce the estimation uncertainty of each set.

Stochastic Bayesian estimation merges the principles of probability theory with Bayesian inference to address complex estimation problems. Stochastic processes model the random behaviour of systems over time or across different scenarios. These processes are described by probabilistic models that capture the variability and uncertainty inherent in the data. Stochasticity introduces randomness into the estimation process. Bayesian estimation involves updating the probability distribution of a parameter or model based on observed data. The process starts with a prior distribution that reflects initial belief or knowledge about the parameter before observing the data. Once data is available, Bayes' theorem

combines the prior distribution with the likelihood of observing the given data under various parameter values to produce a posterior distribution. This posterior distribution represents updated beliefs and is central to Bayesian inference. The main steps of this update process are correction and prediction. They are detailed in Section 4.1. In all the following references, the Probability Map concept is introduced and used. One difference lies in the assumptions about prior distributions, which are based on assumptions. The zone of interest is divided into a discretised grid and for each cell, a value indicating the probability that a target exists in that cell is given. The set of all probabilities is called *Probability Map*. [5] focuses on a robust approach for search operations in the presence of uncertainty. Rather than modelling the imprecise information of the prior probabilities with point value, a Beta distribution is used to be robust to the uncertainty. With this framework, the authors report an analytical expression of the minimum number of looks needed to reach a certain level of confidence in the target's existence in an uncertain environment. The likelihood distribution is represented by a Bernoulli distribution. It is effective because simplifies all the complexities of image processing algorithms and results in the binary result: *target detection* and *no target detection*. [6] extends the previous paper for dynamic targets. To face a more realistic environment, probabilistic target motion is considered and used to update the *uncertain Probability Map*. This additional part of the estimation process is called the prediction step. The prediction and measurement updates are done on the means and variances of each Beta distribution. In the present work instead, the inferences are done for each punctual value of the grid. In [23] the probability expresses the state knowledge of each cell, the cell contains or does not contain a target. Also in this case the likelihood is modelled as Bernoulli distribution and the update step is done with the Bayes' rule. The targets are considered as statics. [42] proposes a multi-agent decentralised search of a Probability Map with communication constraints. Pop-up targets are here modelled and taken into consideration. The arrival rate of pop-targets in each cell is modelled using a Poisson process. [27] introduces a method for constructing a Probability Map of uncertain dynamic environments containing multiple obstacles. The approach assumes that the vehicle has a limited sensor range, thus lacking global information. The Probability Map is continually updated using sensor data and prior statistics of the dynamic environment. Additionally, the paper presents a data association algorithm for indistinguishable obstacles, allowing the vehicle to determine whether a currently detected obstacle has been previously detected.

1.5. Multi-Agent Path Finding (MAPF)

All the previously introduced trajectory design approaches have been initially suggested for a single agent. Depending on the potential organisation of the fleet, whether centralised or distributed (also referred to as decentralised), the single-agent approach may prove insufficient.

In a centralised scheme, actions of all vehicles are determined by a single element utilising all available information. This approach enables the simultaneous coordination and optimisation of the various vehicles' actions, facilitating synchronisation. However, this method requires potentially intensive computation to address the centralised problem, which may limit the number of vehicles that can be managed. It also necessitates robust transmission links to monitor all vehicles effectively. Additionally, the central control element could become a single point of failure if there is no backup to assume its role.

The distributed control instead reduces the computational burden by allowing each vehicle to compute its control input, enhancing scalability and robustness. However, there are two main drawbacks: first, the limited computational capabilities of each vehicle may impede the resolution of complex problems, and second, information about other vehicles may be inaccurate, incomplete, or delayed.

Extensions of these methods have been proposed that account for the potential interaction of the agents. The simultaneous movements of multiple agents to the same position or other types of situations are referred to as conflicts. Several types of conflicts invalidate the solutions found by the different algorithms. A solution is valid for a MAPF problem if it forms a conflict-free joint plan.

MAPF problems can be solved by either optimal or sub-optimal solvers. An optimal solver is expected to provide the optimal solution in terms of trajectory quality criteria and the absence of conflict. In the case of sub-optimal solvers, there is no guarantee of convergence to this optimal solution if it exists.

Optimal solvers can then be divided into search-based solvers or reduced solvers. Search-based methods perform a systematic search over the set of possible paths that different agents can follow. There is an initial family of algorithms based on the concept of A*. Other search-based algorithms correspond to tree-based searches, the main ones being: the Increasing Cost Tree Search (ICTS) [56], Safe-Interval Path Planning (SIPP) [43], and Conflict-Based Search (CBS) [57]. For reduced solvers, the principle is to translate the MAPF problem into a known solution problem such as SAT, Integer Linear Programming, and Answer Set Programming. These methods allow for very quick results on small

graphs. However, they can be very slow on large graphs.

Regarding sub-optimal solvers, the problem has been simplified by introducing a hierarchy of searches or by limiting it to specific types of paths. In this case, as well, we have some methods based on search. Each agent's path is determined individually, and then, during the confrontation of all paths, conflicts and deadlocks are resolved afterwards. These solvers are fast, easy to understand, and implement, but they cannot guarantee optimality and completeness. The main approach for this type of solver is *cooperative A**, [58]. This is called priority-based planning; each agent adheres to a priority order and plans for itself. The following agent must plan in a way that avoids conflicts with the previous agent.

The proposed methods for Search for Trajectory using Optimal Control in Section 1.3 can also be extended to multi-agent planning. In [37], the extension of the detection criterion to multiple agents for the MCM problem is already discussed. The same applies to the optimisation problem using Bernstein polynomials, [10]. Two constraints, in particular, can be added in this case. The first is called *spatial separation*, which requires that the global trajectories of the various members do not intersect at any point in time. The second, less stringent, is *temporal separation*, which ensures that multiple agents do not overlap at the same point in time.

2 | Trajectory Representation

Among the various options for describing trajectories presented in the previous chapter, this work focuses on the approach based on generating optimal trajectories using Bernstein polynomials. Specifically, the optimal motion planning problem is formulated as a continuous-time optimal control problem and then approximated in discrete time using Bernstein polynomials, [8]. This chapter addresses the representation of trajectories, starting from the mathematical definition of Bernstein polynomials and moving on to their properties and algorithms. The aim is to summarise the fundamental concepts used throughout the thesis. The contents are taken from the following articles: [15], [8], [9], [31], [33], and [14]. Therefore, anyone interested in studying certain aspects in more detail is advised to start with the references above.

2.1. Bernstein Polynomials

The general definition of a n -dimensional, N^{th} order Bernstein Polynomial (BP) is:

$$\mathbf{C}_N(t) = \sum_{i=0}^N \mathbf{P}_{i,N} B_{i,N}(t), \quad (2.1)$$

with $t \in [0, t_f]$. $\mathbf{P}_{i,N} \in \mathbb{R}^n$ are the control points, $i = 0, \dots, N$. The number of control points is $N + 1$. The control points are the parameters used to describe Bernstein polynomials and are represented with black dots in Figure 2.1 and Figure 2.2. For example, to describe a spatial trajectory in two dimensions, we have that $n = 2$. If, on the other hand, one wants to use a BP to describe the evolution of the norm of a velocity, $n = 1$.

$B_{i,N}(t)$ are the BP bases and are defined as:

$$B_{i,N}(t) = \binom{N}{i} \frac{t^i (t_f - t)^{N-i}}{t_f^N},$$

for all $i = 0, \dots, N$ where

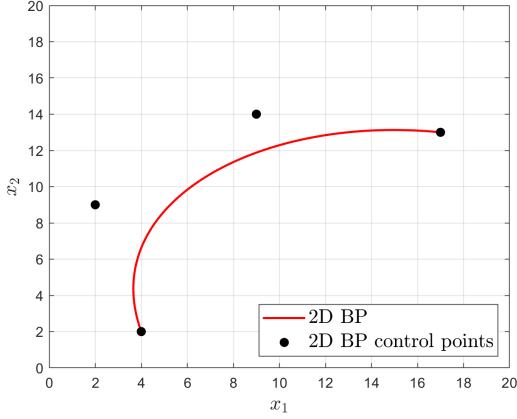


Figure 2.1: 2D BP example

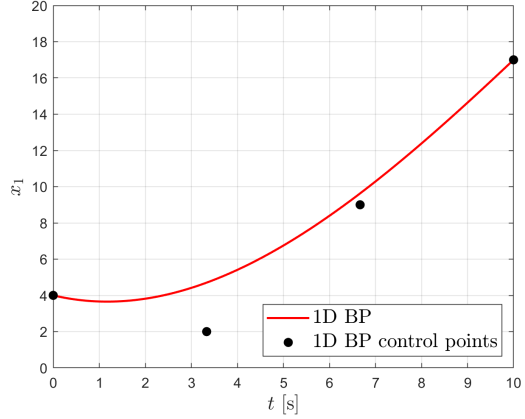


Figure 2.2: 1D BP example

$$\binom{N}{i} = \frac{N!}{i!(N-i)!}.$$

It is possible to define a 1-dimensional, N^{th} order *rational* BP:

$$R_N(t) = \frac{\sum_{i=0}^N P_{i,N} w_{i,N} B_{i,N}(t)}{\sum_{i=0}^N w_{i,N} B_{i,N}(t)}, \quad (2.2)$$

with $t \in [0, t_f]$ and where $w_{i,N} \in \mathbb{R}$, $i = 0, \dots, N$, are referred to as weights.

In Figure 2.1 it is reported one example of a 2-dimensional (x_1 and x_2) BP. In Figure 2.2 instead a 1-dimensional BP represents the trend of x_1 as a function of time.

BPs can be used to approximate functions. Consider a n -vector valued function $\mathbf{f} : [0, t_f] \rightarrow \mathbb{R}^n$. The N^{th} order Bernstein approximation of $\mathbf{f}(t)$ is $\mathbf{f}_N(t)$ and is computed as in equation 2.1 with $\mathbf{P}_{i,N} = \mathbf{f}(t_i)$, $t_i = i \frac{t_f}{N}$ for all $i = 0, \dots, N$:

$$\mathbf{f}_N(t) = \sum_{i=0}^N \mathbf{f}\left(i \frac{t_f}{N}\right) B_{i,N}(t). \quad (2.3)$$

One can prove the uniform convergence of the Bernstein approximation to the function it approximates, see *Lemma 1* of [8].

2.2. Properties

Let us now introduce some important properties.

Property 1: Convex Hull

A BP defined by equation 2.1 lies within the convex hull defined by its control points, see Figure 2.1 for an example of a 2-dimensional polynomial. Instead, for a 1-dimensional BP, it is equivalent to state that each point of the curve is included in the interval defined by the minimum and maximum value of the control points, see Figure 2.2. For example, let's consider the curve that describes velocity as a function of time. It is certain that at all time instances, the points on the trajectory will have a value between the velocity corresponding to the minimum and maximum values of the control points that describe the curve. This is the fundamental property that can be used to obtain dynamically feasible trajectories at every mission moment.

A rational BP represented by equation 2.2 is subjected to the same properties only if $w_{i,N} > 0$ for $i = 0, \dots, N$. In addition, in [14], a procedure is shown to obtain a tighter convex hull for rational BPs.

Property 2: End Point Values

The first and last control points of a BP and a rational BP are their end points:

$$\mathbf{C}_N(0) = \mathbf{P}_{0,N} \quad \text{and} \quad \mathbf{C}_N(t_f) = \mathbf{P}_{N,N}. \quad (2.4)$$

In addition, a BP is tangent to the convex polygon defined by its control points at the end points:

$$\dot{\mathbf{C}}_N(0) = \frac{N}{t_f}(\mathbf{P}_{1,N} - \mathbf{P}_{0,N}), \quad (2.5)$$

$$\dot{\mathbf{C}}_N(t_f) = \frac{N}{t_f}(\mathbf{P}_{N,N} - \mathbf{P}_{N-1,N}). \quad (2.6)$$

In a similar way also for rational BP:

$$\dot{\mathbf{C}}_N(0) = \frac{Nw_1}{t_f w_0}(\mathbf{P}_{1,N} - \mathbf{P}_{0,N}), \quad (2.7)$$

$$\dot{\mathbf{C}}_N(t_f) = \frac{Nw_{N-1}}{t_f w_N}(\mathbf{P}_{N,N} - \mathbf{P}_{N-1,N}). \quad (2.8)$$

Property 3: Derivatives

The derivative of a BP of order N is an $(N - 1)^{\text{th}}$ order BP defined as:

$$\dot{\mathbf{C}}_{N-1}(t) = \sum_{i=0}^{N-1} \mathbf{P}'_{i,N-1} B_{i,N-1}(t). \quad (2.9)$$

The vector of control points \mathbf{P}'_{N-1} is given by:

$$\mathbf{P}'_{N-1} = \mathbf{P}_N \mathbf{D}_N, \quad (2.10)$$

where

$$\mathbf{D}_N = \frac{N}{t_f} \begin{bmatrix} -1 & 0 & \cdots & 0 \\ 1 & \ddots & \ddots & \vdots \\ 0 & \ddots & \ddots & 0 \\ \vdots & \ddots & \ddots & -1 \\ 0 & \cdots & 0 & 1 \end{bmatrix}. \quad (2.11)$$

Property 4: Degree elevation and reduction

Any BP of degree N can be expressed as a BP of degree M , with $M > N$. The control points vector of the degree elevated BP, $\mathbf{P}_M = [\mathbf{P}_{0,M}, \dots, \mathbf{P}_{M,M}]$, is calculated as:

$$\mathbf{P}_M = \mathbf{P}_N \mathbf{E}, \quad (2.12)$$

where $\mathbf{E} = \{e_{i,k}\} \in \mathbb{R}^{(N+1) \times (M+1)}$ is the degree elevation (DE) matrix with elements given by

$$e_{i,j+1} = \frac{\binom{M-N}{j} \binom{N}{i}}{\binom{M}{i+j}}, \quad (2.13)$$

where $i = 0, \dots, N$ and $j = 0, \dots, M - N$. All the other values in the matrix \mathbf{E} are zeros. Performing the degree elevation of the control points defining a Bernstein polynomial results in a greater number of control points that move closer to the curve they generate, see Figure 2.3. As N tends to infinity, the control points asymptotically belong to the BP.

It is possible to do the opposite operation, called degree reduction. The goal is to express a polynomial of degree N as a combination of polynomials of degree M , $M < N$, while

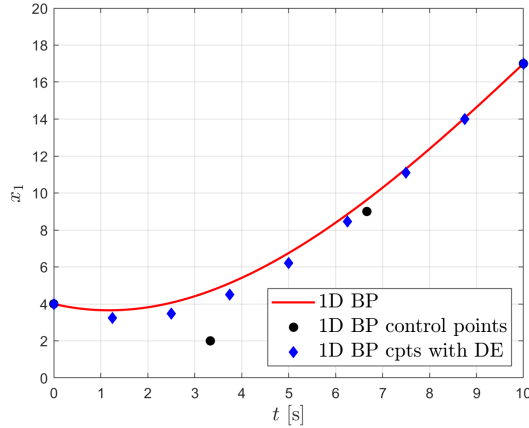


Figure 2.3: Degree Elevation of 1D BP of Figure 2.2

minimising the error introduced by the reduction. Refer to [40] for more details. Alternatively, an efficient algorithm is implemented in the BeBOT toolkit introduced in Section 1.3, [31].

Property 5: Arithmetic Operations

The sum (or difference) of two BP of the same order can be obtained by adding (or subtracting) their control points.

For what concerns the product consider two 1-dimensional BPs of degree M and N , $F_M(t)$ and $G_N(t)$, with control points $[X_{0,M}, \dots, X_{M,M}]$ and $[Y_{0,N}, \dots, Y_{N,N}]$. The product $C_{M+N}(t) = F_M(t)G_N(t)$ is a BP of degree $M + N$ with control points $P_{k,M+N}$, $k \in \{0, \dots, M + N\}$ given by

$$P_{k,M+N} = \sum_{j=\max(0,k-N)}^{\min(M,k)} \frac{\binom{M}{j} \binom{N}{k-j}}{\binom{M+N}{k}} X_{j,M} Y_{k-j,N}. \quad (2.14)$$

The ratio instead has to be done between two 1-dimensional BPs of the same order, $F_N(t)$ and $G_N(t)$, with control points $[X_{0,N}, \dots, X_{N,N}]$ and $[Y_{0,N}, \dots, Y_{N,N}]$. The ratio $R_N(t) = F_N(t)/G_N(t)$ can be expressed as a rational BP as defined in equation 2.2, with the following control points and weights

$$P_{i,N} = \frac{X_{i,N}}{Y_{i,N}}, \quad w_{i,N} = Y_{i,N}. \quad (2.15)$$

2.3. Algorithms

In this section, the main concepts behind some algorithms that use the properties of Bernstein polynomials to obtain additional information are presented and explained.

A. The de Casteljau Algorithm

The de Casteljau algorithm, [15], is an efficient recursive method to evaluate a BP at any given point. It is also used to split a BP into two independent ones. Given a BP defined by the equation 2.1 with control points $\mathbf{P}_N = [\mathbf{P}_{0,N}, \dots, \mathbf{P}_{N,N}]$ and a scalar $t_{div} \in [0, t_f]$, the BP at t_{div} is computed using the following recursive relationship:

$$\mathbf{P}_{i,N}^{[0]} = \mathbf{P}_{i,N}, \quad i = 0, \dots, N,$$

$$\mathbf{P}_{i,N}^{[j]} = \mathbf{P}_{i,N}^{[j-1]} \left(\frac{t_f - t_{div}}{t_f} \right) + \mathbf{P}_{i+1,N}^{[j-1]} \frac{t_{div}}{t_f}, \quad (2.16)$$

with $i = 0, \dots, N - j$, and $j = 1, \dots, N$. The BP evaluated at t_{div} is given by $C_N(t_{div}) = \mathbf{P}_{0,N}^{[N]}$. The BP can be subdivided at t_{div} into two N^{th} order BPs with control points

$$\mathbf{P}_{0,N}^{[0]}, \mathbf{P}_{0,N}^{[1]}, \dots, \mathbf{P}_{0,N}^{[N]} \quad \text{and} \quad \mathbf{P}_{0,N}^{[N]}, \mathbf{P}_{1,N}^{[N-1]}, \dots, \mathbf{P}_{N,N}^{[0]}.$$

In Figure 2.4 an example of the application of *de Casteljau* algorithm is reported for a 2-dimensional, 3^{rd} order BP. The control points that generated the red curve are the ones in black. Each set of control points in a different colour comes from a subsequent iteration of the recursive formula 2.16. The process continues until the desired point at t_{div} is obtained, $\mathbf{P}_{0,3}^{[3]}$.

B. Evaluating Bounds

By utilising *Property 1*, it is possible to determine conservative bounds for the BP. This is particularly easy and intuitive for polynomials of degree 1. The lower and upper bounds of the BP are given by the minimum and maximum values of the control points. To obtain tighter bounds, it is possible to use the property of degree elevation.

C. Minimum Spatial Distance

The minimum spatial distance between two BPs or between a BP and a polygon is another important characteristic that can be retrieved thanks to the above-mentioned properties, [31]. In particular, the *Convex Hull* and the *End Point Values* properties. In addition,

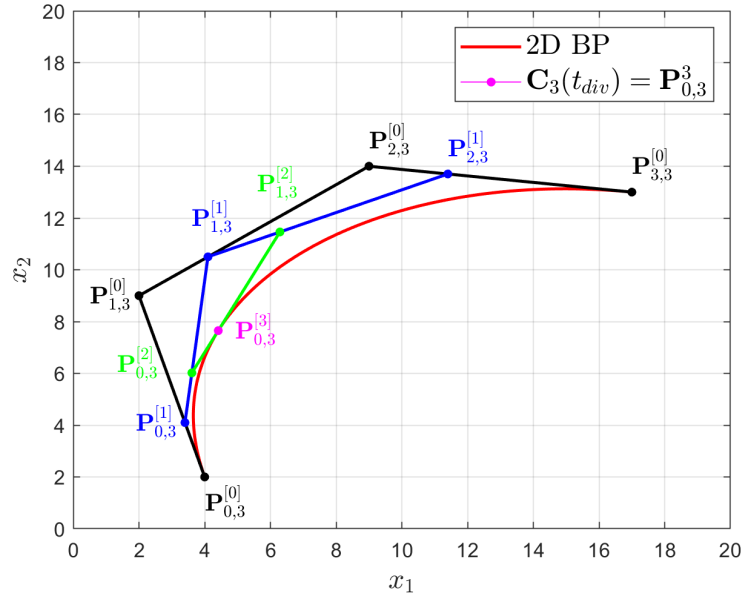


Figure 2.4: Example of *de Casteljau* algorithm for a 2-dimensional, 3^{rd} order BP

two algorithms are used: *de Casteljau* and Gilbert-Johnson-Keerthi (GJK), [20]. GJK is an algorithm used to compute the minimum distance between convex shapes.

The algorithm to evaluate the minimum spatial distance between two BPs has as inputs:

- Vector containing the control points of the first BP.
- Vector containing the control points of the second BP.
- Tolerance value.

Two different functions are used to find respectively an upper and lower bound for the distance. As the upper bound the maximum distance between the end points of the two BPs is considered. Instead, for the lower bound the GJK algorithm is used to find the minimum distance between the convex hull of the control points of the two BPs. The curve is then split and the same procedure is repeated for each pair until the difference between the upper and lower bounds satisfies a given tolerance. A similar algorithm can be used to calculate the minimum distance between a BP and a polygon.

D. Collision Detection

For trajectory planning, it is crucial to assess the potential collisions between the drone's trajectory and obstacles or other drones in the fleet.

The collision detection algorithm operates on a principle similar to that of minimum spatial distance. However, rather than calculating the minimum distance using GJK, it

produces a binary result indicating whether or not there is an intersection between the two convex hulls. As mentioned earlier, the two convex hulls can either be the convex hulls of the drone's trajectory and the obstacle, or the convex hulls of the trajectories of two drones. Two similar but distinct algorithms are implemented for these two cases. In the first case, the inputs are the control points of a trajectory and the vertices of the obstacle; in the second case, the inputs are the two sets of control points for the two trajectories. In addition, a maximum iteration value is fixed. An additional function is employed to process these inputs: it takes the two initial sets and returns two subsets, each representing convex hulls from one set that has been detected to intersect with at least one convex hull from the other set. If no intersections are detected, the algorithm indicates that no collision is found. For curves where collisions are detected, they are split at $t = t_f/2$. This is done to refine the convex hull further and determine whether a collision has occurred. The original curve is removed from the set, and the newly created curves from the split are added to the set. If the algorithm reaches the maximum number of iterations, it concludes that a collision is possible.

This algorithm was initially used in the early stages of the work when the obstacles were modelled as polygons but was set aside later on.

2.4. Advantages of Trajectory Representation using BP

Using these polynomials to represent trajectories in solving an optimal trajectory search problem offers three key advantages. Firstly, the resulting trajectories are smooth, making them more easily usable as reference paths for the guidance phase. In contrast, other methods provide only a sequence of waypoints, which are not easily exploitable. Secondly, related to this, certain properties of these polynomials can be leveraged to ensure that the curve meets specific characteristics throughout its entire length, not just at the control points. This ensures that the resulting trajectories are not only smooth but also feasible from, for example, a dynamic perspective. Finally, the third strength lies in the fact that continuous trajectories can be described using a limited number of variables, the control points.

3 | Problem Description

In this chapter, the problem setting is described in detail. Starting from the assumptions about the physics of the problem and the resulting choices regarding the dynamic model, we then move on to the description and translation in terms of Bernstein polynomials. The first element that is analysed is the UAV, its dynamics and the modelling of the onboard sensor. Secondly, different types of targets are characterised. In particular, the modelling of prior knowledge is presented, which refers to the information the drone possesses before starting the mission. Subsequently, the environment is introduced, which involves describing and modelling the zone and the obstacles.

3.1. UAV

The drone under consideration has four propellers, it is called a quadcopter. Specifically, there are two reference drones. See images 3.1 and 3.2 and technical specifications in Chapter 5.



Figure 3.1: DJI Matrice 100 from [62]



Figure 3.2: DJI Matrice 300 RTX from [63]

3.1.1. Dynamics

The first assumption is that the drones in the mission move at a constant altitude relative to the ground, which is assumed to be flat. The dynamics are therefore simplified, repre-

senting a planar movement with coordinates x_1 and x_2 . In addition, no dynamic relation is considered between the pitch angle and the horizontal velocity of the drone. A third assumption is that the sensor used to detect the targets is rigidly mounted and oriented in the UAV's motion direction, see Figures 3.3a and 3.3b. The equations governing the motion of the drones are:

$$\begin{cases} \dot{x}_1^u(t) = V(t) \cos \psi(t) \\ \dot{x}_2^u(t) = V(t) \sin \psi(t) \\ \dot{\psi}(t) = \omega(t) \end{cases}, \quad (3.1)$$

where ψ represents the heading angle, positive counter-clockwise from the axis defining x_1 to the direction of the velocity, tangent to the trajectory, see Figure 3.3b. The control input is the vector composed of the velocity magnitude V and the angular rate ω .

The UAV is also subjected to input constraints:

$$V_{min} \leq V(t) \leq V_{max}, \quad (3.2)$$

$$-\omega_{max} \leq \omega(t) \leq \omega_{max}, \quad (3.3)$$

with V_{min} , V_{max} , and $\omega_{max} > 0$. The system state vector is of dimension 5 and consists of x_1^u , x_2^u , ψ , V , and ω . It would be desirable to reduce the number of parameters involved to achieve greater computational efficiency for the optimisation performed later. Due to the different assumptions made previously, and to reduce the number of parameters considered, the system 3.1 is chosen. In fact, the system is differentially flat, [8]. A differentially flat system is a nonlinear control system for which a set of variables exists, known as flat outputs, such that the state and control variables of the system can be expressed as algebraic functions of these flat outputs and their derivatives, [59]. Given the flat outputs and their derivatives, one can reconstruct the original system states and control inputs. This is particularly useful for designing control strategies because the control design can be performed in the space of the flat outputs, which is often simpler than dealing with the original nonlinear system directly.

In the system under consideration, we assume that, except for ψ , all the states are measured and so the flat output is $\mathbf{x}^u(t) = [x_1^u(t), x_2^u(t)]^T$. To express the constraints on the velocity input solely in terms of the flat outputs, equation 3.2 is rewritten as:

$$V_{min}^2 \leq V^2(t) \leq V_{max}^2. \quad (3.4)$$

Therefore, the following expressions, which depend solely on $\mathbf{x}^u(t)$, are obtained:

$$V^2(t) = (\dot{x}_1^u(t))^2 + (\dot{x}_2^u(t))^2, \quad (3.5)$$

$$\omega(t) = \frac{\dot{x}_1^u(t)\ddot{x}_2^u(t) - \ddot{x}_1^u(t)\dot{x}_2^u(t)}{(\dot{x}_1^u(t))^2 + (\dot{x}_2^u(t))^2}. \quad (3.6)$$

Using the constraints 3.4 and 3.3 rewritten through 3.5 and 3.6, one is simultaneously imposing control inputs constraints and the system dynamics 3.1. For this reason, the two functions chosen for the optimisation are $x_1^u(t)$ and $x_2^u(t)$.

In addition, boundary conditions for the trajectory are taken into consideration. In particular, the initial position, direction and final position are imposed:

$$\mathbf{x}^u(0) = \mathbf{x}_0^u, \quad \psi(0) = \Psi_0, \quad \text{and} \quad \mathbf{x}^u(t_f) = \mathbf{x}_f^u. \quad (3.7)$$

BP modelling

We consider the Bernstein Polynomial of order N that approximates the flat output: $\mathbf{x}_N^u(\mathbf{P}_{i,N}^u, t) = [x_{1_N}^u(\mathbf{P}_{i,N}^u, t), x_{2_N}^u(\mathbf{P}_{i,N}^u, t)]^\top = \sum_{i=0}^N \mathbf{P}_{i,N}^u B_{i,N}(t)$, where $\mathbf{P}_{i,N}^u \in \mathbb{R}^2$ are the control points of the trajectory. For notation clarity, the two control points components, one for each dimension, are split into two separate variables: $\mathbf{P}_{1_{i,N}}^u = [P_{1_{0,N}}^u, \dots, P_{1_{N,N}}^u]$ and $\mathbf{P}_{2_{i,N}}^u = [P_{2_{0,N}}^u, \dots, P_{2_{N,N}}^u]$. These control points are the parameters that convert the optimisation from functional to parametric. The control input inequality constraints can be expressed with the optimisation parameters thanks to the *Convex Hull* property, *Property 1*. The boundary conditions instead thanks to the *End Point Values* property, *Property 2*. The system obtained is:

$$V_{min}^2 \leq (\dot{\mathbf{P}}_{1_{i,N}}^u)^2 + (\dot{\mathbf{P}}_{2_{i,N}}^u)^2 \leq V_{max}^2, \quad (3.8)$$

$$-\omega_{max} \leq \frac{\dot{\mathbf{P}}_{1_{i,N}}^u \ddot{\mathbf{P}}_{2_{i,N}}^u - \ddot{\mathbf{P}}_{1_{i,N}}^u \dot{\mathbf{P}}_{2_{i,N}}^u}{(\dot{\mathbf{P}}_{1_{i,N}}^u)^2 + (\dot{\mathbf{P}}_{2_{i,N}}^u)^2} \leq \omega_{max}, \quad (3.9)$$

$$\mathbf{P}_{0,N}^u = \mathbf{x}_0^u, \quad \angle(\mathbf{P}_{1,N}^u - \mathbf{P}_{0,N}^u) = \Psi_0, \quad \mathbf{P}_{N,N}^u = \mathbf{x}_f^u. \quad (3.10)$$

To evaluate equations 3.8 and 3.9, the BP properties for derivatives, products, and arithmetic operations have been utilised. Consider the equation 3.8. The control points of the resulting BP used to impose the constraint on the squared velocity are given by:

$$P_{i,N}^u = (\dot{\mathbf{P}}_{1_{i,N}}^u)^2 + (\dot{\mathbf{P}}_{2_{i,N}}^u)^2 \quad (3.11)$$

For BPs defined by the equation 2.1, there are no conditions on the sign of the control points for the validity of the convex hull property. However, it is important to note that the resulting control points, $\mathbf{V}\mathbf{2}_{i,N}^u$, are obtained through the operations of differentiation, multiplication, and summation of the BPs of the flat outputs $\mathbf{x}_N^u(\mathbf{P}_{i,N}^u, t)$. In particular, it should be noted that the square appearing in equation 3.11 follows the rules of BP multiplication, see equation 2.14. As a result, the obtained control points may have a negative sign. However, regarding the behaviour of the squared velocity evaluated at each time instant, i.e., the Bernstein polynomial defined by these control points, it is positive by definition of the system 3.1. Therefore, the sign of the control points depends on the chosen degree N . In fact, as the degree N increases, the control points get closer and closer to the curve they describe. Since the curve is always positive there is a certain degree N beyond which all the control points become positive.

Now we consider the middle term of equation 3.9 that describes the turning rate convex hull. It is important to note that the convex hull property is being applied to a 1-dimensional rational BP to rewrite the constraint under this form. It is recalled that to apply the properties of BPs to rational BPs of dimension one, written in the form 2.2, all control weights must be positive, while there are no conditions regarding the sign of the control points. By making an analogy between equations 2.2, 2.15, and 3.6, we can obtain the weights $w_{i,N}$ and the control points $P_{i,N}^u$ defining the rational BP for the turning rate:

$$w_{i,N} = (\dot{P}_{1i,N}^u)^2 + (\dot{P}_{2i,N}^u)^2 \quad (3.12)$$

$$P_{i,N}^u = \frac{\dot{P}_{1i,N}^u \ddot{P}_{2i,N}^u - \ddot{P}_{1i,N}^u \dot{P}_{2i,N}^u}{(\dot{P}_{1i,N}^u)^2 + (\dot{P}_{2i,N}^u)^2} \quad (3.13)$$

It can be observed that the weights defining the rational BP for the turning rate, $w_{i,N}$, correspond to the control points of the BP describing the squared velocity. The same considerations made earlier for equation 3.11 can be applied to the weights of the turning rate. The goal is to ensure that all weights of the rational BP are positive. To achieve this, the degree of the BP for the squared velocity must be sufficiently high to make its control points positive, which will in turn ensure that the weights for the turning rate are also positive.

We can therefore conclude that the validity and accuracy of the constraints depend on the number of control points that define the squared velocity curve. For higher accuracy and validity, a larger number of control points is required; however, this comes at the cost of an increased number of parameters that need to be optimized. We can use the *Degree Elevation* property to address this issue. Instead of using a greater degree for the spatial

trajectory, x_1^u and x_2^u , what can be done is to use degree elevation only when verifying the constraints on squared velocity norm and the turning rate, equations 3.8 and 3.9. In this way, the number of optimisation variables remains the same but the convex hulls on which the constraints are verified are less conservative. In other words, the degree elevation allows the optimisation algorithm to discard fewer feasible solutions.

Alternative BP modelling

There is another possibility for choosing optimisation variables for this kind of problem. Instead of using the differentially flat system properties, all the states and inputs are modelled as BPs with their respective control points and are optimised in the process: x_1^u , x_2^u , ψ , V , and ω . In this case, the dynamics are enforced by using derivatives of x_1^u , x_2^u , and ψ and imposing system 3.1. The inequality constraints are imposed directly on variables V and ω . There are two main drawbacks to writing the optimisation problem in this form.

The first one is that the problem passes from $2(N+1)$ to $5(N+1)$ optimisation variables, increasing the computational resolution time.

The second and most important one is that the cosine and sine of a BP are not BPs. In this way, we are imposing the dynamics correctly only at the control points. Solely the first and last one, the end points, belong to the curve, so the dynamics will be correct only at these points. The first solution involves increasing the order of the polynomials. By doing so, the control points converge to the curve, making the approximation sufficiently accurate for high orders of polynomials. A more accurate approach is to use *composite* Bernstein polynomials, [21], which are polynomials that are interconnected at the end points, where the dynamics are correctly enforced. Both propositions to overcome the approximation problem propose an augmentation of the number of optimisation variables. For this reason, this second way of solving the problem is not considered for the next part of this work. Nevertheless, it cannot be ruled out that, for other dynamic systems, it might be a valid choice.

3.1.2. Sensor Modelling

To integrate the detection characteristics of the sensor into the trajectory planning optimisation, it is necessary to model the sensor, specifically the function describing the instantaneous detection rate, which is defined as the probability of identifying a target if it is inside the drone's field of view (*FOV*). This probability is influenced by the position of the UAV and the target, and by the probability of successfully identifying whether it

is a target or a decoy.

In a general form, the instantaneous detection rate can be expressed as $r(\mathbf{x}^u(t), \mathbf{x}^t(t), t)$, where $\mathbf{x}^t(t)$ is the target position. In [65] the function used to model the sonar is the Poisson Scan Model. It takes into consideration some equipment-specific constants. The concept is reprised and further developed in [37]. For the authors, the instantaneous detection rate depends on several factors: the probability that a given sonar can detect an echo from a specific target at the expected ambient noise level, the two-way propagation losses, functions modelling the 3D geometry FOV , and the turning rate. If the sonar systems are rigidly mounted onto the vehicle, the FOV depends on the trajectory in terms of position, orientation, and velocity. Since these quantities are derived from the equations of motion, the quality of detection is based not only on the characteristics of the sensor but also on the dynamics of the vehicle.

Also in our case, the sensor is assumed fixed on the drone's structure. Therefore, the detection depends on the trajectory, in terms of position and direction. However, the model in this thesis is simplified and does not account for velocity and turning rate. In other words, the sensor characteristics and the identification algorithm do not account for velocity and turning rate dependent parameters.

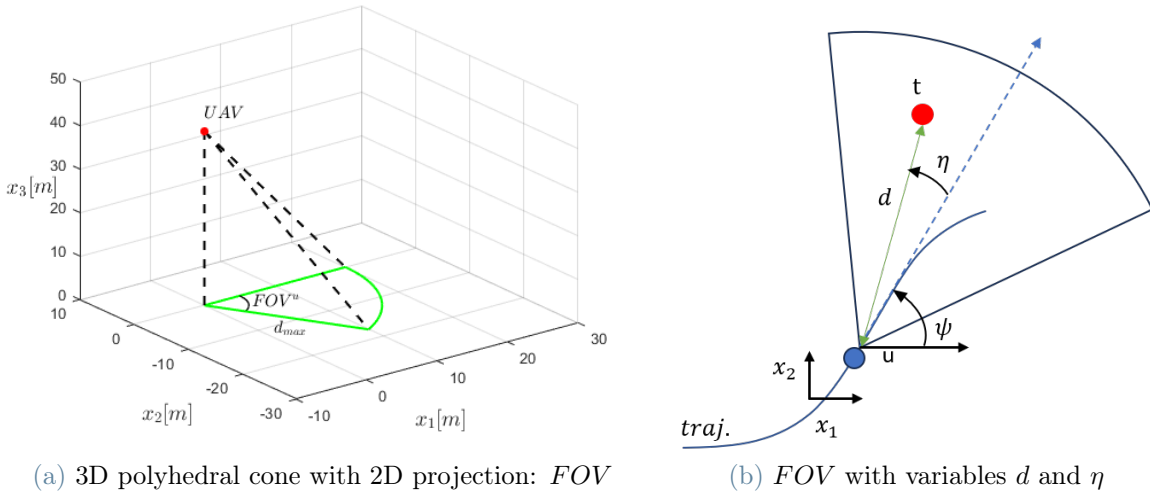


Figure 3.3: FOV

In general, for UAVs, the term FOV refers to the polyhedral cone with the UAV itself as the origin. From now on in this work, we consider the FOV as the projection of the polyhedral cone onto the ground, which is considered flat, see Figure 3.3a. Two quantities are considered to evaluate the instantaneous detection rate function. The first one is the distance between the UAV and the target, d . The second one is the relative angle between

the UAV's trajectory direction and the direction connecting the UAV to the target, η . We suppose FOV as bounded for both variables as shown in Figures 3.3a and 3.3b. When referring to the angle amplitude of the FOV we note FOV^u . Instead, the maximum distance within which it is possible to observe the environment is d_{max} . Additionally, it is assumed that the function takes a value of zero at the boundary points of the FOV , while at the centre, it takes a value of one.

Below, a recap is provided of all the quantities and parameters related to the modelling of the sensor.

- d : the distance between the UAV and the target.
- d_{max} : the maximum projected distance within which it is possible to observe the environment.
- d_{min} : the minimum projected distance beyond which it is possible to observe the environment. In Figure 3.3 and in the results of Chapter ??, it is considered equal to zero.
- η : the relative angle between the UAV's trajectory direction and the direction connecting the UAV to the target.
- FOV : the projection of the polyhedral cone onto the ground.
- FOV^u : the angle amplitude of the FOV .

The function being studied and designed is a two-dimensional Beta distribution, see equations 3.14, 3.15, and 3.16. This function can be modelled to assume zero values at the boundaries and normalised to obtain a maximum value of one in the middle, as illustrated by Figure 3.4. This last characteristic of symmetry is achievable by imposing that $\beta = \alpha$. See equation A.1 in Appendix A for the general definition of the probability density function of a Beta distribution.

A target can be considered inside the FOV area if:

$$\begin{aligned} d_{min} &\leq \|\mathbf{x}^u - \mathbf{x}^t\|_2 \leq d_{max} \\ -FOV^u/2 &\leq \angle(\mathbf{x}^u) - \angle(\mathbf{x}^t) \leq FOV^u/2 \end{aligned}$$

For the general FOV domain $[-FOV^u/2, FOV^u/2] \times [d_{min}, d_{max}]$ we have the following equations, one for each variable:

$$r_1(\eta) = (\eta - (-FOV^u/2))^{\alpha-1} \cdot (FOV^u - (\eta - (-FOV^u/2)))^{\alpha-1}, \quad (3.14)$$

$$r_2(d) = (d - d_{min})^{\alpha-1} \cdot ((d_{max} - d_{min}) - (d - d_{min}))^{\alpha-1}. \quad (3.15)$$

The final instantaneous detection rate function is obtained as:

$$r(\eta, d) = \frac{r_1(\eta) \cdot r_2(d)}{r_1(0) \cdot r_2((d_{max} - d_{min})/2)}. \quad (3.16)$$

Finally, it is necessary to specify the relationship between the positions of the UAV and the target and the variables η and d required to evaluate $r(\eta, d)$. In this way, also the instantaneous detection rate function is written in function of the BP control points. Given the position of the target $\mathbf{x}^t(t)$ and the trajectory of the drone $\mathbf{x}_N^u(\mathbf{P}_{i,N}^u, t)$, it is possible to rewrite: $\eta(\mathbf{x}_N^u(\mathbf{P}_{i,N}^u, t), \mathbf{x}^t(t)) = \eta(\mathbf{P}_{i,N}^u, \mathbf{x}^t, t)$ and $d(\mathbf{x}_N^u(\mathbf{P}_{i,N}^u, t), \mathbf{x}^t(t)) = d(\mathbf{P}_{i,N}^u, \mathbf{x}^t, t)$. The instantaneous detection rate function is then rewritten as

$$r(\eta, d) = r(\eta(\mathbf{P}_{i,N}^u, \mathbf{x}^t, t), d(\mathbf{P}_{i,N}^u, \mathbf{x}^t, t)) = r(\mathbf{P}_{i,N}^u, \mathbf{x}^t, t) \quad (3.17)$$

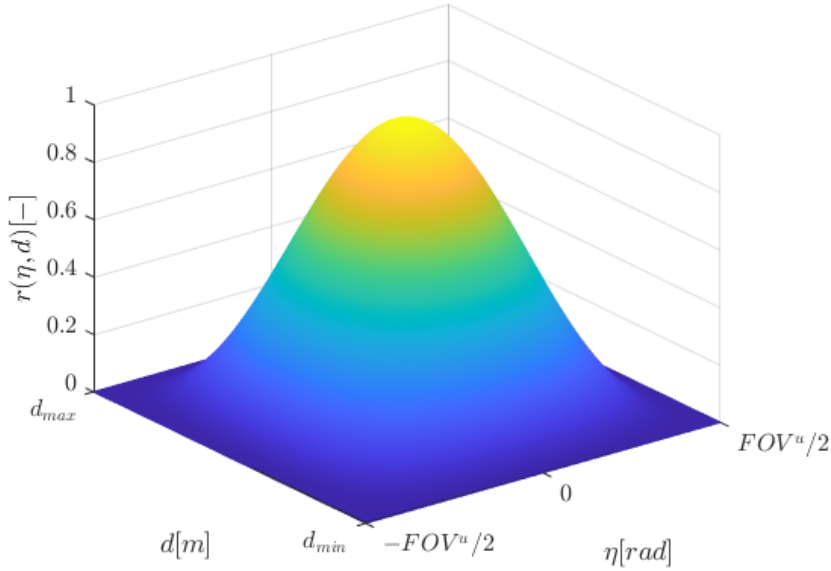


Figure 3.4: Instantaneous detection rate function $r(\eta, d)$

3.2. Targets

The targets are simplified and represented as points that lie in a two-dimensional space. [46] divides search theory into two categories based on how the target's dynamics are modelled.

In the first category, the motion is described by a Markov process. A Markov process refers to a stochastic process where the future behaviour depends only on the current state and is independent of its past states. In other words, the Markov property implies that the future movements or transitions of the target are determined solely by its present position or state, without any memory of how it reached that state. [22] considers the motion of the target in a given region as a diffusion process.

The second category consists of targets whose dynamics are conditionally deterministic. This means that the motion depends on a stochastic parameter. If this parameter is known, the target's location is known at any time. The target can be dynamic as in [65], moving from right to left with constant velocity. The stochastic parameters are the initial position coordinates. Otherwise, if the target is assumed to be stationary then it can be described by a probability distribution function as in [37]. The uncertain parameter is the location of the target.

What specifically interests us is modelling the information the UAV has about the targets and the area of interest, which the optimisation algorithm uses to design the trajectory. Instead, for what concerns the numerical simulations of the target refer to Section 4.1.2.

Within this study, three different scenarios are considered and analysed below. For each scenario, after describing its characteristics, the modelling of the probability map for the entire mission area is presented. As described in Section 1.4 and in 3.3, the probability map is obtained by dividing the zone into a discretised grid. For each cell, a value representing the probability of a target's presence is assigned.

3.2.1. Known Fixed Target

In Scenario 1, the target is known with certainty and is located at a precise position. In this case, it can be said that there is no stochastic parameter, or rather, the parameter is unique and always defined. The situation is not very interesting as it is unrealistic. Nevertheless, we consider it to see if, even in an ideal case, optimisation can leverage the prior information to design a trajectory that attempts to approach and observe the target. The probability map is set to zero everywhere except for the target position point where a value of one is assigned, as illustrated by Figure 3.5

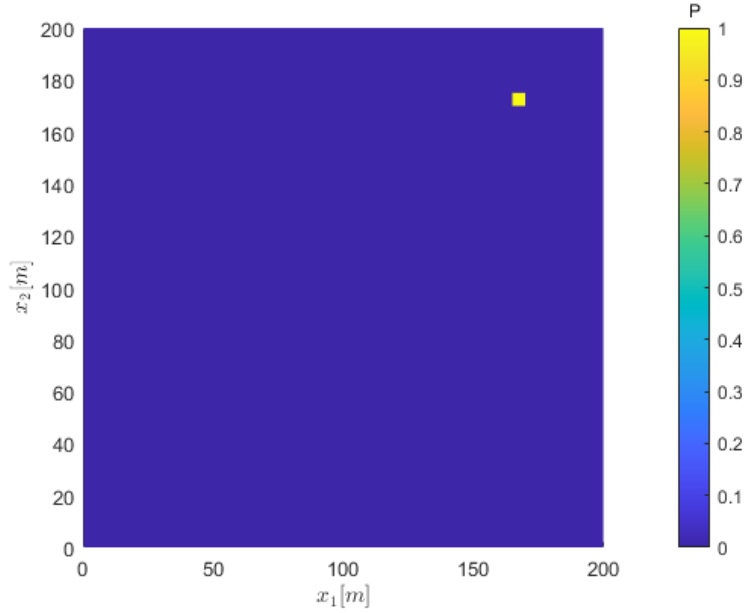


Figure 3.5: Probability map for a known fixed target

3.2.2. Partially Known Fixed Target

In Scenario 2, consider that from a previous mission or other types of measures, there is information indicating that a target is located in a specific area. Therefore, the goal is to refine the existing knowledge to better delineate the subsequent search efforts. In this case, as in [37] the target is static and the presence is modelled with a probability density function. In particular, a joint normalised Beta distribution is chosen to model the prior information. This family of distributions is easy to manipulate and design with the parameters α and β . In addition, they have a finite radius of effectiveness. In other words, the support is bounded, unlike, for example, the normal distribution. The uncertain parameter is the location of the target \mathbf{x}^t . Therefore, a random variable \mathbf{X}^t is introduced, $\mathbf{X}^t \in \Omega \subset \mathbb{R}^2$, where Ω is a subset of the zone of interest. The probability density function is noted as $f_{\mathbf{X}^t}(\mathbf{x}^t) : \Omega \mapsto \mathbb{R}$, and is given by the product of two Beta distributions, one for each dimension of the search area:

$$f_{\mathbf{X}^t}(\mathbf{x}^t) = f_{X_1^t}(x_1^t) \cdot f_{X_2^t}(x_2^t) \quad (3.18)$$

$$f_{X_i^t}(x_i^t) = (x_i^t - x_{i,0})^{\alpha_i - 1} (D_i - (x_i^t - x_{i,0}))^{\beta_i - 1}. \quad (3.19)$$

where $i = 1, 2$ and indicates the spatial dimension, D_i defines the domain Ω , and $x_{i,0}$ the origin of Ω . The origin is retrieved from $x_{i,c}$, the position with the maximum probability,

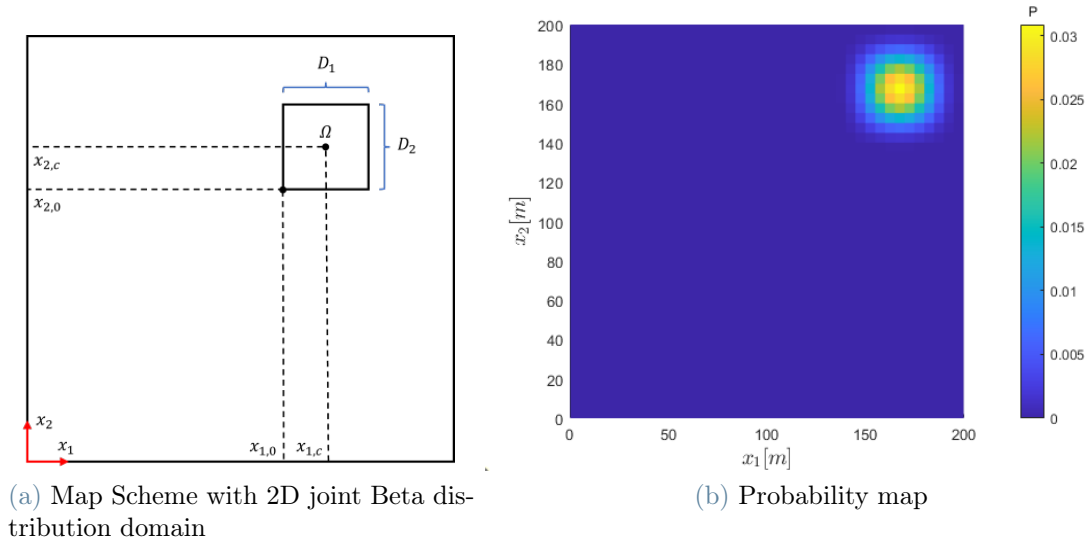


Figure 3.6: Partially Known Fixed target

$x_{i,0} = x_{i,c} - D_i/2$. This position is fixed at the centre of Ω since we are considering a symmetric distribution, $\beta = \alpha$.

To obtain the joint probability density function 3.18 it is necessary to normalise 3.19 to obtain a volume equal to 1. Since the area is subsequently discretised, the transition is made from a probability density function (PDF) to a probability mass function (PMF). The normalisation is now made by dividing equation 3.18 by the sum of the values of the discrete points. The probability map in this scenario is made by zeros everywhere except for the zone Ω where the probability mass function is applied. Note that the probability scale in Figure 3.6b is different from the previous case where the maximum was set to 1. Here, the maximum, represented by the colour yellow, is set to the highest value it takes across the entire area, which is 3.0899×10^{-2} .

Another way of describing this scenario is by using a non-symmetric joint Beta distribution over the entire zone of interest. See Appendix A for more details.

3.2.3. Unknown Moving Target

In this section, Scenario 3, the general case, is considered. A target is present in the area with no prior information available, and it can move in an almost entirely unknown manner. The only available information is about its maximum achievable speed. The probability map is composed of cells representing the probability that target i is located in cell j , $P_{i,j}$. With the assumption that the target exists and cannot exit from the zone of interest, it follows that $\sum_j P_{i,j} = 1$. Since no information is given before the mission

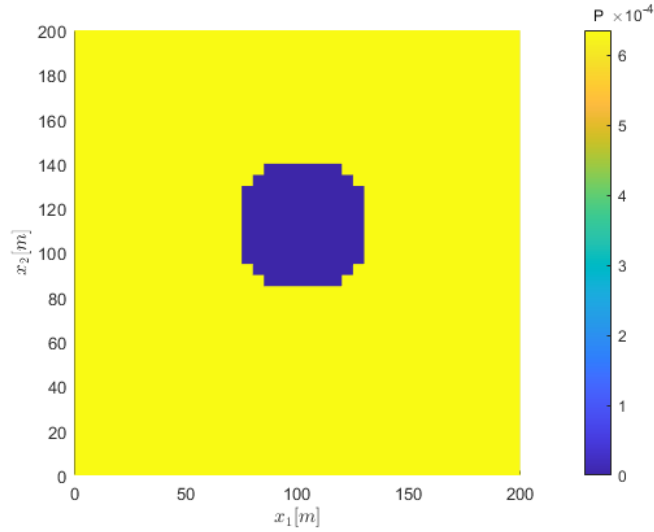


Figure 3.7: Probability map for unknown target

the probability map is composed of cells with $P_{i,j} = 1/N_{free}$, where N_{free} is the number of cells where the target could be located. The target can move everywhere except where an obstacle is defined, where $P_{i,j} = 0$. This is the map given to the optimiser to design the trajectory, see Figure 3.7. Note that also in this case the probability scale is different. The maximum value is $1/N_{free}$, in this case 6.3613×10^{-4} . What concerns the motion estimation is more detailed in Section 4.1. Figure 4.2 presents the matrix used to predict the target motion information under certain assumptions.

3.3. Environment and Obstacles

The area considered is a square portion of territory, $(x_{max} - x_{min}) \times (x_{max} - x_{min}) m^2$. Although it is assumed that the targets are on flat ground, thus at a different altitude than the drones, they are projected onto the same plane with drones and obstacles. The zone of interest is discretised with a uniform grid of nodes. The number of discretised points for each dimension is K_s . Therefore, the total points that describe the area are $M = K_s^2$. The points form a grid of cells. Each cell is assigned the value of the point at its bottom-left corner. The dimension of the cells, $\delta x \times \delta x$, depends on the spatial discretisation, $\delta x = (x_{max} - x_{min}) / (K_s - 1)$.

Obstacles are considered areas where neither the drone nor the target can move and which also obstruct detection, in terms of field of view. Additionally, due to potential inaccuracies in knowing both the searcher's position and the obstacles' positions, it is desirable to maintain a minimum safety distance, E , from each obstacle. The obstacles

are geometrically represented by circles and polygons. The focus is on integrating obstacle avoidance within the optimisation. This can be achieved by including it either in the cost function or as a constraint in the problem. The description of some possibilities for the first option is provided in Appendix B for scenarios involving polygonal obstacles. However, throughout the thesis, obstacle avoidance through constraint formulation is considered.

The constraints have to be written always in function of BP control points. Circular obstacles are defined by the centre \mathbf{c}_{O_i} and the radius r_{O_i} , where $i = 1, \dots, N_{obs}$. Polygonal obstacles are defined through their vertices coordinates ordered counter-clockwise. What is enforced by the constraint is that the minimum distance between the trajectory defined by the BP and the obstacle must be greater than the safety distance. This is achieved by utilising the properties of BP and particularly the algorithm used to compute the minimum distance between a BP and a polygon, as detailed in Section 2.3. When dealing with a circle, the computed distance is between the trajectory and the circle's centre, adjusted by subtracting the radius length. In the case of polygonal obstacles, the algorithm directly calculates the minimum distance between the trajectory and the obstacle. The function is implemented in MATLAB in the toolkit BeBOT, coded by V. Cichella and C. Kielas-Jensen, and is called `MinDistBernstein2Polygon.m`. The two inputs are the control points defining the BP and an array containing the vertices defining a polygon (\mathbf{obs}_{pol}), or simply a point for the case of the circle centre. Here in the text, it is named *minDist* and the constraints respectively for a circle and a polygon are:

$$\mathit{minDist}(\mathbf{x}_N^u(\mathbf{P}_{i,N}^u, t), \mathbf{c}_{O_i}) + r_{O_i} \geq E \quad (3.20)$$

$$\mathit{minDist}(\mathbf{x}_N^u(\mathbf{P}_{i,N}^u, t), \mathbf{obs}_{pol}) \geq E \quad (3.21)$$

The *minDist* function does not create issues when used to calculate the minimum distance between trajectory 1 and 2 and a point, specifically the centre of the circle, Figure 3.8a. In this case, the *minDist* function is always differentiable. However, when calculating the minimum distance between a trajectory defined by a BP and a polygon, a result is obtained only if the trajectory does not intersect the obstacle, Figure 3.8b. If there is an intersection, the function cannot handle it. For this reason, the *minDist* function is not differentiable when considering trajectories that intersect a polygon. This causes problems if the trajectory initial guess for the optimisation, see Section 4.2.2, intersects the obstacles present in the environment.

From this point onward, the representation used in the thesis will involve circular obstacles. To correctly utilise polygons, a solution is here proposed. It consists of performing an initial optimisation with circular obstacles that encompass the polygonal obstacles to

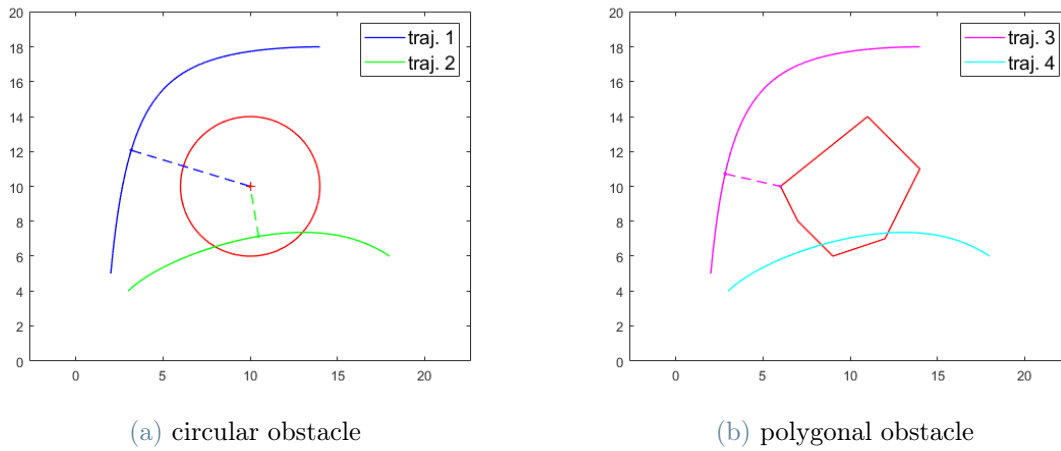


Figure 3.8: Minimum distance between a BP and obstacles

obtain a feasible trajectory. Then, this trajectory can be used as an initial guess for a new optimisation, this time with polygonal obstacles and imposing a certain safety distance.

4 | Estimation and Trajectory Optimisation Methods

In the third scenario of the previous chapter, an unknown target moves within the area of interest. In such a case, the estimation of the target location must account for its potential evolution. The estimation scheme thus integrates a prediction step which is completed by the correction step following the collection of measurements. The first part of the following chapter is dedicated to this aspect and also to the measurement updates. The second part, on the other hand, focuses on the description of the optimisation problem that generates the trajectory, specifically on the definition of the objective function and the selection of the initial guess.

4.1. Estimation of Target Location

It is important to note that the measurements we consider consist directly of the position of the targets. The measurements are also taken at a certain frequency. This aspect is not thoroughly analysed in this work. They are assumed to be obtained via processing of the raw measurements provided by the sensors, images or any other signal quantities with an ad hoc algorithm extracting the main features, i.e. positions of the target in a frame common to the UAVs. To account for the uncertainty in this process, a probability is assigned to each position, representing the chance that a target is present in that specific position. Specifically, this is done as previously explained through the instantaneous detection rate function, Section 3.1.2. The probability map represents the set of probabilities for all possible positions within the area of interest. Therefore, the estimation is performed not only to calculate the target's location but also to update the probability map.

The process of updating information during the mission is here explained. Specifically, the alternation between measurements (correction step) and propagation (prediction step) is detailed.

Both target locations and the probability map are stochastic with an a priori distribution.

Therefore, the Bayesian estimation framework is selected as the most relevant to our context of application. First of all, it allows for the integration of prior knowledge or beliefs about a system through the prior distribution. This prior distribution, combined with new measurements, helps refine the estimates of system parameters. Then, Bayesian estimation naturally incorporates uncertainty from both the measurements and the model. By combining prior distributions with likelihood functions from new data, it updates the estimate and adjusts for uncertainty in a mathematically rigorous way. Finally, it supports recursive updates, which is particularly useful for real-time applications. As new measurements are obtained, the posterior distribution from previous updates becomes the prior for the next update.

Propagation, as detailed later, is performed using a state transition matrix.

4.1.1. Correction Step

So far, the initial probability map, $P_{i,j}$, has been considered. The time variable is now introduced. A subscript k is added to the probability map, $P_{i,j,k}$. It is recalled that i denotes the target, j denotes the cell, and k denotes the time instant. The position of the cell j is a deterministic value noted x_j . There is no subscript i because it represents a position in the zone of interest, and no subscript k because the position of a cell is always the same. On the other hand, the position of each target is represented by random variable $X_{i,k}$. In this case, the subscript i is necessary to distinguish the random variable of each target and also k to define the time instant.

Regarding the measurements, the random variable $Y_{i,j,k}$ and its realisation $y_{i,j,k}$ are used to represent the measurements as a random process. This random variable is described by a Bernoulli distribution. That is, the result is 0 (no detection) or 1 (detection). Note that detection does not mean that the object that has been detected is a real target. This aspect will be handled by the definition of false detection. The probability that target i is in cell j at time k , given the measurement $y_{i,j,k}$, is:

$$\begin{aligned}
 P(X_{i,k} = x_j \mid Y_{i,j,k} = y_{i,j,k}) &= \frac{P(Y_{i,j,k} = y_{i,j,k} \mid X_{i,k} = x_j)P(X_{i,k} = x_j)}{P(Y_{i,j,k} = y_{i,j,k})} & (4.1) \\
 &= \frac{P(Y_{i,j,k} = y_{i,j,k} \mid X_{i,k} = x_j)P(X_{i,k} = x_j)}{P(Y_{i,j,k} = y_{i,j,k}, X_{i,k} = x_j) + P(Y_{i,j,k} = y_{i,j,k}, X_{i,k} \neq x_j)} \\
 &= \frac{P(Y_{i,j,k} = y_{i,j,k} \mid X_{i,k} = x_j)P(X_{i,k} = x_j)}{P(Y_{i,j,k} = y_{i,j,k} \mid X_{i,k} = x_j)P(X_{i,k} = x_j) + P(Y_{i,j,k} = y_{i,j,k} \mid X_{i,k} \neq x_j)P(X_{i,k} \neq x_j)}
 \end{aligned}$$

For what concerns the likelihood, 4 cases are possible:

$$P(Y_{i,j,k} = 1 \mid X_{i,k} = x_j) = p_D, \quad (4.2)$$

$$P(Y_{i,j,k} = 1 \mid X_{i,k} \neq x_j) = p_F, \quad (4.3)$$

$$P(Y_{i,j,k} = 0 \mid X_{i,k} = x_j) = p_N = 1 - p_D, \quad (4.4)$$

$$P(Y_{i,j,k} = 0 \mid X_{i,k} \neq x_j) = 1 - p_F. \quad (4.5)$$

p_D is the probability of detection, p_F is the probability of false detection, i.e. the probability of detecting a target on a cell which does not contain one, and p_N is the probability of no detection. Considering these definitions and the recursive updates, the equation 4.1 becomes:

$$P_{i,j,k} = \begin{cases} \frac{p_D P_{i,j,k-1}}{p_D P_{i,j,k-1} + p_F (1 - P_{i,j,k-1})} & \text{if } y_{i,j,k} = 1 \\ \frac{(1 - p_D) P_{i,j,k-1}}{(1 - p_D) P_{i,j,k-1} + (1 - p_F) (1 - P_{i,j,k-1})} & \text{if } y_{i,j,k} = 0 \\ P_{i,j,k-1} & \text{if } y_{i,j,k} = \emptyset \end{cases} \quad (4.6)$$

The probability of detection, p_D , is calculated thanks to the instantaneous detection rate function 3.17. This function is applied to each discrete point within the drone's field of view. The resulting values range from zero at the edges of the *FOV* to one at the centre.

p_D and p_F are strictly related. In general, when aiming for a high detection probability, this also increases the likelihood of more false detections (decoys). On the contrary, to reduce decoys and consequently the false alarm rate (p_F), the detection probability (p_D) will decrease. For this reason, a trade-off is necessary. p_F is chosen as a percentage of p_D , for example we consider $p_F = 0.1 \cdot p_D$. Two extreme cases are worth examining to assess the physical significance of the model in limit cases. When $p_D = 1$, in the centre of *FOV*, equation 4.6 is:

$$P_{i,j,k} = \begin{cases} \frac{P_{i,j,k-1}}{P_{i,j,k-1} + p_F (1 - P_{i,j,k-1})} & \text{if } y_{i,j,k} = 1 \\ 0 & \text{if } y_{i,j,k} = 0 \end{cases} \quad (4.7)$$

This means that if in the centre of the *FOV*, the process does not provide a potential measure of the presence of the target, the value that results after correction is always zero.

The second extreme case is when $p_D = 0$, in the border of the *FOV*. In this case, the probability of a false alarm is no longer defined. There are no longer real measurements and we only use the propagation steps of estimation. This situation could occur in adversarial

environment conditions: smog, occlusions, etc.

Before passing to the step of propagation it is necessary to normalise the PMF describing the probability map to obtain always a sum over all the cells that is equal to 1. This is true if we consider that one and only one target is always present in the zone. This implies that the magnitude of the values in the probability map is highly dependent on the discretisation and the size of the mission area.

To face a scenario with different targets a possible solution is to save one map for each detected target and one map where only no detection measurements are updated. In this case, it is necessary to implement a strategy to not only identify a target but also recognise and distinguish it from the others. To finally put all the data together and obtain a single map, $P_{j,k}$, describing the probability that a cell contains at least one target we can use the complementary probability of the event that no target is present in a cell: $\tilde{P}_{i,j,k} = P(X_{i,k} \neq x_j, X_{i+1,k} \neq x_j, \dots, X_{N_{det},k} \neq x_j)$, where N_{det} is the number of target detected so far. Making the hypothesis of independent events it is obtained:

$$P_{j,k} = 1 - \prod_{i=1}^{N_{det}} \tilde{P}_{i,j,k} \quad (4.8)$$

4.1.2. Prediction Step

If we consider the general case where unknown moving targets are present, it is necessary to add a prediction step. The information available to the UAV can change over time, especially in areas not under observation. This step is also essential to account for the sampling and analysis frequency of the measurements. The probability that the target is in cell l , given that it was previously in cell j , is given by:

$$P(X_{i,k+1} = x_l) = \sum_{j=1}^M P(X_{i,k+1} = x_l, X_{i,k} = x_j) \quad (4.9)$$

$$= \sum_{j=1}^M P(X_{i,k+1} = x_l | X_{i,k} = x_j) P(X_{i,k} = x_j) \quad (4.10)$$

where M is the number of total cells of the grid. $P(X_{i,k+1} = x_l | X_{i,k} = x_j)$ is the probability of transition of target from the cell j to cell l . In the Paragraph 4.1.2 the procedure to characterise this probability is described.

Practical Issues

In this section, some implementations are described in detail, as they are considered interesting for the readers. The first is to recreate the random process of the sensor in the simulation, while the second is to create the state transition matrix that allows the prediction of the target's motion.

Measures To be consistent with the Bayesian updating of equation 4.6 and, in particular, with the definitions of detection probability, p_D , and false alarm probability, p_F , during the simulation, it is necessary to account for the randomness of the measurements. To do this, a random value m between 0 and 1 is generated for each point within the *FOV*. Based on the result of m and the real presence or absence of the target at that point (known since we are simulating), the appropriate formula from equations 4.6 is selected, as reported in Figure 4.1.

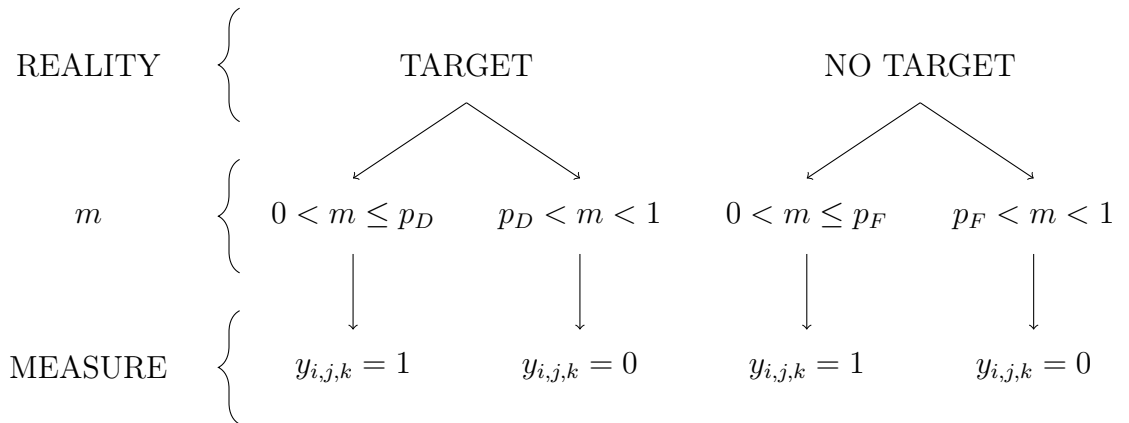


Figure 4.1: Scheme for reproducing randomness in measurements

State Transition Matrix construction The only assumption done on the target dynamics is the maximum velocity, V_{max}^t . No direction is privileged. The maximum velocity is used to diffuse the probability map information. In particular, it is used to define the number of samples of the UAV trajectory to properly propagate the target dynamics.

In the thesis, two temporal discretisations are considered. The first one, $K_{t,1}$, is the one used to evaluate the cost function of the optimisation, see Section 5.2.2 for some considerations and Section 4.2.2 for a suggestion in choosing a value. This parameter is chosen by the user. The second one, which we are considering here, is $K_{t,2}$. This is used to sample the BP trajectory obtained from the optimisation. This value depends on the spatial discretisation of the zone of interest, K_s , and on the final time of the mission,

t_f . From K_s the distance between the centre of the cells, δx , is retrieved. We impose $V_{max}^t = \delta x / \delta t$ to find a lower bound for the interval of time to pass the information from one cell to the adjacent ones, 8 because also the cell in the diagonals are taken into account. Finally, $K_{t,2}$ is calculated as $K_{t,2} = t_f / \delta t$. To calculate the state transition probability we consider also that a target can stay stationary in its current cell, 9 possibilities of moves in a general case when no obstacles are encountered. To propagate the information of the probability map $P_{i,k|k}$ it is easier to reshape the problem and define the state transition matrix A such that:

$$P_{i,k+1|k} = AP_{i,k|k}. \quad (4.11)$$

$P_{i,k+1|k}$ and $P_{i,k|k}$ are both $M \times 1$ matrices representing the probability map. The probability map is reshaped by concatenating each row to form a one-dimensional array. A is of dimension $M \times M$ and each element a_{vz} represents the probability that a target in cell v moves to cell z . In this case, where information is propagated only to adjacent cells, the matrix A is sparse, meaning that most of its elements are zeros. In addition, it is characterised by the propriety $\sum_{v=1}^M a_{vz} = 1$.

The state transition matrix is used to diffuse the information in a step similar to the Markov Chain theory. Given a fixed environment, the state transition matrix is constructed only once. A matrix representing the environment, Env , is used to build A . Env is made of zeros where obstacles are defined and ones where the target can freely move. In addition, a frame of zeros is added to Env to ensure that the diffusion process remains inside the zone of interest. The code to build the matrix A is reported in Appendix C. A brief explanation is given here. Once the frame of zeros to Env is added, an outer double for loop is performed to go through all the elements of Env . For each element, two steps are done. First, the sum of all elements of the small 3×3 matrix, $Neigh$ (centred in the actual element), is performed to know the number of possible moves that the target can do, N_{moves} . Then, another double for loop is done to cross all the elements of $Neigh$ and if there is a 1 it assigns to the respective index in A the value of $1/N_{moves}$. After these loops, it is necessary to set entirely to zero the rows of A whose index corresponds to a starting cell v that contains an obstacle.

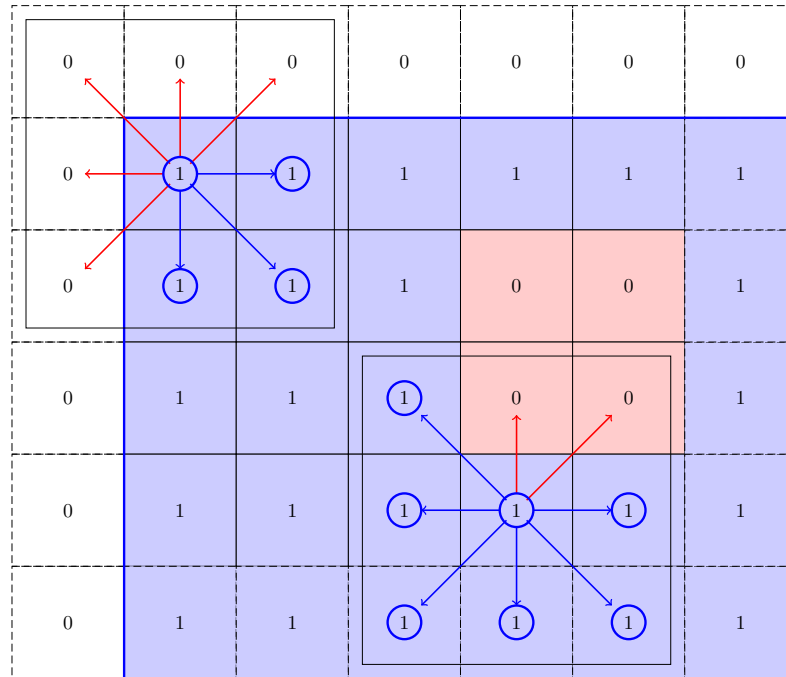


Figure 4.2: Example of *Env* matrix for state transition matrix construction, top-left corner portion. Examples of two *Neigh* matrices (3×3) with the possible movements of the central cell.

4.2. Trajectory Optimisation

It should be recalled that the trajectory is described by Bernstein polynomials in the form shown in equation 2.1. The optimisation parameters are the control points of the spatial trajectory, thus along the two directions x_1 and x_2 . The number of control points, and therefore the degree of the BP, is fixed before the optimisation process and does not vary. The elements that constitute the optimisation problem have therefore been rewritten using the variables, the control points. Thanks to the assumption of a differentially flat system, the dynamics and constraints on the control inputs are imposed via equations 3.8 and 3.9. Equation 3.10 contains the initial and final conditions of the trajectory. Additionally, given the position of the drone and the target, the detection probability is calculated using equation 3.17. The last two fundamental elements, which are explained in the following section, are the objective function and the choice of the initial guess trajectory.

4.2.1. Objective Function

The heart of an optimal search problem is the definition of the objective function, also called the cost function. The optimiser searches for the solution that minimises it. In this section, the exponential detection model is presented. It was first described in [36] and then revisited in [37] and others. The objective function is defined here as the risk that a fleet of UAVs fails to detect the targets in a zone of interest by the end of a search mission. Given the instantaneous detection rate, $r(\eta(t), d(t))$, the exponential detection model is derived from the following two principles:

1. the probability of detection in the short time interval $[t, t + \Delta t]$ is $p(\Delta t) = r(t)\Delta t$
2. detection events in all such non-overlapping time intervals are independent.

The probability of detection failure is given by $q(t) = 1 - p(t)$. With the two assumptions and making Δt tends to 0 it is obtained:

$$\begin{aligned} q(t + \Delta t) &= q(t) \cdot q(\Delta t) \\ &= q(t) \cdot (1 - r(t)\Delta t) \end{aligned} \quad (4.12)$$

$$\frac{q(t + \Delta t) - q(t)}{\Delta t} = -q(t) \cdot r(t) \quad (4.13)$$

$$\dot{q}(t) = -q(t) \cdot r(t) \quad (4.14)$$

The solution over a mission duration t_f is given by:

$$q(t_f) = e^{-\int_0^{t_f} r(t)dt}. \quad (4.15)$$

In Section 3.1.2, the instantaneous detection rate is also described with BP control points of the UAV trajectory and the target position. Since the description of the target is conditionally deterministic, the target variable has to be changed with respect to the equation 3.17. From the realisation \mathbf{x}^t to the random variable \mathbf{X}^t , obtaining as instantaneous detection rate the random function $r(\mathbf{P}_{i,N}^u, \mathbf{X}^t, t)$ and consequently the following expression:

$$q(\mathbf{P}_{i,N}^u, \mathbf{X}^t, t_f) = e^{-\int_0^{t_f} r(\mathbf{P}_{i,N}^u, \mathbf{X}^t, t)dt}. \quad (4.16)$$

Given the area of interest Δ , \mathbf{X}^t represents the set of possible positions that the target can

assume. Given our two-dimensional problem, this random variable has two components, corresponding to the position along the x_1 and x_2 axes, respectively. In the numerical application, the cardinality of the possible solutions depends on the spatial discretization, a factor detailed in Section 5.1. Each possible position is associated with a probability of target presence. The probability distribution over the area of interest has been defined as a probability map in Section 3.2 and is here represented using the notation $f_{\mathbf{X}^t}(\mathbf{X}^t)$.

The risk of no detection becomes a random variable, and its actual value cannot be explicitly minimised. Instead, it is possible to minimise the expected value, conditioned on the probability density function of the target: $f_{\mathbf{X}^t}(\mathbf{X}^t)$. It therefore becomes necessary to integrate the cost function, weighted by the PDF, over the spatial domain Δ :

$$J = \mathbb{E}[q(\mathbf{P}_{i,N}^u, \mathbf{X}^t, t_f)] = \int_{\Delta} e^{-\int_0^{t_f} r(\mathbf{P}_{i,N}^u, \mathbf{X}^t, t) dt} f_{\mathbf{X}^t}(\mathbf{X}^t) d\mathbf{X}^t. \quad (4.17)$$

This is the objective function considered for the problem with one UAV. It is possible to extend the concept and the formula for more than one member.

$f_{\mathbf{X}^t}(\mathbf{X}^t)$ and Δ depend on the considered scenario.

- Scenario 1: $f_{\mathbf{X}^t}(\mathbf{X}^t) = 1$ and Δ coincides with the target position.
- Scenario 2: $f_{\mathbf{X}^t}(\mathbf{X}^t)$ is described by the Beta distribution and $\Delta = \Omega$.
- Scenario 3: $f_{\mathbf{X}^t}(\mathbf{X}^t)$ is given by the probability map and Δ is the entire zone of interest of the mission.

4.2.2. Initial Guess of the Optimisation

Most numeric optimisation methods necessitate an initial guess. In the context of an optimal control problem, this guess consists of a possible solution, so in the space of the solution. In our case, the BP control points describe a trajectory from the initial to the final position. The solver uses this guess trajectory to evaluate the objective function. It then iteratively refines its candidate solution to minimise the objective function, stopping when it reaches a local minimum or when other conditions are fulfilled, such as the maximum number of function evaluations, the maximum number of iterations, and a step size less than the defined step tolerance. A well-chosen initial guess can significantly affect the optimisation process by directing the solver's search to smaller regions of the search space, potentially reducing the computational time substantially. In some situations, such as when multiple local minima are present, the initial guess can be crucial in determining whether the solver finds the optimal or sub-optimal solution.

Before proceeding, a brief note on the nomenclature is provided. From now on, the final point where the drone must arrive at the end of its mission is referred to as the waypoint. This point can either be fixed or left free in the optimisation process. The nomenclature becomes more meaningful in the context of an online planning problem. Indeed, when a drone begins its mission, it requires a guiding trajectory to follow and a recovery point to reach. Later, as the mission progresses, additional information may become available, leading to local modifications of the trajectory. However, if desired by the user, this point can remain unchanged.

Straight Line

The simplest and trivial initial solution is a straight-line trajectory from the starting point to the waypoint with a constant speed. Therefore, the control points of the BP are chosen to be aligned in the defined direction and equidistant from each other. The number of points depends on the user's choice, see the considerations in Section 5.2.1. To obtain a feasible trajectory in terms of dynamics, the chosen mission time, t_f , must be selected in relation to the maximum speed achievable by the drone and the minimum distance ideally travelled, which is a straight-line trajectory in the absence of obstacles. In the presence of obstacles, it is more challenging to obtain an initial guess that is already feasible, i.e., one that avoids them. Therefore, it is advisable to choose t_f with a safety margin in mind. This is necessary if the initial and final conditions and the mission time are fixed in the optimisation framework. Otherwise, there are two interesting possibilities for setting up the optimisation problem: fixing t_f but not the waypoint, or fixing the waypoint and leaving t_f free. Some examples of these two kinds of trajectories are reported in Chapter 5.3.2.

RRT

Ideally, an initial guess that avoids obstacles would be preferable. This way, the optimisation algorithm could focus on enforcing dynamic constraints, if they are not met, and minimising the probability of not detecting a target. A viable approach involves generating a guess trajectory using a standard motion planning algorithm, such as the *RRT* algorithm. This section describes the main steps to obtain it, the challenges, and possible improvements. More examples, additional figures, and the code for *RRT* are provided in Appendix D because this method is not used in the remainder of the work.

Once a trajectory is retrieved with a basic implementation of *RRT*, the degree of the BP, N , that we want to get is chosen. This degree must be greater than 1 to avoid obtaining a

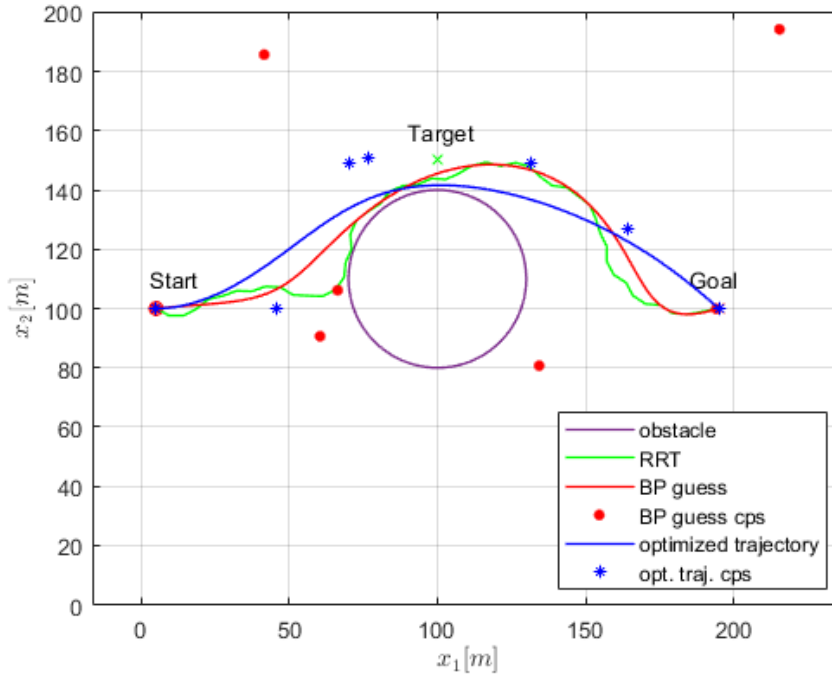


Figure 4.3: Example of initial guess from *RRT* method with $N = 6$

straight line. At the same time, N has to be strictly less than the number of nodes of the *RRT* trajectory. In addition to the trajectory in terms of node positions, it is necessary to know the time instant of each node. From the definition of BP, equation 2.1, and fixing the extremity points of the trajectory, it is possible to inverse the system and obtain the BP control points that impose the passage of the BP in $N + 1$ equidistant points of *RRT* trajectory, including the two extremities.

In Figure 4.3, an illustrative example is provided, showcasing both the trajectory generation process described above and the main limitations of this implementation. It can be observed that the control points obtained from the inversion of the problem, shown in red, are very "distant" from those optimised, shown in blue. This discrepancy is the main reason why the optimisation takes longer, compared to starting with a straight initial trajectory. Another drawback is that the behaviour of the red trajectory, and thus its control points, is random. This is due both to the nature of the *RRT* result, green trajectory, and to the choice of polynomial degree N . The behaviour as N changes does not follow a defined pattern, which makes the choice of N very challenging and specific to each environment in terms of obstacles. Another issue is the system inversion. In the implemented code, matrix inversion is performed for non-singular square matrices, but there are cases where the system is singular. To address this situation, a method such as the pseudoinverse should be implemented.

For the reasons mentioned, this method will not be considered further in the thesis. However, a more detailed study might still lead to better results. Some potential improvements include adapting the *RRT* algorithm to consider detection. Indeed, the currently implemented algorithm does not consider a priori knowledge of the target position. Additionally, applying post-processing to the trajectory obtained with *RRT* could yield a smoother solution. For instance, using a polynomial of a fixed degree that minimises the mean squared error, rather than enforcing passage through specific points, could enhance the trajectory. Once this polynomial has been obtained, the respective BP can be found using a change-of-basis matrix, and consequently its control points.

ODO - Optimisation with Detection Only

The idea proposed in this section is to perform a first optimisation to obtain the initial guess, which is then used as input for the second and final optimisation. The need arises primarily in response to a problem different from the high computational time of infeasible initial guesses, as was the case with the proposed use of *RRT*. Consider the following example. The environment contains a circular obstacle with its centre slightly off-centre with respect to the line connecting the start point and the waypoint. Additionally, we have prior information indicating that there is a target to be controlled in the direction in which the obstacle is offset, see Figure 4.4a. Choosing the straight-line initial guess results in the obstacle being avoided, but from the opposite side of the presumed target. This occurs because the optimiser first attempts to satisfy the constraints and then minimises the criterion. The optimisation algorithm favours avoiding the obstacle. What is desired instead is to prioritise passing over the target, even if it requires a longer path.

The initial guess of the first optimisation is the simple straight trajectory with constant speed. To primarily facilitate detection, what is done is to set up the optimisation problem without the obstacle avoidance constraint, equation 3.20. The obtained trajectory is therefore influenced exclusively by the prior knowledge of the target. If the resulting trajectory already allows avoiding obstacles with a certain safety margin, then the initial guess of the second optimisation will be the result of the optimisation itself. Otherwise, the second optimisation will be necessary to obtain a feasible final trajectory from this point of view as well.

However, an important observation must be made regarding the first optimisation. For the algorithm to be influenced by the target and minimise the objective function, allowing it to approach the target, the target must be within the field of view *FOV* of the initial guess for at least one instant of time. The *FOV* of the drone depends on the characteristics of the sensor. For a real mission, it makes sense for the area of interest to be much

more extensive than the UAV's FOV . What may happen is that information about the target lies outside the cumulative FOV of the initial trajectory, making it unavailable for planning. Instead, it is desirable to exploit this a priori knowledge. To achieve this, it was decided to define two different FOV s. One virtual, FOV_{ODO} , is to be used in the first optimisation, which is defined based on the temporal discretisation $K_{t,1}$ so that the entire area of interest is covered. The second FOV is the one we have already defined based on the characteristics of the sensor.

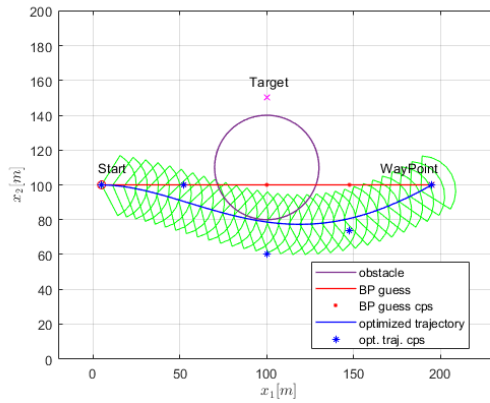
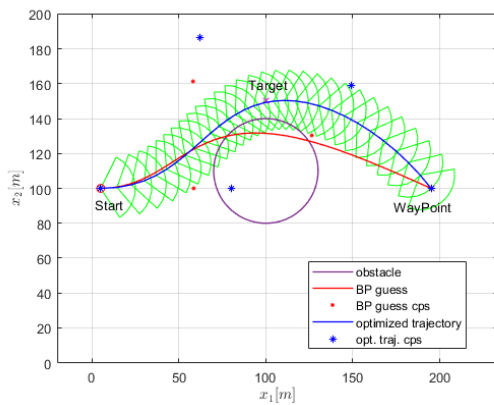
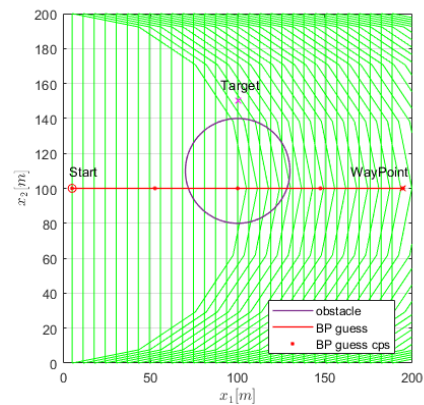
(a) sensor FOV for both optimisations(b) FOV_{ODO} for 1st opt. and sensor FOV for 2nd opt.(c) straight initial guess FOV_{ODO} of 1st opt.

Figure 4.4: Initial guess with ODO

5 | Results

This chapter presents the simulation results to evaluate the performance of the proposed methodology. After a recap of the modelling hypotheses and the resulting optimal control problem, a discussion follows on the choice of algorithm parameters, polynomial degree, and discretisation step. The simulation results are then presented, analysing obstacle avoidance, target detection, and computational time.

5.1. Optimal Problem Formulation

The hypotheses are:

1. The drone is assumed to fly at a constant altitude relative to a flat ground. Bidi-
mensional problem.
2. Dynamic relation between the pitch angle and the horizontal velocity of the drone
is not considered.
3. The detection sensor is rigidly mounted and oriented in the drone's motion direction.

These assumptions led to the choice of the dynamic system reported in equation 3.1. The optimal problem can be formulated as follows.

Given:

- the UAV trajectory, $\mathbf{x}_N^u(\mathbf{P}_{i,N}^u, t)$,
- the target random position $\mathbf{X}^t(t)$ with probability distribution $f_{\mathbf{X}^t}(\mathbf{X}^t)$,
- the instantaneous detection rate, $r(\mathbf{P}_{i,N}^u, \mathbf{X}^t, t)$,

find the sequences of control points $\widehat{\mathbf{P}}_{i,N}^u$ that minimise the probability of non-detection over the area Δ and within the time mission domain t_f :

$$J(\mathbf{P}_{i,N}^u) = \int_{\Delta} e^{-\int_0^{t_f} r(\mathbf{P}_{i,N}^u, \mathbf{X}^t, t) dt} f_{\mathbf{X}^t}(\mathbf{X}^t) d\mathbf{X}^t,$$

such that the following equations are verified:

$$\begin{aligned}
V_{min}^2 &\leq (\dot{\mathbf{P}}_{1,i,N}^u)^2 + (\dot{\mathbf{P}}_{2,i,N}^u)^2 \leq V_{max}^2, \\
-\omega_{max} &\leq \frac{\dot{\mathbf{P}}_{1,i,N}^u \ddot{\mathbf{P}}_{2,i,N}^u - \ddot{\mathbf{P}}_{1,i,N}^u \dot{\mathbf{P}}_{2,i,N}^u}{(\dot{\mathbf{P}}_{1,i,N}^u)^2 + (\dot{\mathbf{P}}_{2,i,N}^u)^2} \leq \omega_{max}, \\
\mathbf{P}_{0,N}^u &= \mathbf{x}_0^u, \quad \angle(\mathbf{P}_{1,N}^u - \mathbf{P}_{0,N}^u) = \Psi_0, \quad \mathbf{P}_{N,N}^u = \mathbf{x}_f^u, \\
\min Dist(\mathbf{x}_N^u(\mathbf{P}_{i,N}^u, t), \mathbf{c}_{O_i}) + r_{O_i} &\geq E.
\end{aligned}$$

Discretisation To numerically evaluate the cost function, it is necessary to discretise both the time and the space domain. The discrete formulation for the general case where the spatial integration domain coincides with the entire area of interest is presented here. We recall that given the number of nodes for each spatial direction, K_s , it is possible to retrieve the cell dimension: $\delta x = \frac{x_{max} - x_{min}}{K_s - 1}$. Instead, with the time discretisation, it is possible to define the discrete-time instants $t_k = (k - 1) \frac{t_f}{K_{t,1} - 1}$ with $k = 1, \dots, K_{t,1}$. The objective function becomes:

$$J(\mathbf{P}_{i,N}^u, K_s, K_{t,1}) = \delta x \delta t \sum_{j=1}^M \exp\left(-\frac{t_f}{K_{t,1} - 1} \sum_{k=1}^{K_{t,1}} r(\mathbf{P}_{i,N}^u, \mathbf{x}_j^t, t_k)\right) f_{\mathbf{X}^t}(\mathbf{x}_j^t). \quad (5.1)$$

$f_{\mathbf{X}^t}(\mathbf{x}_j^t)$ is represented by the probability map. To evaluate $r(\mathbf{P}_{i,N}^u, \mathbf{x}_j^t, t_k)$, it is necessary to obtain the position of the BP at the time instant t_k . To do this, the *de Casteljau* algorithm is exploited; see more details of this algorithm in Section 2.3.

The cost function, and consequently the optimisation, depends on three tuning parameters: the BP control points (number and position), as well as the spatial and temporal discretisation.

5.2. Parameters and variables choice

The present section analyses the influence of these three parameters on the trajectory description and the optimisation problem.

5.2.1. Bernstein Polynomials Degree

The most critical parameter in the optimisation process is the degree of the Bernstein polynomial used to define the trajectory. This section specifically discusses the selection of control points (cps) for the spatial coordinates, x_1^u and x_2^u , within the context of optimising

according to the first approach presented in Section 3.1. We recall that the degree, or order, of a BP is given by the number of cps minus one. Therefore, the degrees of freedom in the optimisation design are influenced by the number of cps that define the spatial trajectory. The determination of cps governing velocity and turning rate relies exclusively on leveraging Bernstein properties. Any reduction in the degree of the velocity and turning rate BPs will inevitably distort, to a greater or lesser extent, the original information imposed by the chosen degree of the spatial trajectory. The initial straightforward observation is that as the degree of BP increases, so does the number of variables that need to be optimised. Two important questions then arise:

- Is there a minimum necessary order to satisfy a given mission?
- Beyond this value, how does increasing the number of cps influence the results?

To answer the first question, some considerations must be made. From the *End Point Values* property, equation 2.4, it is known that the first and last points of the BP coincide with the first and last cps. Considering a trajectory where the ending position differs from the starting one, one should begin by setting a minimum of 2 cps, with a BP of degree 1. The trajectory obtained is a straight line. Always from the same property, it is possible to impose the direction and velocity of the initial and final points. The trajectory at the initial point is tangent to the line connecting the first and second cps, and at the final point, it is tangent to the line connecting the second-to-last and last cps. If the second and second-to-last cps differ from the first and last ones, fixing their values is equivalent to fixing the direction and velocity simultaneously. The angle between points defines the direction, and the distance defines the velocity. Remember that the degree N and the final time t_f also influence the value of extremities velocity, see equations 2.5 and 2.6. In summary, using 4 cps allows us to define the initial and final positions, along with the directions and velocities (excluding zero velocity). To enforce zero velocity at an endpoint, it is necessary to add a coincident cp. This adjustment ensures that the first derivative equals zero at that point. Generally, to set the n^{th} derivative to zero at a specific endpoint, n coincident points are required at that endpoint.

In the simulations that follow, we impose constraints only on the initial position and direction, as well as the final position. With 3 cps, if we impose that the initial direction (from first to second cp) is equal to the direction from first to last cp, the 3 cps are aligned and the resultant trajectory is a straight line. To avoid a simple obstacle, for example, a circle, it is necessary to add a curvature, right or left side, and so another cp has to be added. The minimum number of cps is 4. This can also be regarded as the fundamental block for future sequential online trajectory generation, where the boundary

condition between different blocks requires the initial position and direction to be equal to the final position and direction of the preceding block. The initial mission phase, transitioning from zero velocity to the specified position and direction, can be designed and studied separately. As for the mission's conclusion, it is considered complete once the drone reaches the desired position, regardless of the direction, while adhering to dynamic constraints and environmental considerations.

In the presence of other obstacles or more complicated ones, such as an indoor environment, more cps are needed. There are no definitive rules dictating a specific number concerning the environment and mission at hand, but certain recommendations can be suggested. If a lot of direction changes are expected, it is necessary to add more cps. If cps are added in proximity or in the same position as another node (multiplicity greater than one), the influence in this part of the trajectory will be greater with respect to the rest. The effect is similar to increasing the potential field associated with the area containing these points.

Concerning an upper limit, the situation becomes even more complex. The first consideration is that as N approaches infinity (a very large number), the cps converge to the polynomial that is being described. Although this would theoretically add an infinite number of degrees of freedom, it contradicts the goal of using a representation that relies on a limited number of variables to describe a dynamically feasible trajectory. We will fall into the disadvantages of methods that discretise the trajectory. The principal disadvantage is the computational time required by the optimisation.

5.2.2. Temporal and Spatial discretisation

It was previously mentioned that to evaluate the objective function, both time and space need to be discretised. Now we examine further the reasons and effects of this. Firstly, temporal discretisation, $K_{t,1}$, is performed to evaluate the UAV's position at defined time instances. Spatial discretisation, K_s , on the other hand, is necessary to describe the position of the target. The target's location is represented by the cell it occupies on the probability map. Only once these two positions are obtained for a defined time instant, the instantaneous detection rate function can be calculated, and thus the objective function. As expected, the greater the number of points in both time and space, the more precise the value calculated by the objective function will be. However, this leads to an increase in computational time because there are more combinations of UAV and target positions to evaluate. In particular, spatial discretisation results in a more significant increase in computation time. Two considerations can be made as guidelines for the two discretisation

choices. Regarding time, it is recommended to choose a number of points that respect the sensor’s onboard sampling frequency. On the other hand, for space, it is important that at least one point, in the worst-case scenario, falls within the area described by the FOV. This last consideration is of great importance, particularly when considering the implementation of scenario 3. In this case, the area of integration is the entire zone of interest. The corresponding probability map is composed of points of zero values, obstacles, and points with a value different from zero. What is desirable is not only to drive the drone where there is more information and avoid obstacles but also to exploit to the maximum the available FOV. In other words, the goal is to obtain a trajectory where the FOV does not intersect with the obstacle and captures the maximum amount of information possible. In Paragraph 5.3.2 the results with $t_f = 30$ s for $K_s = 21$, $K_s = 41$ and $K_s = 81$ are reported. Respectively represented in Figures 5.13a, 5.12c, and 5.13b.

5.3. Simulations

In the following section, the results of some significant simulations are reported. After an initial part in which obstacle avoidance is verified, the focus shifts to target detection. The results are obtained using MATLAB built-in function, *fmincon*. It is designed to find the minimum of a constrained nonlinear multivariable function. It has been chosen for its versatility in terms of constraints. It can handle both equality and inequality constraints, as well as linear and nonlinear constraints. Additionally, it is possible to choose the optimisation algorithm from a provided list. The algorithm used for the simulations is the *SQP*, Sequential Quadratic Programming. *SQP* approximates the nonlinear optimisation problem by solving a series of quadratic subproblems. Each subproblem optimises a quadratic model of the objective function subject to linearised constraints. The computer processor used is Intel(R) Xeon(R) W-2123 CPU @ 3.60GHz. The numerical values used in the simulations to obtain the subsequent results are reported. In the remainder of the chapter, the following values are considered unless stated otherwise. Starting with the variables that directly depend on the drone in question. The values reported in Table 5.1 are taken from the respective manuals, [12] and [13]. Let’s consider, for future simulations, the specifications of the DJI Matrice 100 drone.

	DJI Matrice 100	DJI Matrice 300 RTX
ω_{max} [deg/s]	150	100
V_{max} [m/s]	17	23
V_{min} [m/s]	0	0

Table 5.1: DJI Matrice 100 and 300 RTX drone specifications

Regarding the other variables, the values are reported in Table 5.2.

Variable	Value	Unit
N	4	—
$K_{t,1}$	41	—
K_s	41	—
t_0	0	s
t_f	30	s
E	1	m
FOV^u	120	deg
d_{max}	20	m
d_{min}	0	m
$\alpha = \beta$	3	—

Table 5.2: Simulation Numerical Values

Other variables that are used to customise the problem further are:

- `initial_guess`: 1 if *ODO* or 0 if straight initial guess
- `final_cond`: 1 if final position is imposed, 0 if free
- `deg_elevation`: 1 if degree elevation is applied to velocity and turning rate
- `MaxFunEva`: number of maximum function evaluations for *fmincon*

5.3.1. Obstacle avoidance

Initially, we focus exclusively on obstacle avoidance, excluding target information from our considerations. We select different scenarios and, in the optimisation function, we disregard the probability map, concentrating only on meeting the constraints related to obstacle avoidance. The additional variables reported above assume the following values:

- `initial_guess` = 0
- `final_cond` = 1
- `deg_elevation` = 0
- `MaxFunEva` = 4000

Two different environments are considered. In the first case, a single obstacle is present, centered at (100, 100) with a radius of 30. In the second environment, three obstacles are included, with centres at (50, 90), (100, 120), and (150, 90), all having a radius of 20. In

both environments, the initial point and the final point, referred to as a waypoint, are fixed at $(5, 100)$ and $(195, 100)$, respectively.

Obstacle avoidance Scenario 1

In the first scenario analysed for the obstacle avoidance simulations, only one centred obstacle is considered. The degree of PB varies: $N = 4, 8, 12$. See Figure 5.1.

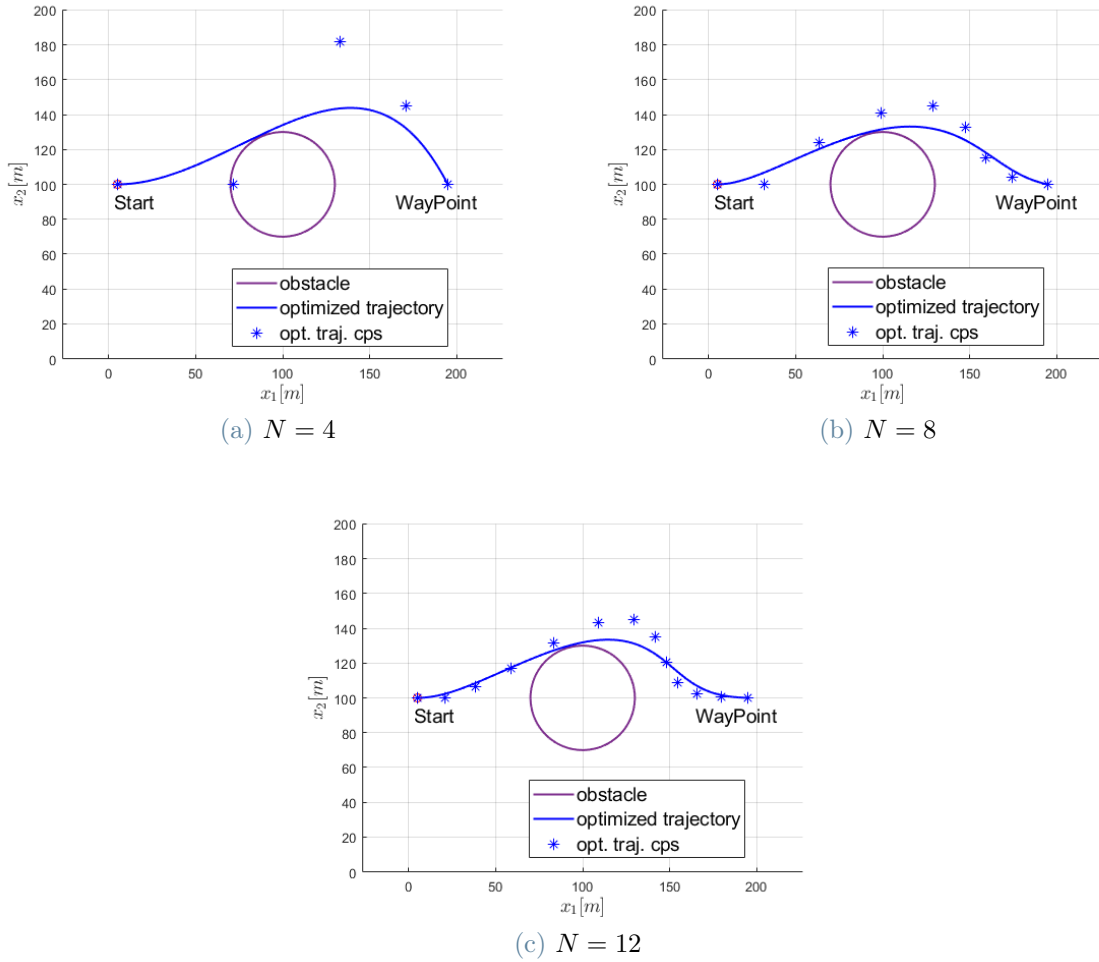
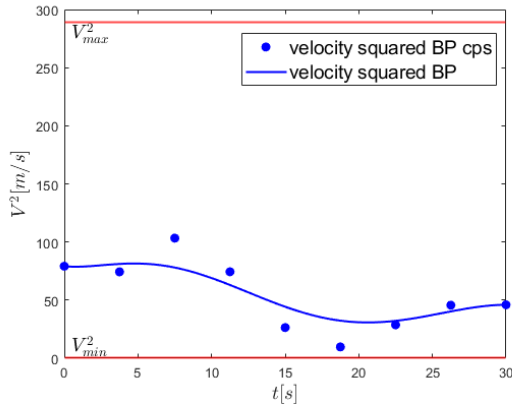
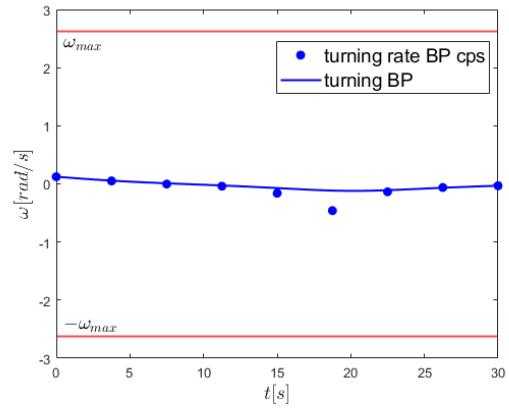
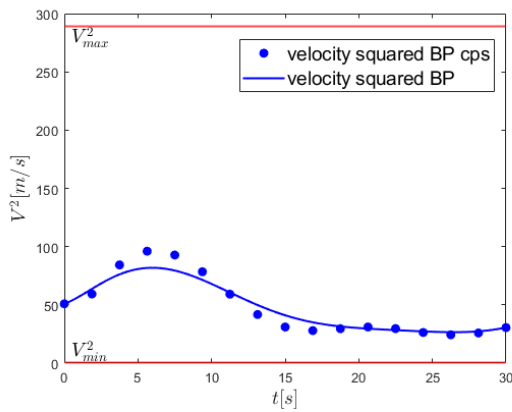
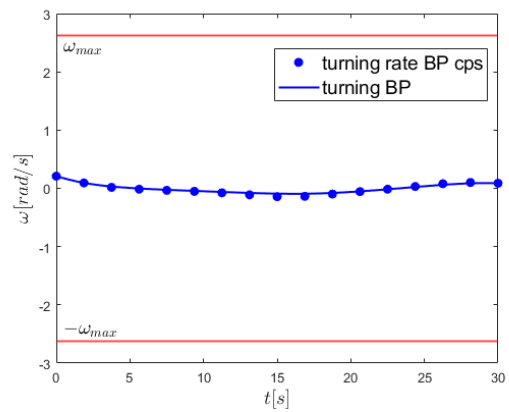
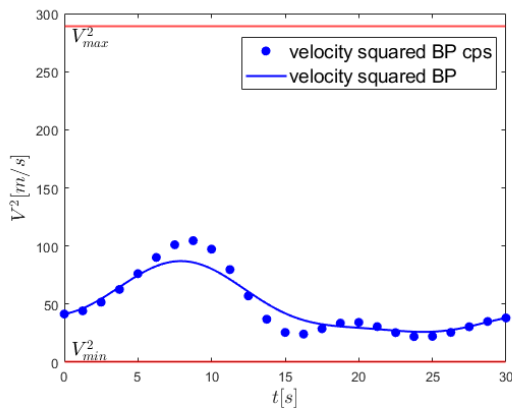
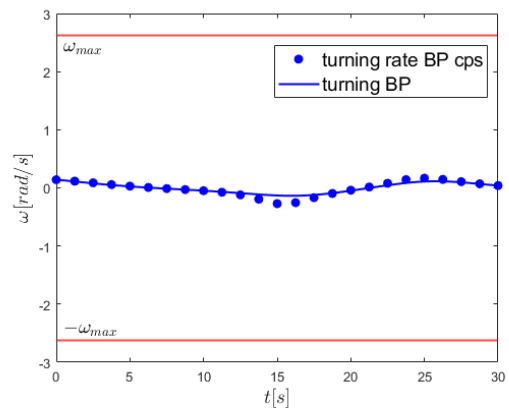


Figure 5.1: Obstacle avoidance Scenario 1 trajectories with different N

All three trajectories are feasible in terms of obstacle avoidance, input constraints, and initial and final conditions. For the initial condition, the angle ψ is fixed to zero. It can be observed that as the degree of the BPs increases, the trajectory gets closer to the obstacle due to the higher degrees of freedom. It is important to remember that in this case, the final time is fixed. Therefore, if a shorter spatial trajectory is obtained, the average speed along the trajectory has to be lower. Between $N = 8$ and $N = 12$, the difference in the optimal solution is minimal. Therefore, it is not worth increasing N beyond those values

(a) V^2 with $N = 4$ (b) ω with $N = 4$ (c) V^2 with $N = 8$ (d) ω with $N = 8$ (e) V^2 with $N = 12$ (f) ω with $N = 12$ Figure 5.2: Obstacle avoidance Scenario 1, V^2 and ω

because the improvement in the trajectory is negligible. As expected, the computational time increases with the degree of the polynomials, see Table 5.3 for numerical values. It can be observed that for both the squared velocity and the turning rate, the degree of the polynomial that describes them is doubled. This is because the product of two identical BPs results in a BP with a degree that is double that of the original polynomial.

BP Degree	$N = 4$	$N = 8$	$N = 12$
C.T. [s]	0.3654	0.7958	1.5016

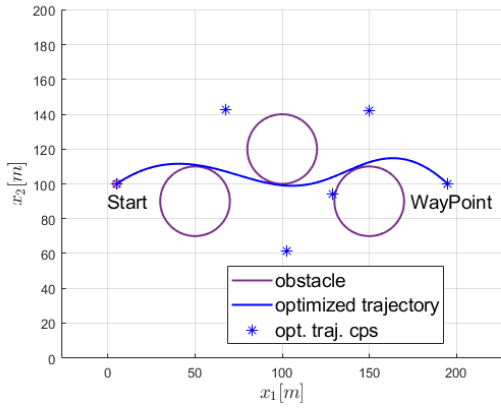
Table 5.3: Obstacle avoidance Scenario 1 Computational Time (C.T.) for different N

Obstacle avoidance Scenario 2

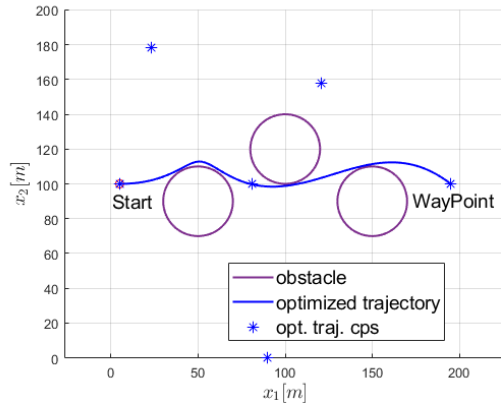
In this second scenario for obstacle avoidance, three obstacles are considered. To avoid obstacles and to impose an initial direction equal to zero, $N = 4$ is not sufficient, even with degree elevation. This is because, even assuming that the input constraints are satisfied, there are not enough control points to start with a heading angle of zero and make three-direction changes to avoid obstacles. It is necessary to add a control point. With $N = 5$ and without degree elevation, the trajectory obtained is not yet feasible. The initial heading angle is not zero (Figure 5.3a), there is a cp of the squared velocity norm that is just below zero (Figure 5.3c), and a cp of the turning rate that is greater than ω_{max} (Figure 5.3e). Even though the trends of V^2 and ω are within the limits, due to the control points describing them, the optimisation process considers the input constraints as not satisfied. During the optimisation process, a lot of solutions are discarded because the convex envelopes of the squared velocity norm and the turning rate do not respect the constraints. In addition, in some iterations, the control points of the squared velocity norm (recall that they are the weights of the turning rate) are negative. First of all, the solution is discarded because the squared velocity norm is less than zero. Additionally, the *Convex Hull* property is not applicable to the turning rate. In Figures 5.3c and 5.3e it can be seen that the envelopes are not well exploited and there is a lot of margin. Degree elevation (DE) for the squared velocity norm and turning rate is performed. From $N = 5$ to $N_{DE} = 8N = 40$.

To compare the computational times, more simulations are done. In particular, $N = 8, 12, 16$ with and without a degree elevation to reach $N_{DE} = 40$ in all cases, Figure 5.4.

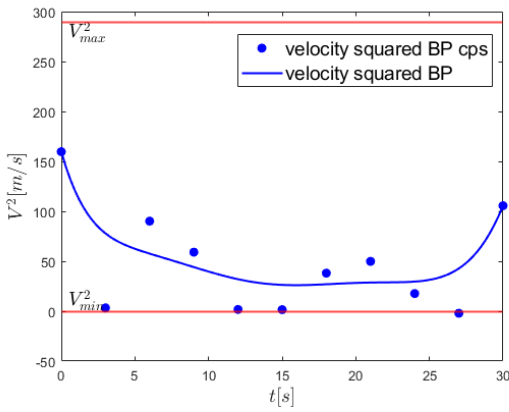
As for the optimisation computational times, what can be observed is that there is no linear relationship between N and C.T. Contrary to what we stated earlier, we would expect that as N increases, the time should also increase. Instead, there is a minimum present, both in the case without and with degree elevation. The first segment of the



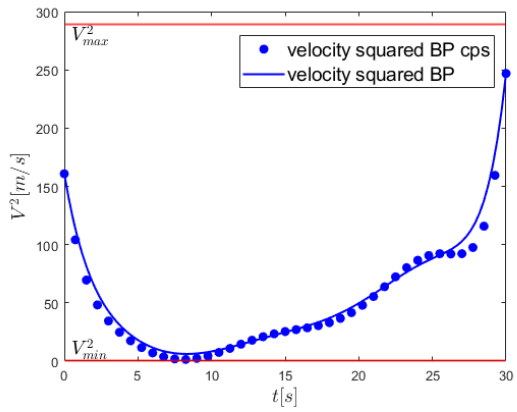
(a) Trajectory without DE



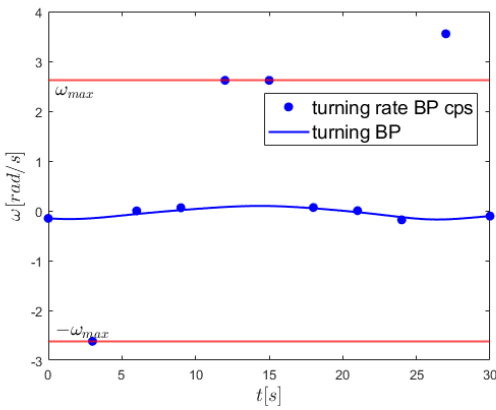
(b) Trajectory with DE



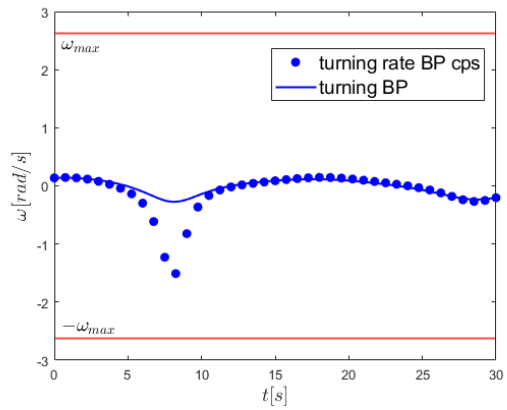
(c) V^2 without DE



(d) V^2 with DE



(e) ω without DE



(f) ω with DE

Figure 5.3: Obstacle avoidance Scenario 2 with $N = 5$, with and without DE

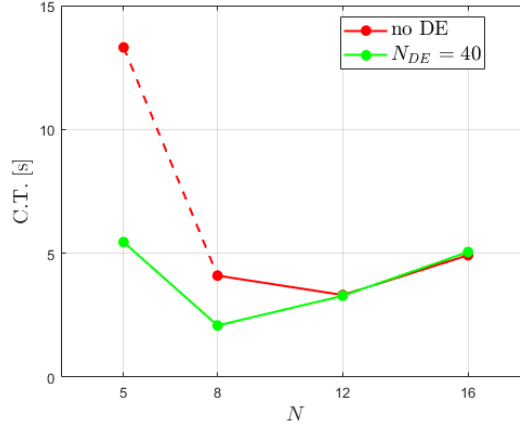


Figure 5.4: Obstacle avoidance Scenario 2
C.T. for different N and with or without DE

red curve is dashed because, for $N = 5$ without DE, the constraints are not met. It can be concluded that for each scenario, there is an optimal number of control points that allows for achieving a feasible trajectory in the shortest possible time. Similarly, there is a degree for the DE that speeds up the optimisation process due to a more restricted convex envelope. Finding a general criterion to determine the optimal number N for each environment in advance does not yet exist, but some guidelines have been provided in Section 5.2.1. As for the optimal degree for the DE, it remains a very challenging task.

5.3.2. Target detection

In this section, prior information about the area of interest is added. Therefore, the influence of this knowledge on trajectory optimisation is analysed. To do this, it is necessary to add the objective function described by equation 5.1 to the problem. In the area of interest considered, there is an off-centre obstacle of centre (100, 110) and radius 30. It is the same environment already used in Figure 4.4b.

Fixed Mission Time and Final Position

In these initial simulations, as has been done so far, a specific waypoint, `final_cond = 1`, and a time mission are selected a priori. As for the prior information, the three scenarios analysed are those reported and discussed in Section 3.2. For scenarios 1 and 2, the first optimisation with only detection, ODO, is done, `initial_guess = 1`.

Scenario 1 The target position is in the cell represented by the point in the discretised grid at coordinates (165, 165). The degree of BP is $N = 4$. Two simulations are reported

in Figure 5.5. The first one is performed without DE, Figure 5.5a, while the second one is with $N_{DE} = 40$, Figure 5.5b.

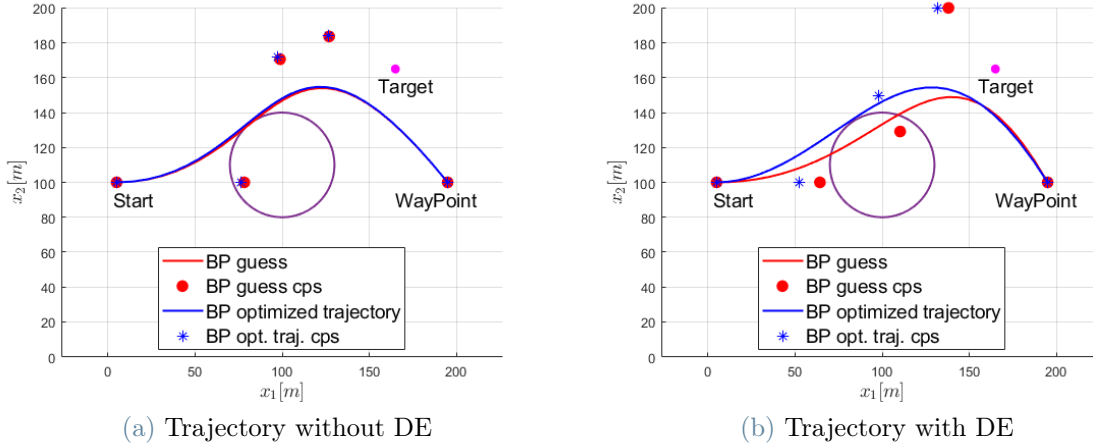


Figure 5.5: Target detection Scenario 1 without and with DE

With the degree elevation, Figure 5.5b, the first optimisation is shorter in terms of computational time because the constraints are met, and the minimum is reached in fewer iterations. Two control points, one from the initial guess and the other from the optimised trajectory are located at the boundary of the area of interest. To ensure that the trajectory remains within the area, we have utilised additional arguments of the *fmincon* function. These are the lower and upper bounds, respectively **lb** and **ub**. They are vectors of length equal to the number of control points, $N + 1$, containing the boundary values of the variables. In this case, $x_{1,min} = x_{2,min} = 0$ m and $x_{1,max} = x_{2,max} = 200$ m. In the subsequent simulation, Figure 5.6, the limits were removed.

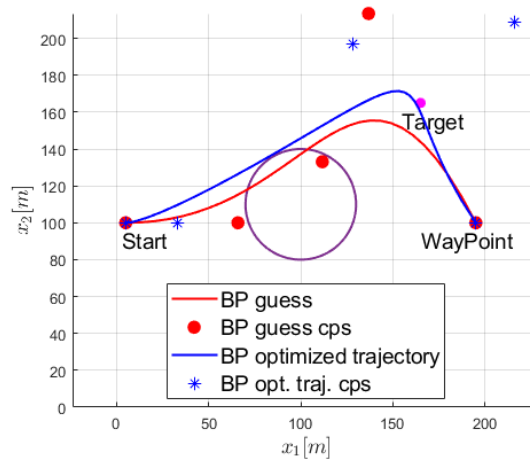


Figure 5.6: Target detection Scenario 1 trajectory with DE and no limits

It can be observed that the trajectory approaches the target more closely. This is possible due to a larger optimisation domain. However, the increased precision is obtained with a significantly higher computation time, as seen in Table 5.4. In this table, two computational times are reported, C.T. 1 and 2, respectively, regarding the first and the second optimisation. As mentioned earlier, the case where we know the target with precision is not very realistic. Despite this, the result obtained is demonstrative. In fact, by using a criterion related to the sensors, it has been possible to obtain a trajectory that is influenced by the areas of the map that contain relevant information.

DE	Limits	C.T. 1 [s]	C.T. 2 [s]	C.T. total [s]
NO	YES	0.5235	0.2089	0.7324
YES	YES	0.2187	0.2366	0.4553
YES	NO	0.2724	3.8101	4.0825

Table 5.4: Target detection Scenario 1 computational times

Scenario 2 The target position is represented by a Beta distribution centred always at coordinates (165, 165) and with a spatial domain of 70 m in both directions, x_1 and x_2 . Therefore, the area is [130, 200] m \times [130, 200] m. In this case, the degree elevation to $N_{DE} = 40$ is performed and the area limits are imposed for the control points. The mission time t_f is always fixed at 30 s. The initial guess from ODO is exploited also in this situation, the red curve in Figure 5.7. In the same figure, the FOV is reported in green and the domain of the Beta distribution in black with its centre in magenta. C.T. 1 = 9.9298 s and C.T. 2 = 6.5930 s.

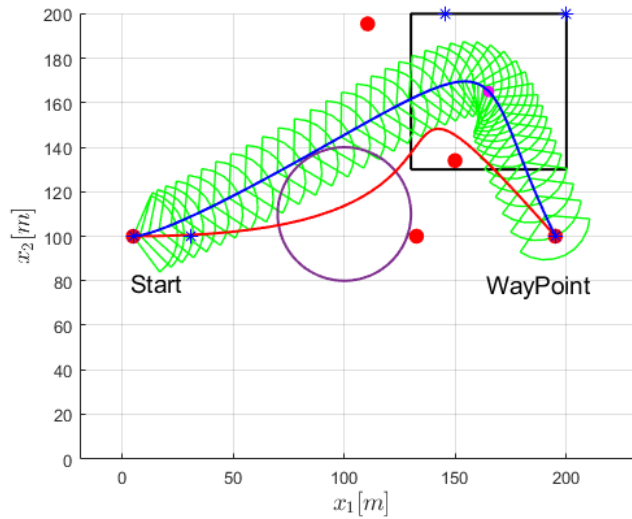


Figure 5.7: Target detection Scenario 2 trajectory with DE

In practice, such a situation can occur. In particular, prior knowledge may come from instruments or information collected directly in the field. However, the location is not precise, and the information corresponds to a rather large area. It is desirable to plan a path to explore the specified area and get more precise information about the target once the path is travelled. This leads to the topic of target localisation estimation discussed in Section 4.1. The next results are obtained by simulating the presence of a fixed target located exactly at the centre of the distribution. Knowing that the target cannot move, the prediction step is omitted. Only the correction step, based on the measurements, and the normalisation are performed.

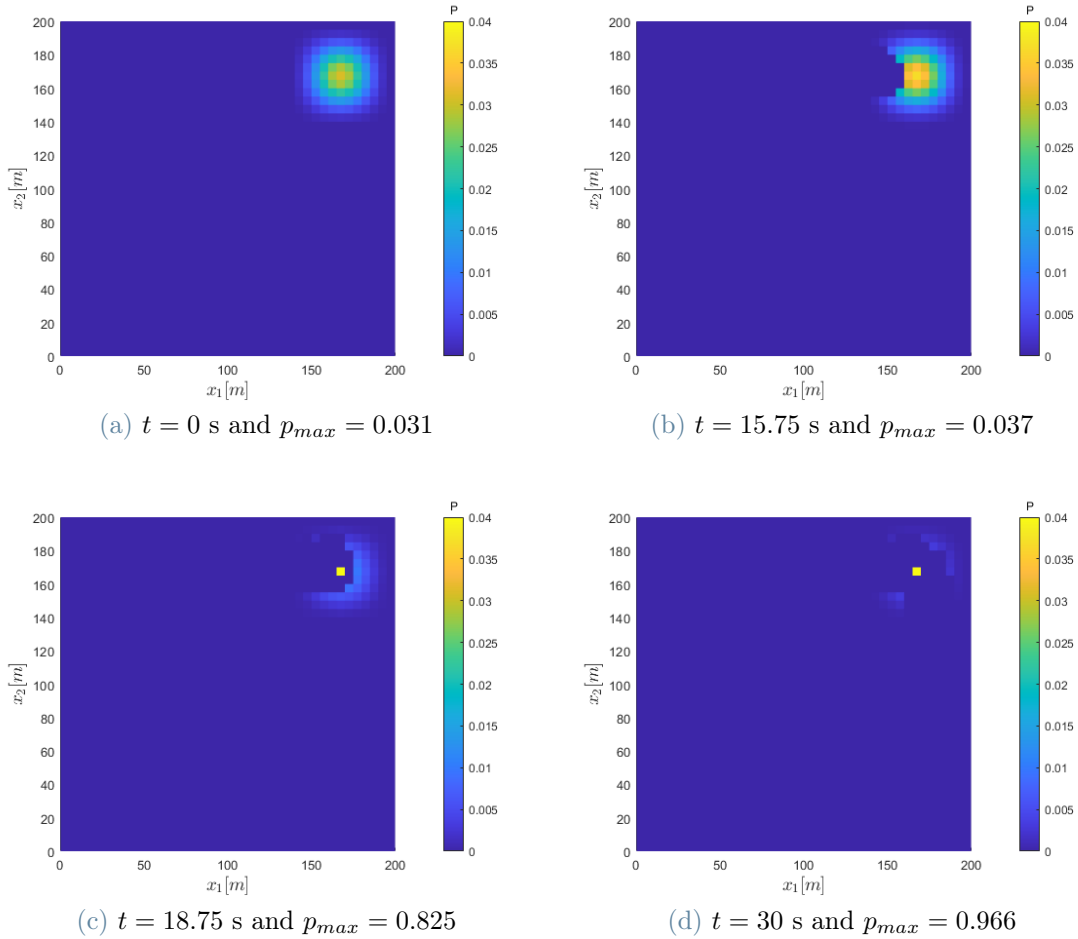


Figure 5.8: Target detection Scenario 2 probability map with ideal updates

Once the optimised trajectory is obtained, shown in blue in Figure 5.7, it is propagated based on the sampling time. The parameters involved in the detection process of the simulation are p_D and p_F . Let's start with the ideal case where $p_D = 1$ and $p_F = 0$. With this setting, it is considered that if a target is within the FOV, the probability of detecting it is 100% and the probability of a false detection is zero. For this reason, we can say that

there is no randomness in the simulation, and we expect to identify the target exactly where it has been placed. Given $t_f = 30$ s and $K_{t,1} = 41$, the time interval is $dt = 0.75$ s. Note that in Figure 5.8 the maximum colour for the probability, yellow, is fixed at the value of 0.04 throughout the entire simulation.

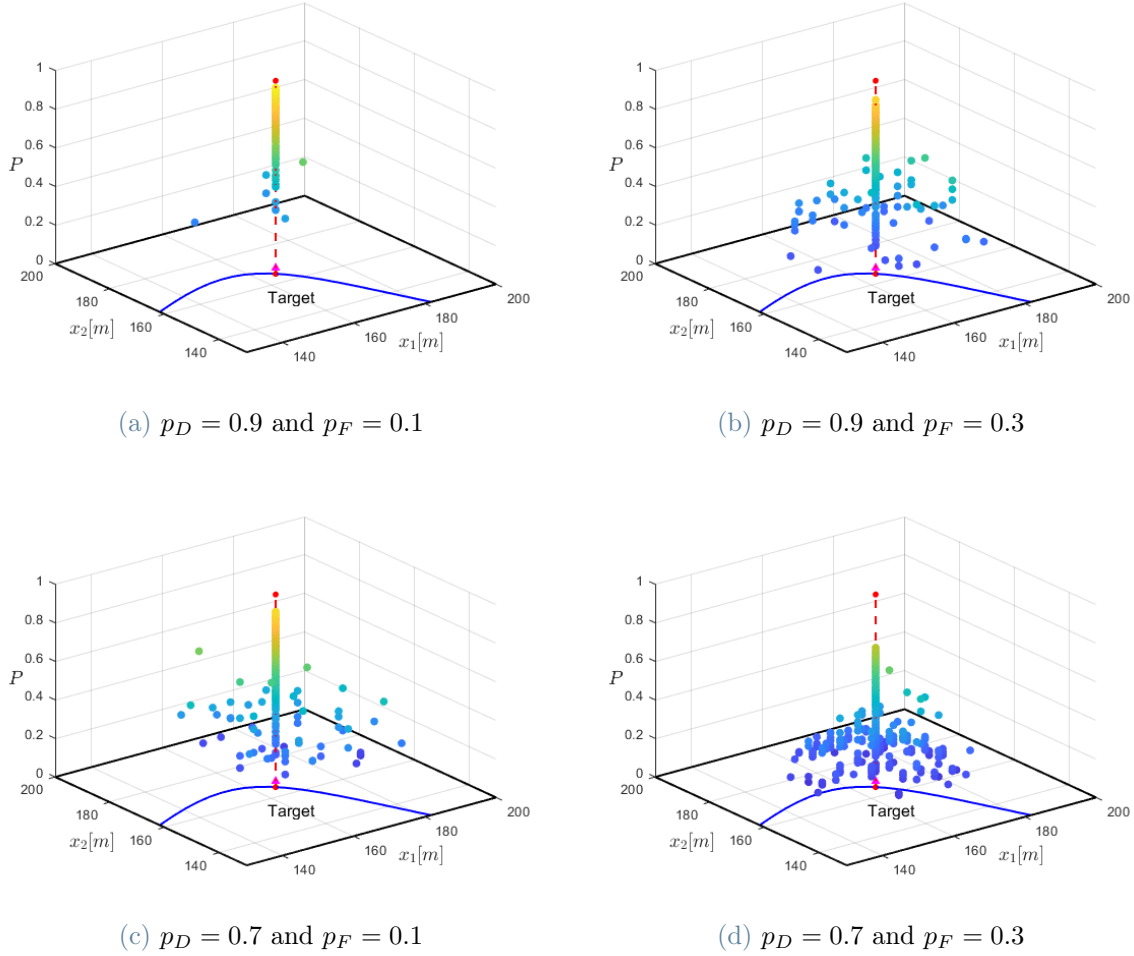


Figure 5.9: Target detection Scenario 2 maximum probability target positions with different settings

The behaviour just described, however, is not very realistic. For this reason, we now introduce the detection probability, p_D , and the false alarm probability, p_F , which are different from 1 and 0, respectively. By doing so, we introduce randomness into the simulation. We conduct an initial analysis assuming that the detection probability, p_D , and the false detection probability, p_F , are constant. Four combinations of p_D and p_F are considered. It is important to remember that the sum of p_D and p_F is not necessarily equal to 1 as they don't represent complementary events. For each setting, 500 simulations are run, and for each simulation, the position with the highest probability on the probability

map is saved. In Figure 5.9, the results are shown. Here are some notes for a better understanding of the following figures. The red dots define the endpoints of the segment perpendicular to the search area with coordinates equal to the true position of the target. The magenta triangle indicates the maximum probability known a priori before taking the trajectory.

In Figure 5.9a, it can be seen that approximately 99% of the points match the actual target position. The probabilities of these positions are distributed in the upper part, ranging from approximately 0.4 to nearly 1, but with a greater concentration for higher values. When the false alarm probability increases, as shown in Figure 5.9b, it is observed that the maxima drift from the actual position, particularly those with lower probability values. The peak probability decreases compared to the previous case. We observe a similar behaviour also in the case where only the detection probability is modified (decreased), as shown in Figure 5.9c. In the final situation, 5.9d, both probabilities have been modified, representing a worse scenario compared to the initial one. There is a lower probability of correctly detecting the targets and a higher probability of false alarms. Firstly, the probability values have decreased, both for positions coinciding with the true target and for those that deviate. Additionally, the total number of the latter has increased compared to the previous cases. Approximately 62% of the points match the actual target position.

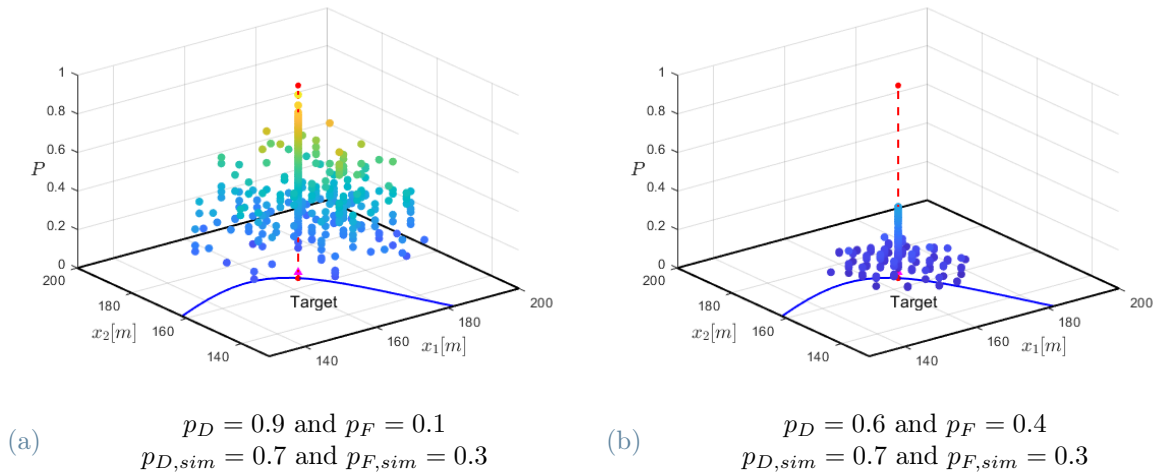


Figure 5.10: Target detection Scenario 2 maximum probability target positions with different p_D and p_F used for correction step and simulation

In a realistic situation, the selection of p_D and p_F may be incorrect and therefore does not reflect what happens when collecting measurements. In Figure 5.10, two sets of values of p_D and p_F were considered for the correction step, while in the simulation to reproduce the randomness of the process different values were used, $p_{D,sim} = 0.7$ and $p_{F,sim} = 0.3$.

In Figure 5.10a we have set $p_D = 0.9$ and $p_F = 0.1$, more optimistic values compared to those used in the simulation. What happens is that we are placing more confidence in false detection measures. As a result, the 'cloud' of positions that do not match reality, the red dashed line, generally has a higher probability than the positions predicted by previous simulations. Additionally, underestimating the false alarm probability results in more decoys, about 50%. In Figure 5.10b, we are instead considering more pessimistic values for both parameters, $p_D = 0.6$ and $p_F = 0.4$. In this case, the probabilities decrease significantly, but the number of decoys also decreases, to approximately 30%.

The second set of settings is with p_D calculated using the instantaneous detection rate function and p_F as a percentage of p_D . For the shape of the instantaneous detection rate function, Beta distribution, it is assumed that recognition is better in the central area of the *FOV* and it degrades towards the *FOV* edges until nothing can be detected.

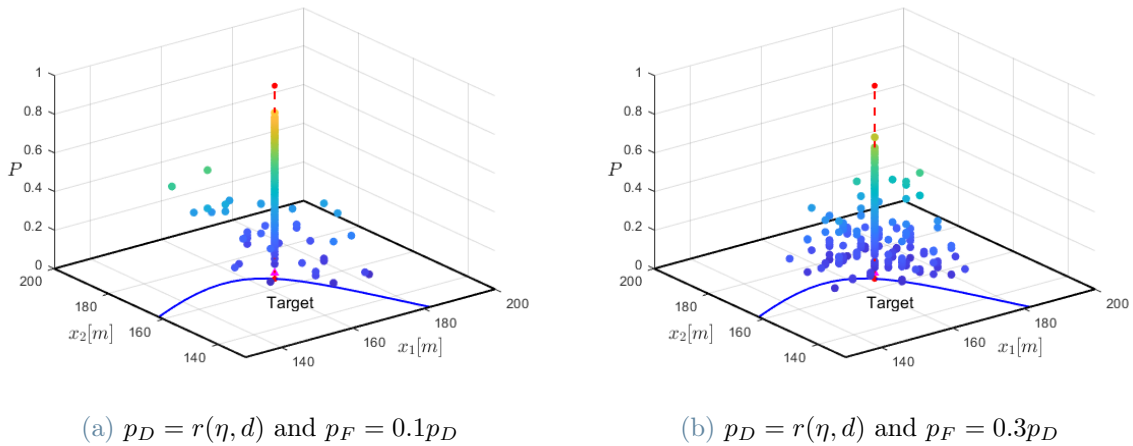


Figure 5.11: Target detection Scenario 2 maximum probability target positions using the instantaneous detection rate function r

We have therefore evolved from a constant model of p_D to the function r , which can assume values between 0 and 1, from Figure 5.9a to Figure 5.11a. This results in a general decrease in probability and an increase in the number of points outside the red dashed segment. Additionally, if p_F increases, as shown in Figure 5.11b, we observe a further decrease in probability values and an increase in decoys.

As expected, the estimation of the position is strongly influenced by the performance of the sensor and the recognition algorithms. Despite this, the results of the estimation are generally positive. The search area has been reduced compared to the initially known one. The probabilities of the maxima are always higher than the central maximum of the initial distribution (magenta triangle).

Scenario 3 In this last scenario, there is no a priori knowledge about the possible position of the target. The only information available a priori is that a target is present and that it can reach a maximum speed of $V_{max}^t = 5$ m/s. The target also has the option to remain stationary, which is considered in Figure 4.2. The prediction step also called the propagation step, is in this case necessary. To propagate the information, V_{max}^t , the spatial discretisation K_s , and the mission time t_f are considered. The procedure is explained in Paragraph 4.1.2. Since there is no a priori information, the initial guess trajectory is considered the straight line from the starting position to the waypoint, $\text{initial_guess} = 0$. Remember that for the following simulations, we consider $N = 4$ without degree elevation. As stated in Section 4.2.1 in this scenario the area of integration, Δ , of the objective function 4.17 is the entire area of interest and the initial probability of each available point is $1/N_{free}$.

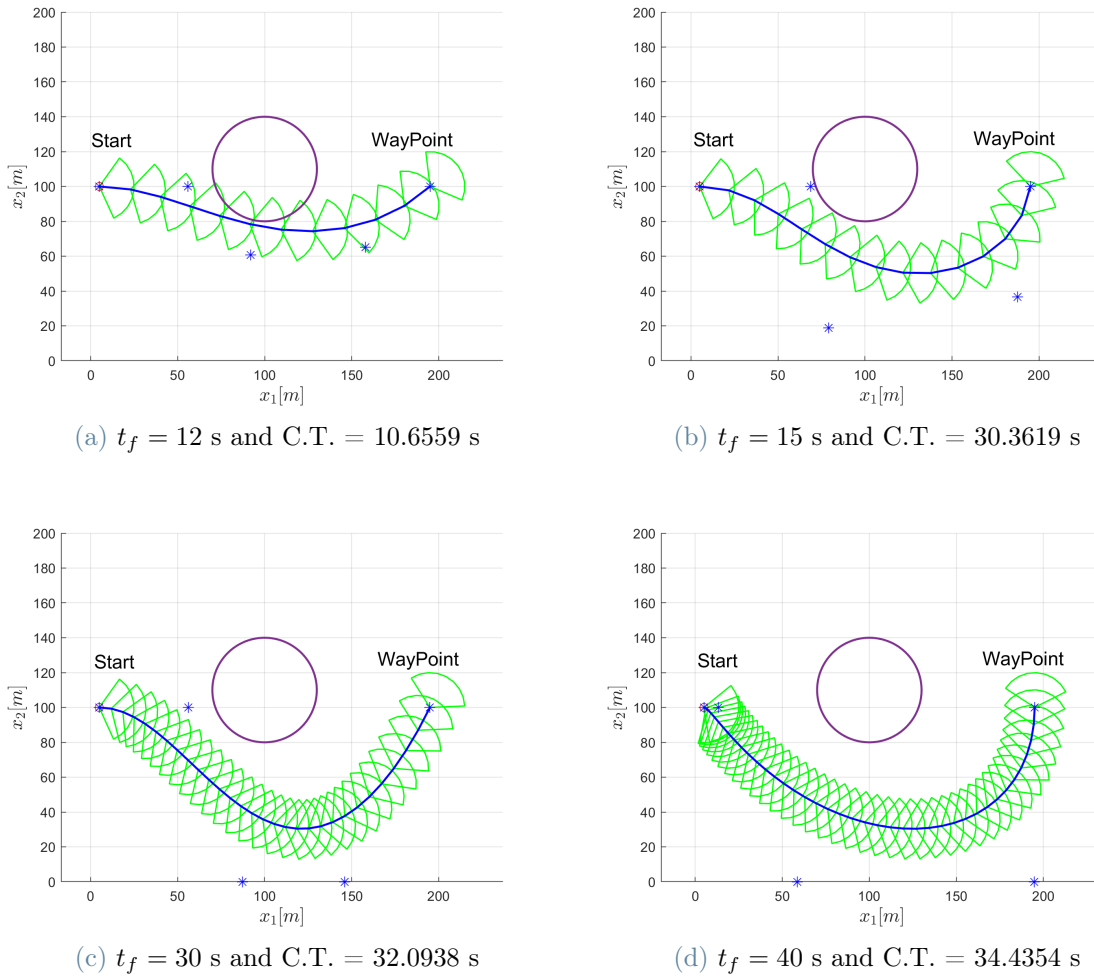


Figure 5.12: Target detection Scenario 3 trajectory design with variable t_f

The first analysis that can be made is on the variation of the final time t_f . It was decided to start with $t_f = 12$ s, Figure 5.12a, because for shorter times there is no solution that respects all the various constraints. With this time available, the trajectory obtained to respect the constraints passes quite close to the obstacle (beyond the minimum safe distance, however). With longer mission times, we notice that the minimisation of the non-detection criterion ensures that the FOV is exploited to the maximum to search for new information. It is therefore observed that all FOVs at different time instants do not intersect the obstacle. Figures 5.12b, 5.12c, and 5.12d show the trajectories calculated with $t_f = 15$, 30, and 40 s. As a negative consequence, the computational times have increased considerably compared to the two previous scenarios. In the first scenario, only a single point was analysed, and in the second, a reduced set of points. In this case, however, all discrete points on the probability map are considered. It is precisely for this reason that the optimisation takes full account of the capabilities of the sensor model. This is a desirable feature and a strength of the proposed method.

For the second analysis, it is considered that t_f is equal to 30 s. The computational times and the obtained trajectory are analysed as the spatial discretisation varies. So far, with $K_s = 41$, square cells with a size of $\delta x = 5$ m have been considered. The next results are obtained with $K_s = 21$ and $K_s = 81$, 10 m and 2.5 m, respectively.

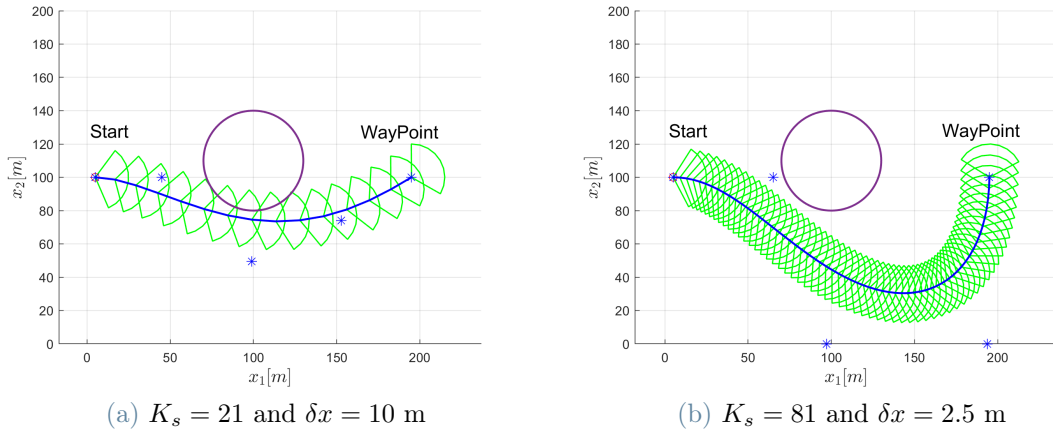


Figure 5.13: Target detection Scenario 3 trajectory design with $t_f = 30$ s and variable K_s

As expected, the computational time increases with the number of points. Remember that the points describing the entire area are K_s^2 . For $K_s = 21$, the computational time is 10.2529 s, instead for $K_s = 81$, it reaches 92.0530 s. With the less fine discretisation, Figure 5.13a, it can be seen that the trajectory passes close to the obstacle, not making full use of the FOV. Recall that the maximum sensor distance is $d_{max} = 20$ m and the aperture $FOV^u = 120$ deg. The points that fall within the FOV with a $\delta x = 10$ m are

very reduced. This means that the optimisation does not effectively take into account the limit between obstacles and free zones.

Finally, having fixed a given trajectory, $t_f = 30$ s and $K_s = 41$, an analysis of the target location estimation can be conducted. Two reference situations are considered: the first, where the target is absent in the lower part of the zone, and the second, where a fixed target is present. Note that even if we consider a moving target, the new measurements would not affect the trajectory design. This is because we are still in a global planning context in the work shown so far. The influence of new measures to re-plan the trajectory is the final and long-term aim of the work set out in this thesis.

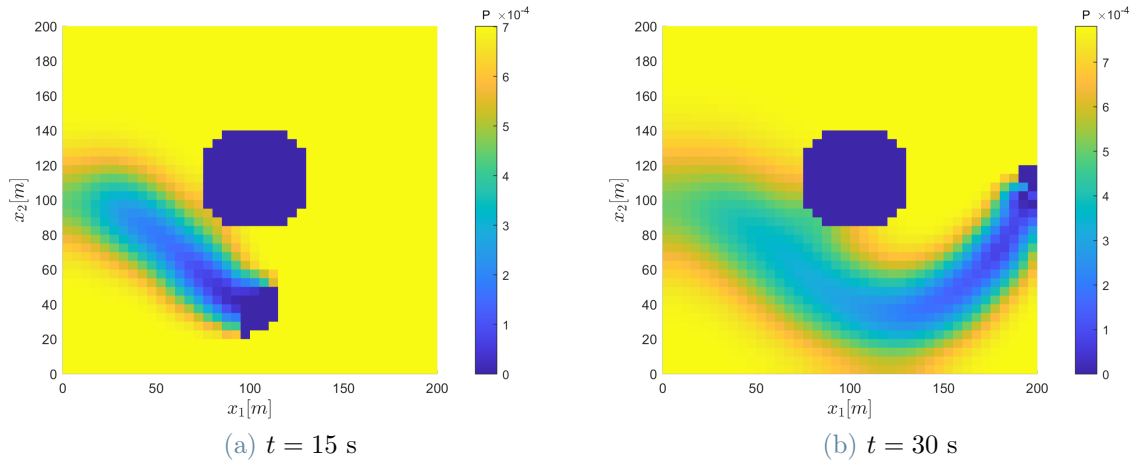


Figure 5.14: Target detection Scenario 3 probability map with ideal updates

In Figure 5.14, the probability map of the first case, without target, is reported for two different time instants. The updates are done with the ideal modelling of the sensor, $p_D = 1$ and $p_F = 0$. Two remarks. Firstly, it is observed that the probability in the unobserved areas increases as non-detection measurements are made. This happens due to normalisation after each measurement update. Secondly, the effect of the moving target hypothesis and therefore the application of propagation in the estimation process causes information to spread over time. If the same area is no longer observed, the probability that had reached almost zero begins to rise. See, for example, the *FOV* near the coordinates (100 40) m in Figure 5.14a and the same area at the final instant in Figure 5.14b.

In the following simulations, a fixed target is added at position (100, 40) m. For this situation, two settings for probabilities are compared. In both cases, p_D is calculated using the instantaneous detection rate function, and p_F is equal to $0.1p_D$ and $0.3p_D$. For each setting, 500 simulations are run. For each simulation, after the trajectory has been

all traversed, the position on the probability map with the highest probability is saved. In Figure 5.15, the results are shown.

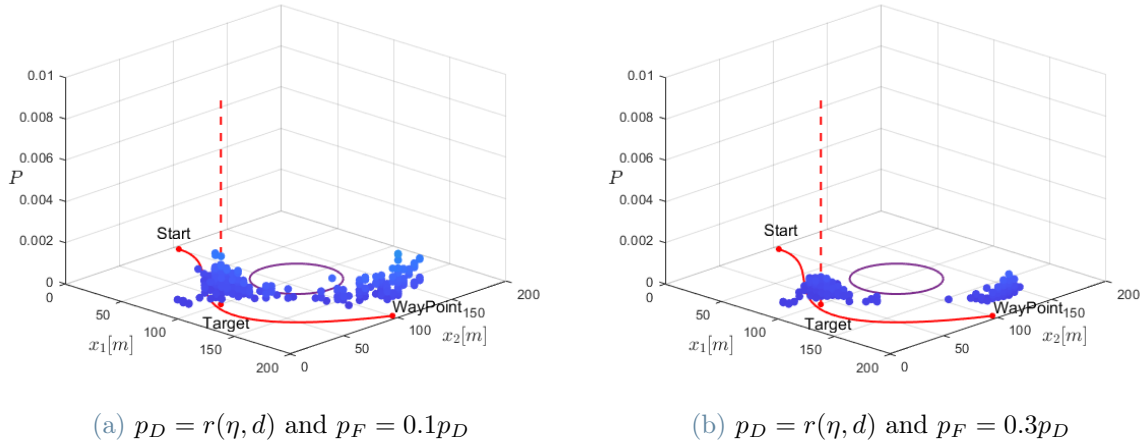


Figure 5.15: Target detection Scenario 3 maximum probability target positions using the instantaneous detection rate function r

Compared to Scenario 2, the maximum probability values are about two orders of magnitude lower. There are two main reasons. On one hand, this is because the initial information is not limited to one sub-area but to the whole zone of interest. Therefore, the probability of a target being present at a specific location is drastically reduced. On the other hand, the information here is propagated.

The maxima are mainly grouped into two groups. One in the vicinity of the true position of the target. The second is around the *FOV* of the last moment of measurements. When the same point is observed several times, the accuracy of measurements increases. For this reason, the latest measurements are more prone to false alarms. It is therefore interesting to note that if we exclude the area of the last measurements, the points around the true target remain as maxima. This behaviour was not observed in scenario 2 because the final part of that trajectory, and consequently the last measurements taken are in an area where nothing is expected to be found, meaning the prior probability is zero.

Free Final Position and Mission Time

For future online implementation and local path planning design, it would be of interest to study the impact of the relaxation of the following two features of the search: final position and mission time.

The first concerns the final position. Instead of imposing that the waypoint must coincide with a predetermined location, it is possible to leave the choice of the waypoint to be

determined by the minimisation of the chosen criterion. Putting `final_cond = 0` specifically removes the third equation of equations 3.10 from the constraints of the problem. Since one must anyway choose an initial guess, the optimisation result will still be influenced by the choice of the waypoint of the initial guess. Take as an example scenario 1 of target detection with $N = 4$, degree elevation $N_{DE} = 40$, and $t_f = 30$ s.

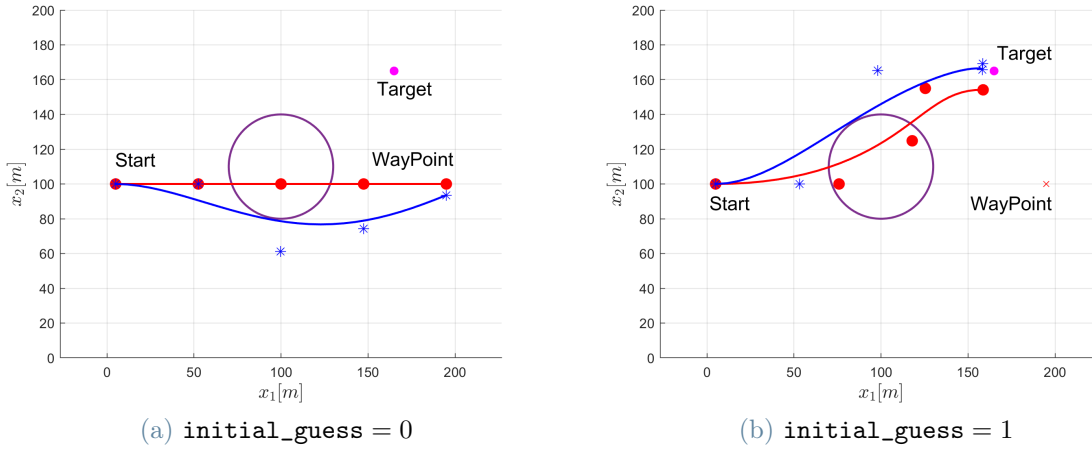


Figure 5.16: Target Detection Scenario 1 without final position constraint

In Figure 5.16 results are reported for `initial_guess` equal both 0 and 1. Since the target is not inside the initial guess cumulative *FOV* for `initial_guess = 0` the trajectory only avoids the obstacle with a final position slightly shifted to the straight line initial guess waypoint, Figure 5.16a. The computational time is 0.5876 s. Instead, when considering as initial guess the result from the optimisation with only detection (*ODO*), the target is reached, Figure 5.16b. In this case, the computational time is much higher. In particular, the one of the second optimisation. C.T. 1 = 0.5697 s and C.T. 2 = 29.9066 s. The second optimisation is stopped not because it has reached a local minimum but because the maximum number of function evaluations has been reached, `MaxFunEva = 4000`. It is not desirable to stop the optimisation for this reason. If the maximum number of evaluations of the function is increased, the result remains almost unchanged and the computational time increases. One reason for this is that the drone tries to slow down more and more to look at the target and minimise the criterion. If one necessarily wants to pass through a particular position, it is better to add a waypoint for simultaneously being sure to pass there and to maintain the computational time within acceptable bounds.

The second possible additional degree of freedom is the mission time. A first home-made way to do this is to insert the optimisation inside a for loop. For each iteration, the same optimisation is carried out (same initial guess) but with an increasingly longer mission

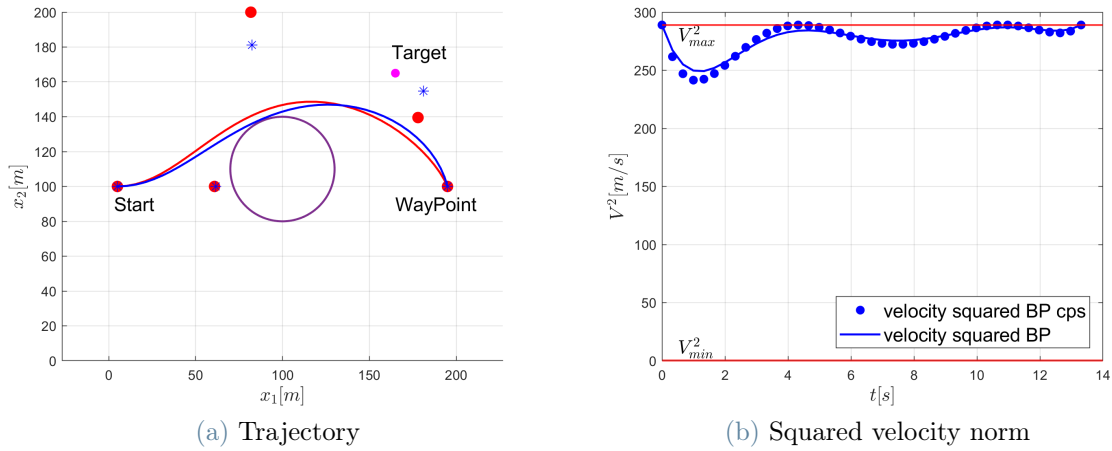


Figure 5.17: Target Detection Scenario 1 with optimisation variable t_f

time. The first t_f for which all constraints are verified is the optimal final time. As the initial t_f , one can start from the minimum time considered to travel the straight line trajectory with the maximum speed. The higher the final time step tested, the greater the number of optimisations. And consequently, the total time. One can think of using this method for offline applications. In particular, to estimate a near-optimal mission time before departure based on a priori knowledge. A more rigorous solution instead proposes to add the variable t_f to the optimisation problem itself and add to the objective function a term related to the mission time to minimise it. Also in this case, the example is taken from scenario 1 of target detection. `initial_guess` is set to 1. The new variable t_f is used in the first optimisation, ODO, but it is not added to the criterion. Instead, in the second optimisation, it is also added as a term in the objective function. The computational times are the following: C.T. 1 = 0.3236 s and C.T. 2 = 6.4424 s. The optimised final time is $\hat{t}_f = 13.3056$ s, Figure 5.17.

5.4. Limitations

The procedure presented so far allows for obtaining a global trajectory for the mission of interest. To transition to online trajectory planning, the goal is to use the same procedure as before, but over a shorter time horizon. At regular intervals, the reference trajectory is recalculated based on the new measurements taken. The measurements may provide information about potential targets or obstacles that were not previously known. As for the junction points between different trajectories, it has already been shown how to impose the boundary conditions. Perhaps the most fundamental aspect of online application is the need for very low computational time for the new reference trajectory. This is because,

ideally, the drone should continue operating without needing to stop and wait for new instructions.

At the moment, this is not the case, particularly for the scenario we are interested in, which involves using the knowledge of the whole probability map. Spatial discretisation affects computation time. However, this is not the limiting factor at the moment. The main reason for this magnitude is that we are dealing with a non-convex problem. The cause of the problem's non-convexity is the presence of obstacles. In Section 6.2, a suggestion to overcome this problem is proposed.

6 | Conclusions

6.1. Conclusions

In this study, we have devised an optimisation framework for UAV trajectory generation in the context of a target search problem. Specifically, a drone and a fixed or mobile target have been considered. The mission area is known, and obstacles are modelled as circles. Neither the drone nor the target can fly over or enter the obstacles. The proposed methodology simultaneously integrates various aspects of the mission. First, it involves the generation of dynamically feasible trajectories that avoid known obstacles. At the same time, to optimise the search mission, the characteristics of the sensor used for target recognition and any information on the target that may be available have been incorporated into the optimisation process. The latter may be derived from prior knowledge or through communication with other UAV fleet members. The method is applicable for generating global, i.e. start-to-end, trajectories but has been designed to be extended and reused in a local re-planning context. For this reason, the final part of the work was dedicated to how information on the target position and, more generally, on the mission area is updated.

All of this has been achieved through two main research streams. The first one concerns the representation of trajectories using Bernstein polynomials and an optimisation method to obtain optimal trajectories with this representation. The way of including constraints on control inputs and obstacle avoidance was especially considered. The criteria that can be optimised include mission time, length, or other relevant metrics. It is within the specific context of the objective function to be optimised that the second research stream comes into play. Indeed, the function considered and reformulated using Bernstein polynomials is the exponential detection model. This model integrates knowledge of the drone's position, potential targets, and the characteristics of the detection instruments. Specifically, minimising this function can be translated into minimising the probability of not detecting a target. Bayesian estimation is used to estimate the target location. By incorporating prior information and the confidence of future measurements, this technique enables an accurate estimation process.

A key advantage of using Bernstein polynomials, as mentioned, is the ability to obtain smooth and dynamically feasible trajectories, contrary to many path-planning methods that produce a set of waypoints that are not directly usable as reference trajectories. This is possible thanks to the convex hull property. In addition, the efficient implementation of certain algorithms that use BP properties allows for effective trajectory description.

A major issue, however, is that the choice of the degree N of the polynomial describing the spatial trajectory and of the order of degree elevation N_{DE} is highly dependent on the environment in terms of obstacles and is not intuitive for an unknown area. This is because, apart from the endpoints, the control points do not lie on the trajectory. In particular, for low-degree BPs (the ones we analysed), the control points are quite distant from the trajectory. Despite this, some guidelines have been provided. The goal is to use these guidelines to build a "brick", in terms of N and N_{DE} , which can then be used on a smaller time and space scale for local trajectory re-planning. For what concerns obstacles, results were analysed only for circular ones. Although it has been shown how polyhedral obstacles can be considered, applying this approach remains very challenging in enclosed or densely obstructed environments.

The presence of obstacles makes the problem non-convex. This has two negative consequences for the optimisation process. Firstly, there is an increased possibility of encountering local minima rather than global minima. Consequently, the obtained solution depends on the choice of the initial guess. This is why the issue of the initial guess trajectory has also been addressed. The second problem is the high computational time. For offline planning, the computation times are generally considered acceptable. However, for an online implementation, much lower values are required. This is particularly important for scenario 3 of target detection, where the entire area is discretised. Although spatial discretisation itself increases computational time, it is in our interest to do it. Minimising the criterion while considering the entire area allows for maximising the observable area with the FOV at various points along the trajectory. Therefore, the trajectory not only avoids the obstacle but also positions itself at a distance such that the obstacle does not obscure part of the sensor's FOV. This characteristic of the proposed method is of great importance.

As for target detection, other considerations can be made. Firstly, the original proposal to perform an initial optimisation considering only the information about the targets was crucial to obtain the resulting performances in terms of target detection. The obtained results show how the trajectory is influenced by prior knowledge of the likely position of a target. At the same time, the modelling of the lack of information, as in scenario 3 of target detection, also allowed us to achieve important results, as we have already

mentioned above. Also regarding the estimation of the target's position, the results shown are promising. The estimation of the position is influenced by the performance of the sensor and the recognition algorithms. This reflects what happens in reality. As for scenario 2 of target detection, the results obtained are positive. The search area has been reduced compared to the initially known one, and the obtained probabilities are always higher than the central maximum of the initial distribution. In scenario 3, we deal with much lower probabilities. This is due to spatial discretisation and the size of the area of interest. However, when considering the order of magnitude relative to the situation, interesting results are obtained. If the most recent measurements are excluded, it has been observed that the estimate of the target's position, obtained after following the entire optimal trajectory previously determined, tends to concentrate around the area where the target was last observed. The latest measurements are more prone to false alarms because the area has been subjected to fewer observations. In this sense, we can also say that the propagation of information filters out recent decoy measurements that are not confirmed by additional measurements.

6.2. Future Works

This thesis should be considered as the foundation and starting point for further research that can be undertaken by others. The following section provides some suggestions for possible developments, which are divided into paragraphs based on specific topics.

Detection Model Refinement The first aspect that can be improved and further explored is the modelling of the sensor and the detection function. It is recommended to start with a specific sensor and its detection algorithms to create a model that accurately represents the recognition process. In addition to relative position, there are other variables involved in the process. For example, the speed and turning rate of the drone significantly affect the ability to recognise a target. Indeed, it can generally be said that an increase in the relative speed between the drone and the object to be observed degrades performance. Regarding the probability of false alarms p_F , only some assumptions about the order of magnitude have been made in this work. It is therefore advisable to further investigate this aspect to obtain more accurate estimates on the target localisation.

Initial Guess Choice As we have seen, the choice of the initial guess significantly impacts the optimisation process. This is both because it affects the optimality of the result and because it influences the computational time required. In the work done, the focus was set on studying an initial guess that prioritises detection over obstacle avoidance.

This is advantageous for target detection, but it does not solve the more computationally expensive problem of obstacle avoidance. To introduce this aspect, the use of the *RRT* method was proposed and briefly analysed to obtain a more "coarse" initial solution. This solution is later optimised using the proposed method. No significant results were achieved. It could be due to the *RRT* implementation used and/or the post-processing procedure to obtain the trajectory. It is therefore proposed that a method that provides a quick and coarse solution in terms of both detection and obstacle avoidance be found. Once this trajectory is obtained, better post-processing should be applied to rewrite it using Bernstein polynomials, and finally, it should be used as the initial guess for the actual optimisation.

Online Path Planning The long-term goal of the proposed work is to develop a framework for the online generation of optimal trajectories influenced by the continuous updating of information about the area of interest. To achieve this, it is proposed to use the method presented and analysed in this thesis over a more limited temporal and, consequently, spatial horizon. At the end of this reduced time interval, a new trajectory is generated that meets the final conditions of the current position of the drone. To accomplish this, the time required to calculate the new trajectory must be negligible compared to the characteristic times of the drone's dynamics. To reduce computational times, it is proposed to make these subproblems convex. Since smaller areas are considered, it is worth dividing the space into convex map sections. In this way, it is expected to achieve better performance. Dividing into convex sub-zones is not a trivial problem, especially in obstacle-dense and not fully known environments. It is recommended to start with known environments and to use existing research in this field.

Multi-agent and 3D Extensions The first possible and interesting extension is to add more UAVs and plan the trajectories of the entire fleet. Regarding obstacle avoidance alone, this has already been addressed in various studies using Bernstein polynomials. At least one of the following constraints must be added to avoid collisions between UAVs: *spatial separation*, which requires that the trajectories of the various UAVs do not intersect at any point in time; or *temporal separation*, which is less stringent and ensures that multiple UAVs do not occupy the same point in space simultaneously. Similarly, the objective function can also be extended for missions involving multiple agents, and there are already some existing studies on this topic. An important factor that will need to be considered is the communication of information. Therefore, the type of fleet organisation must be chosen: centralised or distributed. It is advisable to start with global path planning. Then, move on to local path planning with a single time horizon for all members,

and only later explore the use of different horizons based on event-triggered events.

A final, more general extension concerns the geometric dimension of the problem. So far, the problem has been simplified to two dimensions. Moving to three dimensions significantly increases complexity from both a modelling and computational perspective.

Bibliography

- [1] A. S. Abdel-Rahman, S. Zahran, B. E. Elnaghi, and S. Nafea. Enhanced hybrid path planning algorithm based on apf and a-star. *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, 48:867–873, 2023.
- [2] A. Ait Saadi, A. Soukane, Y. Meraihi, A. Benmessaoud Gabis, S. Mirjalili, and A. Ramdane-Cherif. Uav path planning using optimization approaches: A survey. *Archives of Computational Methods in Engineering*, 29(6):4233–4284, 2022.
- [3] O. Arslan and P. Tsiotras. Use of relaxation methods in sampling-based algorithms for optimal motion planning. In *2013 IEEE International Conference on Robotics and Automation*, pages 2421–2428. IEEE, 2013.
- [4] S. Bertrand, J. Marzat, H. Piet-Lahanier, A. Kahn, and Y. Rochefort. MPC Strategies for Cooperative Guidance of Autonomous Vehicles. *Aerospace Lab*, (8):1–18, 2014. doi: 10.12762/2014.AL08-11. URL <https://hal.science/hal-01103195>.
- [5] L. F. Bertuccelli and J. P. How. Robust uav search for environments with imprecise probability maps. In *Proceedings of the 44th IEEE Conference on Decision and Control*, pages 5680–5685. IEEE, 2005.
- [6] L. F. Bertuccelli and J. P. How. Search for dynamic targets with uncertain probability maps. In *2006 American Control Conference*, pages 6–pp. IEEE, 2006.
- [7] Y.-b. Chen, G.-c. Luo, Y.-s. Mei, J.-q. Yu, and X.-l. Su. Uav path planning using artificial potential field method updated by optimal control theory. *International Journal of Systems Science*, 47(6):1407–1420, 2016.
- [8] V. Cichella, I. Kaminer, C. Walton, and N. Hovakimyan. Optimal motion planning for differentially flat systems using bernstein approximation. *IEEE Control Systems Letters*, 2(1):181–186, 2017.
- [9] V. Cichella, I. Kaminer, C. Walton, N. Hovakimyan, and A. Pascoal. Bernstein approximation of optimal control problems. *arXiv preprint arXiv:1812.06132*, 2018.

- [10] V. Cichella, I. Kaminer, C. Walton, N. Hovakimyan, and A. M. Pascoal. Optimal multivehicle motion planning using bernstein approximants. *IEEE transactions on automatic control*, 66(4):1453–1467, 2020.
- [11] E. W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1(1):269–271, 1959. ISSN 0945-3245. URL <https://doi.org/10.1007/BF01386390>.
- [12] *DJI Matrice 100 User Manual*. DJI, Shenzhen, China, v1.6 edition, 2017. URL <https://www.dji.com/uk/downloads/products/matrice100>.
- [13] *DJI Matrice 300 User Manual*. DJI, Shenzhen, China, v4.0 edition, 2023. URL <https://enterprise.dji.com/matrice-300/downloads>.
- [14] G. Farin. Tighter convex hulls for rational bézier curves. *Computer aided geometric design*, 10(2):123–125, 1993.
- [15] R. T. Farouki. The bernstein polynomial basis: A centennial retrospective. *Computer Aided Geometric Design*, 29(6):379–419, 2012.
- [16] D. Ferguson and A. Stentz. Using interpolation to improve path planning: The field d* algorithm. *Journal of Field Robotics*, 23(2):79–101, 2006.
- [17] J. Foraker, J. O. Royset, and I. Kaminer. Search-trajectory optimization: Part i, formulation and theory. *Journal of Optimization Theory and Applications*, 169:530–549, 2016.
- [18] J. Foraker, J. O. Royset, and I. Kaminer. Search-trajectory optimization: Part ii, algorithms and computations. *Journal of Optimization Theory and Applications*, 169: 550–567, 2016.
- [19] J. Frost and L. Stone. Review of search theory: Advances and applications to search and rescue decision support. us coast guard research and development center rep. Technical report, CG-D-15-01, 2001.
- [20] E. G. Gilbert, D. W. Johnson, and S. S. Keerthi. A fast procedure for computing the distance between complex objects in three-dimensional space. *IEEE Journal on Robotics and Automation*, 4(2):193–203, 1988.
- [21] M. I. Grigor’ev, V. N. Malozemov, and A. Sergeev. Bernstein polynomials and composite bézier curves. *Computational Mathematics and Mathematical Physics*, 46:1872–1881, 2006.
- [22] O. Hellman. On the effect of a search upon the probability distribution of a target

- whose motion is a diffusion process. *The Annals of Mathematical Statistics*, 41(5): 1717–1724, 1970.
- [23] J. Hu, L. Xie, K.-Y. Lum, and J. Xu. Multiagent information fusion and cooperative control in target search. *IEEE Transactions on Control Systems Technology*, 21(4): 1223–1235, 2012.
- [24] C. Huang and J. Fei. Uav path planning based on particle swarm optimization with global best path competition. *International Journal of Pattern Recognition and Artificial Intelligence*, 32(06):1859008, 2018.
- [25] J. Ibenthal. *Multi-target tracking by non-linear set-membership methods*. PhD thesis, Université Paris-Saclay, 2022.
- [26] J. Ibenthal, M. Kieffer, L. Meyer, H. Piet-Lahanier, and S. Reynaud. Bounded-error target localization and tracking using a fleet of uavs. *Automatica*, 132:109809, 2021. ISSN 0005-1098. doi: <https://doi.org/10.1016/j.automatica.2021.109809>. URL <https://www.sciencedirect.com/science/article/pii/S0005109821003290>.
- [27] M. Jun and R. D’Andrea. Probability map building of uncertain dynamic environments with indistinguishable obstacles. In *Proceedings of the 2003 American Control Conference, 2003.*, volume 4, pages 3417–3422. IEEE, 2003.
- [28] S. Karaman and E. Frazzoli. Sampling-based algorithms for optimal motion planning. *The international journal of robotics research*, 30(7):846–894, 2011.
- [29] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. H. Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE transactions on Robotics and Automation*, 12(4):566–580, 1996.
- [30] O. Khatib. Real-time obstacle avoidance for manipulators and mobile robots. *The international journal of robotics research*, 5(1):90–98, 1986.
- [31] C. Kielas-Jensen and V. Cichella. Bebot: Bernstein polynomial toolkit for trajectory generation. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3288–3293. IEEE, 2019.
- [32] C. Kielas-Jensen, V. Cichella, D. Casbeer, S. G. Manyam, and I. Weintraub. Persistent monitoring by multiple unmanned aerial vehicles using bernstein polynomials. *Journal of Optimization Theory and Applications*, 191(2):899–916, 2021.
- [33] C. Kielas-Jensen, V. Cichella, T. Berry, I. Kaminer, C. Walton, and A. Pascoal. Bern-

- stein polynomial-based method for solving optimal trajectory generation problems. *Sensors*, 22(5):1869, 2022.
- [34] S. Koenig and M. Likhachev. D* lite. In *Eighteenth national conference on Artificial intelligence*, pages 476–483, 2002.
- [35] S. Koenig, M. Likhachev, and D. Furcy. Lifelong planning a*. *Artificial Intelligence*, 155(1-2):93–146, 2004.
- [36] B. Koopman. Search and screening. operations evaluation group, office of the chief of naval operations, washington dc. Technical report, OEG Report, 1946.
- [37] S. Kragelund, C. Walton, I. Kaminer, and V. Dobrokhodov. Generalized optimal control for autonomous mine countermeasures missions. *IEEE Journal of Oceanic Engineering*, 46(2):466–496, 2020.
- [38] J. J. Kuffner and S. M. LaValle. Rrt-connect: An efficient approach to single-query path planning. In *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065)*, volume 2, pages 995–1001. IEEE, 2000.
- [39] S. LaValle. Rapidly-exploring random trees: A new tool for path planning. *Research Report 9811*, 1998.
- [40] B.-G. Lee and Y. Park. Distance for bézier curves and degree reduction. *Bulletin of the Australian Mathematical Society*, 56(3):507–515, 1997.
- [41] M. Lyu, Y. Zhao, C. Huang, and H. Huang. Unmanned aerial vehicles for search and rescue: A survey. *Remote Sensing*, 15(13):3266, 2023.
- [42] T. Millet, D. Casbeer, T. Mercker, and J. Bishop. Multi-agent decentralized search of a probability map with communication constraints. In *AIAA Guidance, Navigation, and Control Conference*, page 8424, 2010.
- [43] V. Narayanan, M. Phillips, and M. Likhachev. Anytime safe interval path planning for dynamic environments. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4708–4715. IEEE, 2012.
- [44] M. Otte and E. Frazzoli. Rrtx: Asymptotically optimal single-query sampling-based motion planning with quick replanning. *The International Journal of Robotics Research*, 35(7):797–822, 2016.
- [45] R. Pepy, M. Kieffer, and E. Walter. Reliable robust path planning with application to

- mobile robots. *International Journal of Applied Mathematics and Computer Science*, 2009.
- [46] C. Phelps, Q. Gong, J. O. Royset, C. Walton, and I. Kaminer. Consistent approximation of a nonlinear optimal control problem with uncertain parameters. *Automatica*, 50(12):2987–2997, 2014.
- [47] E. Polak. Algorithms and consistent approximations. *Optimization, Applied Mathematical Sciences*, 124, 1997.
- [48] G. Ramalingam and T. Reps. An incremental algorithm for a generalization of the shortest-path problem. *Journal of Algorithms*, 21(2):267–305, 1996.
- [49] L. Reboul, M. Kieffer, H. Piet-Lahanier, and S. Reynaud. Cooperative guidance of a fleet of uavs for multi-target discovery and tracking in presence of obstacles using a set membership approach. *IFAC-PapersOnLine*, 52(12):340–345, 2019. ISSN 2405-8963. doi: <https://doi.org/10.1016/j.ifacol.2019.11.266>. URL <https://www.sciencedirect.com/science/article/pii/S2405896319312182>. 21st IFAC Symposium on Automatic Control in Aerospace ACA 2019.
- [50] L. Roditty and U. Zwick. On dynamic shortest paths problems. In *Algorithms–ESA 2004: 12th Annual European Symposium, Bergen, Norway, September 14–17, 2004. Proceedings 12*, pages 580–591. Springer, 2004.
- [51] I. M. Ross and M. Karpenko. A review of pseudospectral optimal control: From theory to flight. *Annual Reviews in Control*, 36(2):182–197, 2012.
- [52] S. J. Russell and P. Norvig. *Artificial intelligence: a modern approach*. Pearson, 2016.
- [53] N. Sathvik and S. Patil. Performance analysis of dijkstra’s and the a-star algorithm on an obstacle map. In *Data Engineering and Intelligent Computing: Proceedings of ICICC 2020*, pages 77–86. Springer, 2021.
- [54] A. Schwartz and E. Polak. Consistent approximations for optimal control problems based on runge–kutta integration. *SIAM Journal on Control and optimization*, 34(4):1235–1269, 1996.
- [55] F. Schweppe. Recursive state estimation: Unknown but bounded errors and system inputs. *IEEE Transactions on Automatic Control*, 13(1):22–28, 1968.
- [56] G. Sharon, R. Stern, M. Goldenberg, and A. Felner. The increasing cost tree search for optimal multi-agent pathfinding. *Artificial intelligence*, 195:470–495, 2013.

- [57] G. Sharon, R. Stern, A. Felner, and N. R. Sturtevant. Conflict-based search for optimal multi-agent pathfinding. *Artificial intelligence*, 219:40–66, 2015.
- [58] D. Silver. Cooperative pathfinding. In *Proceedings of the aaai conference on artificial intelligence and interactive digital entertainment*, volume 1, pages 117–122, 2005.
- [59] H. Sira-Ramirez and S. K. Agrawal. *Differentially flat systems*. Crc Press, 2018.
- [60] A. Stentz et al. The focussed d* algorithm for real-time replanning. In *IJCAI*, volume 95, pages 1652–1659, 1995.
- [61] P. Sujit and R. Beard. Multiple uav path planning using anytime algorithms. In *2009 American control conference*, pages 2978–2983. IEEE, 2009.
- [62] L. SZ DJI Technology Co. Support for matrice 100. <https://www.dji.com/it/support/product/matrice100>, .
- [63] L. SZ DJI Technology Co. Matrice 300 rtk. <https://enterprise.dji.com/it/matrice-300>, .
- [64] C. Tabasso and V. Cichella. On-line Motion Planning Using Bernstein Polynomials for Enhanced Target Localization in Autonomous Vehicles. *arXiv e-prints*, art. arXiv:2210.03187, Oct. 2022. doi: 10.48550/arXiv.2210.03187.
- [65] C. L. Walton, Q. Gong, I. Kaminer, and J. O. Royset. Optimal motion planning for searching for uncertain targets. *IFAC Proceedings Volumes*, 47(3):8977–8982, 2014.
- [66] L. Zhou and W. Li. Adaptive artificial potential field approach for obstacle avoidance path planning. In *2014 Seventh International Symposium on Computational Intelligence and Design*, volume 2, pages 429–432. IEEE, 2014.

A | Appendix A

General formula of the Probability Density Function of a Beta distribution with variable $x \in [0, 1]$:

$$\text{prob}(x | \alpha, \beta) = \frac{x^{\alpha-1} (1-x)^{\beta-1}}{\int_0^1 t^{\alpha-1} (1-t)^{\beta-1} dt} \quad (\text{A.1})$$

Here below it is reported the possibility of modelling scenario 2 describing Ω as the entire zone of interest. Some considerations need to be made. Firstly, to consider a general situation where the peak of the Beta distribution is not centred, it is necessary to refer to the definition of the mode of this type of distribution. The following formula is valid only for both α and β greater than 1.

$$\text{mode} = \frac{\alpha - 1}{\alpha + \beta - 1}. \quad (\text{A.2})$$

The value of alpha is fixed a priori. Then, starting from the desired coordinates for the maximum of the distribution, beta is calculated using Equation A.2. This is done for both spatial directions. Once the distribution is obtained, two additional steps are necessary. Firstly, the points where obstacles are present must be set to zero. Then, assuming the presence of a target, the values in the Probability Map are normalised so that their sum equals 1. In Figure ?? two examples are reported. For both situations, the maximum probability is fixed for the position (165 165) m. In Figure ?? $\alpha_x = \alpha_y = 45$, instead in Figure ?? $\alpha_x = \alpha_y = 10$.

It is observed that by decreasing the value of alpha, the distribution becomes less concentrated around the peak. As a result, the probability values decrease. Note that the scales of the probabilities in the two figures are different.

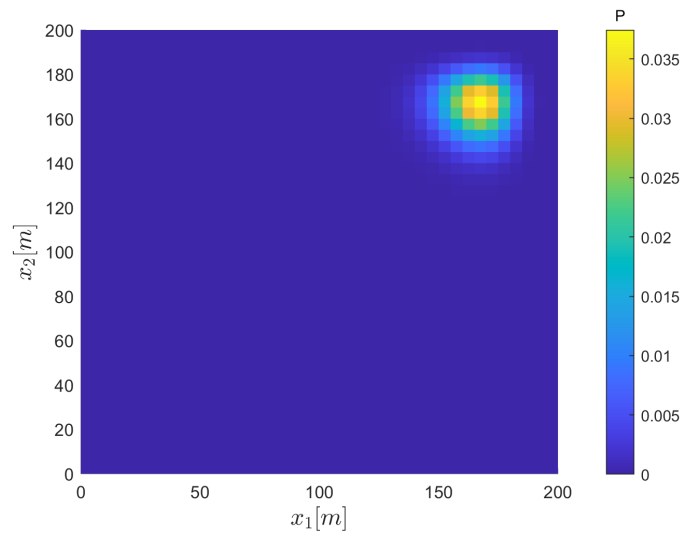
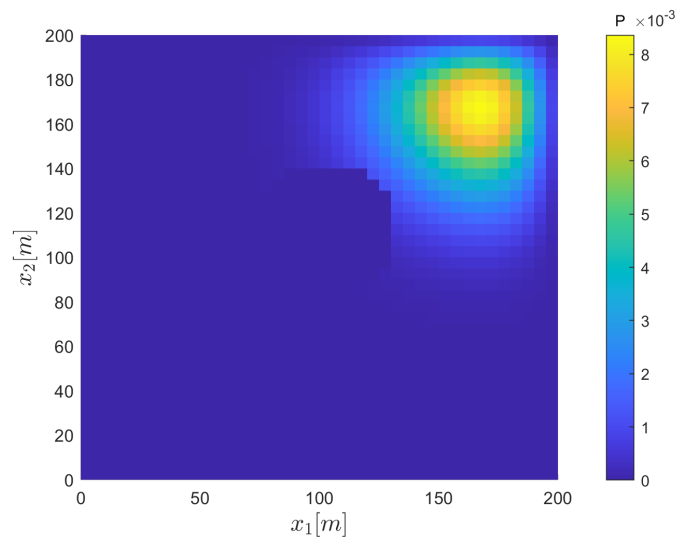
(a) $\alpha_x = \alpha_y = 45$ (b) $\alpha_x = \alpha_y = 10$

Figure A.1: Partially Known Fixed Target Probability Map using the entire zone of interest

B | Appendix B

Obstacles can also be described using convex polygons. Although initially used in this work, this approach was later set aside for the reasons already mentioned in Section 3.3. However, we have decided to include a description of this method. Specifically, in an early stage of the work, an attempt was made to introduce obstacle avoidance within the cost function, and this was done using the polygonal description of the obstacles.

Polygons are defined by an array containing the coordinates of the vertices. The vertices are ordered. In our case, they are arranged in a counterclockwise direction, but they could also be in a clockwise direction; the important thing is to be aware of this for certain details in the implementation of some functions. Furthermore, in the array of coordinates used to describe the obstacle, the initial vertex is repeated at the end to ensure the polygon is properly closed.

Below are the three proposed options for incorporating obstacle avoidance into the objective function to be minimised.

Distance function The first proposition considers a penalising function only if the trajectory is colliding with an obstacle. The penalisation term is a function of the trajectory and the obstacle geometry. For instance, we do not consider a safety distance between trajectory and obstacles. So the cost function can be written as:

$$J_{coll}(\mathbf{P}_{i,N}, \mathbb{O}_m) = \int_0^{t_f} f(\mathbf{x}_N^u(\mathbf{P}_{i,N}^u, t), \mathbb{O}_m) \cdot I_{\mathbb{O}(\mathbf{x}_N^u)}(\mathbf{x}_N^u(\mathbf{P}_{i,N}^u, t), \mathbb{O}_m) dt, \quad (\text{B.1})$$

where \mathbb{O}_m is the set of known obstacles in the zone of interest and

$$I_{\mathbb{O}}(\mathbf{x}_N^u(\mathbf{P}_{i,N}^u, t), \mathbb{O}_m) = \begin{cases} 1 & \text{if } \mathbf{x}_N^u \in \mathbb{O}_m \\ 0 & \text{otherwise} \end{cases} \quad (\text{B.2})$$

The function f is inversely proportional to the distance from the curve to the centre of the obstacle. It is not trivial to know if $\mathbf{x}_N^u \in \mathbb{O}_m$. Alternatively, instead of using $I_{\mathbb{O}}$

one can use an algorithm that computes if an intersection occurs between a BP and a polygonal obstacle. It exits a boolean value. So we do not know the time instants of the possible intersections and consequently, the sub-part of $\mathbf{x}_N^u(\mathbf{P}_{i,N}^u, t)$ that belongs to \mathbb{O}_m . If the above cost function is discretised with respect to the time we obtain:

$$J_{coll}(\mathbf{P}_{i,N}, \mathbb{O}_m) = \frac{t_f}{K-1} \sum_{k=0}^{K-1} f(\mathbf{x}_N^u(\mathbf{P}_{i,N}^u, t_k), \mathbb{O}_m) \cdot I_{\mathbb{O}(\mathbf{x}_N^u)}(\mathbf{x}_N^u(\mathbf{P}_{i,N}^u, t_k), \mathbb{O}_m), \quad (\text{B.3})$$

where $t_k = \frac{k}{(K-1)}t_f$. In the discretised case, the computation of $I_{\mathbb{O}}$ becomes simply a condition on each discretised point. If the point is inside any obstacles the value is 1, otherwise is 0. However, we are checking only a finite number of points, so we cannot be completely sure of avoidance even if J_{coll} becomes 0.

Area function The second proposition concerns the idea of areas intersection. In particular, the intersection between obstacles and an area (*prediction area*) that, for each point in the trajectory, describes the set of possible directions and positions. The dimensions are dictated by the UAV dynamics ($\omega_{max}, V_{max}, V_{min}$) and the time discretisation K chosen for the evaluation of J . Considering the above-mentioned characteristics the matrices of the “prediction” area, $\mathbb{A}_p(s)$, are computed, $A_p(\mathbf{x}_N^u)$ and $b_p(\mathbf{x}_N^u)$. The matrices A_p and b_p characterise the linear system that distinguishes between the inner and outer regions of the "linearised" FOV, see Figure B.1. In this case, a polygon with 4 sides was used to linearise the FOV. In general, a higher number of vertices can be chosen to improve accuracy.

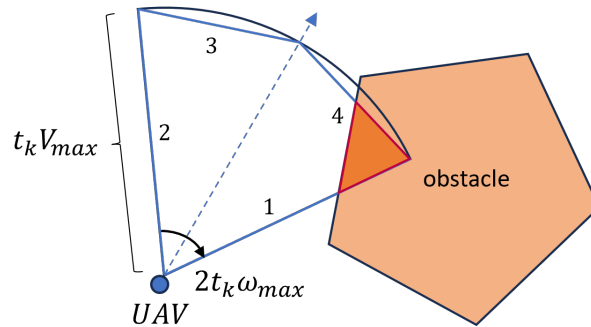


Figure B.1: *Prediction area representation*

The goal is to minimise the overlap between the obstacle area and the potential future movements of the drone. This is a conservative choice.

$$J_{coll}(\mathbf{P}_{i,N}, \mathbb{O}_m) = \int_0^{t_f} \int_{\mathbb{A}_p(\mathbf{x}_N^u) \cap \mathbb{O}_m} dA dt \quad (\text{B.4})$$

In the discrete case, the function is reduced to a finite summation of intersections calculated at a certain number of positions along the trajectory.

Angle function The idea of this proposition is to use the field of view as the only source of information for obstacle avoidance. In this sense, the criterion could be extended to unknown obstacles. The cost function is:

$$J_{coll}(\mathbf{P}_{i,N}, \mathbb{O}_m) = \int_0^{t_f} \int_{\beta_1}^{\beta_2} g(\alpha(\mathbf{x}_N^u), \mathbb{O}_m) d\alpha \cdot I_{\mathbb{O}_{\mathbb{A}_d}(\mathbf{x}_N^u)}(\mathbf{x}_N^u, \mathbb{O}_m) dt, \quad (\text{B.5})$$

where

$$I_{\mathbb{O}_{\mathbb{A}_d}}(\mathbf{x}_N^u(\mathbf{P}_{i,N}^u, t), \mathbb{O}_m) = \begin{cases} 1 & \text{if } \mathbb{A}_d(\mathbf{x}_N^u) \cap \mathbb{O}_m \neq \emptyset \\ 0 & \text{otherwise} \end{cases} \quad (\text{B.6})$$

g is a bell-shaped function constructed with support $[-\frac{FoV}{2}, +\frac{FoV}{2}]$ and max value equal to 1. If $\mathbb{A}_d(s) \cap \mathbb{O}_m \neq \emptyset$ from the intersection points we obtain two angles, β_1 and β_2 that are in $[-\frac{FoV}{2}, +\frac{FoV}{2}]$, see Figure B.2.

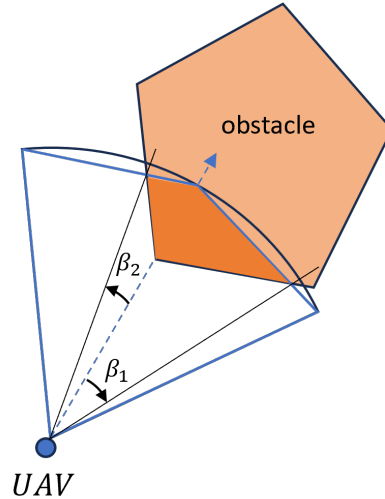


Figure B.2: Representation of *Angle* function situation

The idea is to "move away" the direction of the FOV, and thus the trajectory, from the direction in which the drone is heading if it implies a potential collision with an obstacle.

This adjustment is achieved by using the gradient of the area between β_1 and β_2 under the bell-shaped function.

Considerations The polygonal description allows for simple functions to calculate intersections and thus the functions mentioned above. However, there are two reasons why we decided to abandon this description. The first reason, as mentioned above, is that it does not allow for an efficient solution to the optimisation problem due to the non-differentiable function that calculates the distance between the trajectory and the polygon. Secondly, including obstacle avoidance within the cost function does not guarantee avoidance, which is crucial for the safety of the mission.

C | Appendix C

```

1 function [S] = StateTransitionMatrix(CONSTANTS)
2
3 % StateTransitionMatrix -
4 % Given a specific area of interest in terms of dimensions,
   spatial discretisation, and circular obstacles, the
   StateTransitionMatrix function calculates the matrix that
   describes the Markovian diffusion process. The assumptions
   about the target's diffusion are that it can move to the 8
   adjacent nodes or remain stationary in its current position.
5
6 % Inputs:
7 %   CONSTANTS is a structure that contains most of the information
   used to model the optimal trajectory generation problem. Below
   are listed only the fields that are used within the function.
8
9 %   .Ks (integer) - number of discretised points along one
   direction
10 %   .X (matrix) - grid of position along the first direction
11 %   .Y (matrix) - grid of position along the second direction
12 %   .nbObstacles (integer) - number of obstacles present in the
   zone
13 %   .obstacle (cell) - cell of dimension [nbObstacles x 1]
   containing the characteristics of each circular obstacle
14 %   .obstacle.radius (double) - radius of the obstacle
15 %   .obstacle.centre (array) - x and y positions of centre
16
17 % Outputs:
18 %   S: sparse matrix representing the state transition matrix
19
20 % Author: Giovanni Boscolo Boscoletto
21 % Date: 16-08-2024
22

```

```

23 Env = ones(CONSTANTS.Ks);
24 if CONSTANTS.nbObstacles ~= 0
25     for ii = 1 : size(CONSTANTS.X,1)
26         for jj = 1 : size(CONSTANTS.Y,2)
27             for kk = 1 : CONSTANTS.nbObstacles
28                 if (CONSTANTS.X(ii,jj) - CONSTANTS.obstacle{kk,1}.
29                     center(1))^2 + ...
30                     (CONSTANTS.Y(ii,jj) - CONSTANTS.obstacle{kk
31                         ,1}.center(2))^2 < ...
32                         CONSTANTS.obstacle{kk,1}.radius^2
33                         Env(ii,jj) = 0;
34                     end
35                 end
36             end
37         end
38     end
39 end
40
41 B = zeros(CONSTANTS.Ks+2);
42 B(2:end-1,2:end-1) = Env;
43
44 % Now we construct the state transition matrix A of dimension Ks^2
45 % x Ks^2
46
47 A = zeros(CONSTANTS.Ks^2);
48
49 % for loop on B without frame
50 for ii = 2 : CONSTANTS.Ks+1
51     for jj = 2 : CONSTANTS.Ks+1
52         norm_ij = sum(sum(B(ii-1:ii+1,jj-1:jj+1)));
53         for kk = ii-1 : ii+1
54             for ll = jj-1 : jj+1
55                 % the following condition is to not consider the
56                 % frame of zero in the construction of the state
57                 % transition matrix
58                 if kk ~= 1 && kk ~= CONSTANTS.Ks+2 && ll ~= 1 &&
59                     ll ~= CONSTANTS.Ks+2
60                     % the following condition is to not make move
61                     % a point inside an obstacle
62                     if B(kk,ll) ~= 0

```

```

55         A((ii-2)*CONSTANTS.Ks+(jj-1),(kk-2)*
56             CONSTANTS.Ks+(ll-1)) = 1/norm_ij;
57     end
58 end
59 end
60 end
61 end
62
63 % Since no points inside the obstacles are reachable by the UAVs
64 % we have to put to zero all the state transition matrix lines
65 % that correspond to an obstacle point.
66
67 if CONSTANTS.nbObstacles ~= 0
68     for ii = 1 : CONSTANTS.Ks
69         for jj = 1 : CONSTANTS.Ks
70             for kk = 1 : CONSTANTS.nbObstacles
71                 if (CONSTANTS.X(ii,jj) - CONSTANTS.obstacle{kk,1}.
72                     center(1))^2 + ...
73                     (CONSTANTS.Y(ii,jj) - CONSTANTS.obstacle{kk
74                         ,1}.center(2))^2 < ...
75                     CONSTANTS.obstacle{kk,1}.radius^2
76                     A((ii-1)*CONSTANTS.Ks+jj,:) = zeros(1,
77                         CONSTANTS.Ks^2);
78                 end
79             end
80         end
81     end
82 end
83
84 % to reduce the memory dimension A is saved as a sparse matrix
85
86 S = sparse(A);
87 end

```


D | Appendix D

D.1. BP initial guess with different degrees

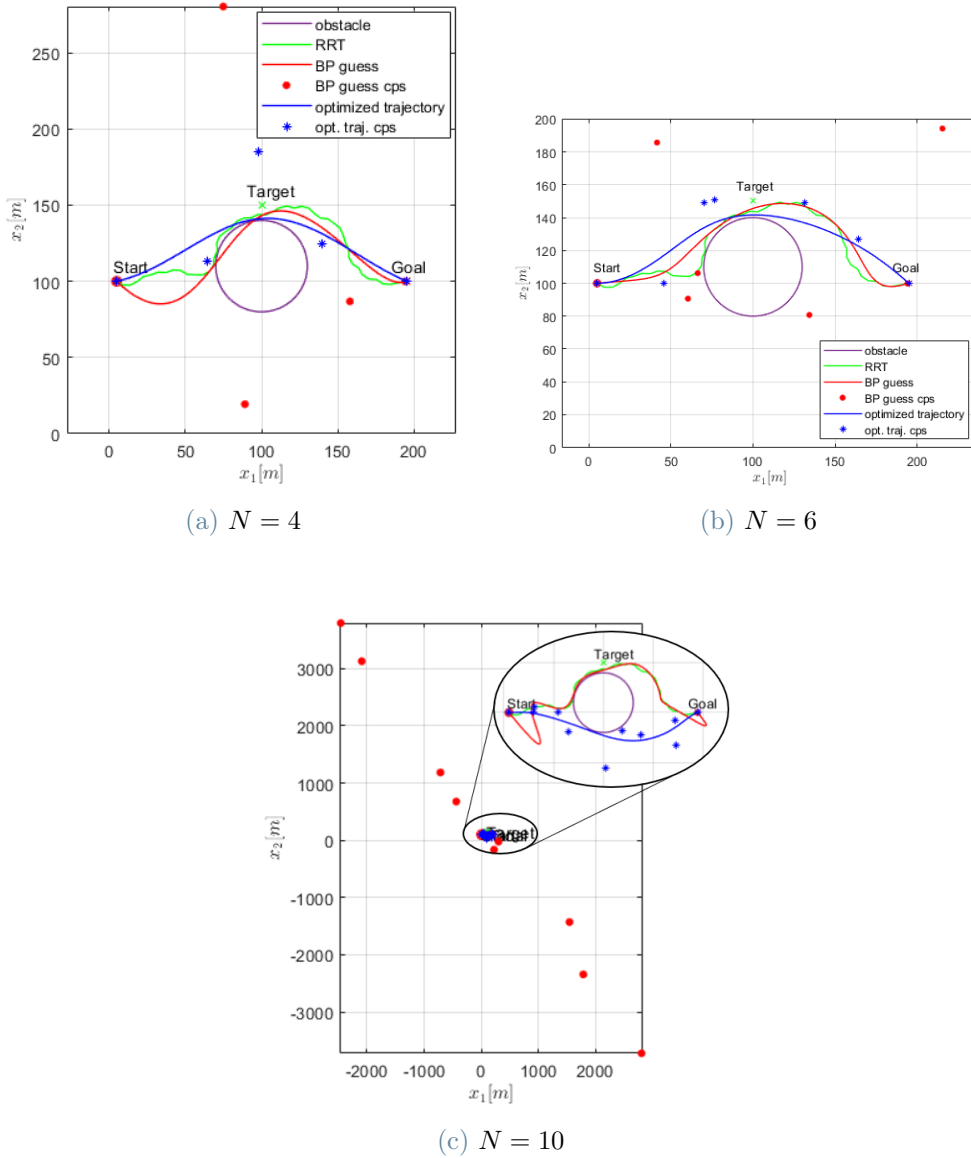


Figure D.1: BP initial guess with different N from RRT planning

D.2. System inversion code

```

1 function [P] = traj2BPcps(traj,t,N)
2
3 % traj2BPcps -
4 % Given a discrete trajectory (points in space and time instances)
   and the chosen degree of the desired Bernstein polynomial, the
   traj2BPcps function calculates the position of the control
   points that describe a Bernstein polynomial passing through (N
   - 1) equidistant points of the discrete trajectory, excluding
   the initial and final points.
5
6 % Inputs:
7 %   traj (matrix) - matrix containing the coordinates of discrete
   points describing a trajectory. The number of rows is equal to
   the dimension of the problem and the number of columns depends
   on the number of points that describe the traj.
8 %   t (matrix) - vector containing the time instants of the
   trajectory
9 %   N (integer) - degree of the desired Bernstein polynomial (N
   +1=#cps)
10
11 % Outputs:
12 %   P : (matrix) matrix containing the BP control points
13
14 % Author: Giovanni Boscolo Boscoletto
15 % Date:   16-08-2024
16
17 len = size(traj,2);
18
19 if ~mod(len-1,N)
20     step = (len-1)/N;
21     traj = traj(:,1:step:end);
22     t = t(1:step:end);
23 else
24     step = ceil(len/N);
25     traj = [traj(:,1:step:end) traj(:,end)];
26     t = [t(1:step:end) t(:,end)];
27 end

```

```

28
29 tf = t(end);
30
31 dim = size(traj,1);
32
33 X = reshape(traj(:,2:end-1),[],1);
34
35 P0 = repmat(traj(:,1),(N-1),1);
36 Pn = repmat(traj(:,end),(N-1),1);
37
38 binom = zeros(1,N+1);
39
40 for ii = 0 : N
41     bin = 1;
42     for jj = 1:ii
43         bin = bin*(N-(ii-jj));
44         bin = bin/jj;
45     end
46     binom(ii+1) = bin;
47 end
48
49 B0 = repmat(binom,dim,1);
50 B_row = reshape(B0,[],1)';
51 B = repmat(B_row,dim*(N-1),1);
52
53 T0 = repmat(t(2:end-1),dim,1);
54 T_col = reshape(T0,[],1);
55 T = repmat(T_col,1,dim*(N+1));
56
57 I0 = repmat(0:N,dim,1);
58 I_row = reshape(I0,[],1)';
59 I = repmat(I_row,dim*(N-1),1);
60
61 f = @(ii,tt) (tt.^ii).*(tf - tt).^(N-ii) ./ tf^N;
62 F = f(I,T);
63
64 Idiags = repmat(eye(dim),N-1,N+1);
65 Arect = B.*F.*Idiags;
66

```

```
67 A = Arect(:,dim+1:end-dim);
68 d = sum(Arect(:,1:dim),2);
69 e = sum(Arect(:,end-(dim-1):end),2);
70 b = X - d.*P0 - e.*Pn;
71
72 P_int = A \ b;
73 P = [traj(:,1), reshape(P_int,dim,[]), traj(:,end)];
74
75 end
```

List of Figures

2.1	2D BP example	20
2.2	1D BP example	20
2.3	Degree Elevation of 1D BP of Figure 2.2	23
2.4	Example of <i>de Casteljau</i> algorithm for a 2-dimensional, 3^{rd} order BP	25
3.1	DJI Matrice 100 from [62]	27
3.2	DJI Matrice 300 RTX from [63]	27
3.3	<i>FOV</i>	32
3.4	Instantaneous detection rate function $r(\eta, d)$	34
3.5	Probability map for a known fixed target	36
3.6	Partially Known Fixed target	37
3.7	Probability map for unknown target	38
3.8	Minimum distance between a BP and obstacles	40
4.1	Scheme for reproducing randomness in measurements	45
4.2	Example of <i>Env</i> matrix for state transition matrix construction, top-left corner portion. Examples of two <i>Neigh</i> matrices (3×3) with the possible movements of the central cell.	47
4.3	Example of initial guess from <i>RRT</i> method with $N = 6$	51
4.4	Initial guess with ODO	53
5.1	Obstacle avoidance Scenario 1 trajectories with different N	61
5.2	Obstacle avoidance Scenario 1, V^2 and ω	62
5.3	Obstacle avoidance Scenario 2 with $N = 5$, with and without DE	64
5.4	Obstacle avoidance Scenario 2 C.T. for different N and with or without DE	65
5.5	Target detection Scenario 1 without and with DE	66
5.6	Target detection Scenario 1 trajectory with DE and no limits	66
5.7	Target detection Scenario 2 trajectory with DE	67
5.8	Target detection Scenario 2 probability map with ideal updates	68
5.9	Target detection Scenario 2 maximum probability target positions with different settings	69

5.10	Target detection Scenario 2 maximum probability target positions with different p_D and p_F used for correction step and simulation	70
5.11	Target detection Scenario 2 maximum probability target positions using the instantaneous detection rate function r	71
5.12	Target detection Scenario 3 trajectory design with variable t_f	72
5.13	Target detection Scenario 3 trajectory design with $t_f = 30$ s and variable K_s	73
5.14	Target detection Scenario 3 probability map with ideal updates	74
5.15	Target detection Scenario 3 maximum probability target positions using the instantaneous detection rate function r	75
5.16	Target Detection Scenario 1 without final position constraint	76
5.17	Target Detection Scenario 1 with optimisation variable t_f	77
A.1	Partially Known Fixed Target Probability Map using the entire zone of interest	92
B.1	<i>Prediction</i> area representation	94
B.2	Representation of <i>Angle</i> function situation	95
D.1	BP initial guess with different N from <i>RRT</i> planning	101

List of Tables

5.1	DJI Matrice 100 and 300 RTX drone specifications	59
5.2	Simulation Numerical Values	60
5.3	Obstacle avoidance Scenario 1 Computational Time (C.T.) for different N	63
5.4	Target detection Scenario 1 computational times	67

Nomenclature

Acronyms

FOV Field of View: sensor detection area

FOV_{ODO} virtual Field of View used in the Optimisation with Detection Only

*LPA** Lifelong planning *A** algorithm

APF Artificial Potential Field

BeBOT Bernstein/Bézier Optimal Trajectories

BP Bernstein Polynomial

C.T. Computational Time

CBS Conflict-Based Search

cp control points

DCL Detection, Classification, and Localisation

DE Degree Elevation

FIM Fisher Information Matrix

GJK Gilbert-Johnson-Keerth

ICTS Increasing Cost Tree Search

MAPF Multi-Agent Path Finding

MCM Mine Countermeasures Mission

MPC Model Predictive Control

ODO Optimisation with Detection Only

PDF Probability Density Function

PMF Probability Mass Function

PRM	Probabilistic Road Map
PSO	Particle Swarm Optimisation
RID	Re-acquisition and Identification
RRT	Rapidly-exploring Random Tree
SAR	Search and Rescue
SIPP	Safe-Interval Path Planning
UAV	Unmanned Aerial Vehicle

Symbols

α	parameter of Beta distribution
β	parameter of Beta distribution
Ω	spatial domain of the random variable \mathbf{X}^t
\mathbf{c}_{O_i}	centre of the i^{th} circular obstacle
\mathbf{x}_N^u	BP approximant of UAV position
\mathbf{x}^t	target position
\mathbf{x}^u	UAV position
\mathbf{X}^t	target position random variable
δt	time interval to move from one cell to the adjacent one at V_{max}^t
δx	cell side length
Δ	spatial domain of integration of the objective function
$\dot{\mathbf{x}}_0^u$	boundary condition initial velocity
η	angle between UAV's trajectory direction and the direction connecting UAV to the target
\mathbf{D}_N	derivation matrix for a set of BP control points
\mathbf{E}	degree elevation matrix for a set of BP control points
$\mathbf{P}_{i,N}$	i^{th} control point of a N order Bernstein Polynomial
$\mathbf{P}_{i,N}^j$	i^{th} control point of a N order BP of the j^{th} iteration of the <i>de Casteljau</i> algorithm
\mathbf{x}_0^u	boundary condition initial position

\mathbf{x}_f^u	boundary condition final position
ω	UAV turning rate
$\omega_{i,N}$	i^{th} weight of a rational N order Bernstein Polynomial
ω_{max}	maximum UAV turning rate
ψ	UAV heading angle
\mathbf{obs}_{pol}	set of vertices coordinates defining a polygonal obstacle
$\widehat{\mathbf{P}}_{i,N}^u$	BP cps that optimise the problem
A	state transition matrix
d	distance between UAV and target position
d_{max}	maximum distance defining the UAV FOV
d_{min}	minimum distance defining the UAV FOV
E	trajectory safety distance from obstacles
Env	matrix composed of zeros (obstacles) and ones (free zones) describing the environment
$f_{\mathbf{X}^t}$	probability density function of the random variable \mathbf{X}^t
FOV^u	angle amplitude of the UAV FOV
K_s	number of discretised points for each spatial dimension
$K_{t,1}$	number of discretised temporal points used in the optimisation process
$K_{t,2}$	number of discretised temporal points used to diffuse the target information
N	order of Bernstein Polynomial
n	dimension of BP
N_{det}	number of detected targets
N_{DE}	degree used for degree elevation of a BP
N_{free}	number of cells where the target could be located
N_{moves}	number of possible target moves starting from its actual position
N_{obs}	number of obstacles
p_D	probability of detection

p_F	probability of false alarm
p_N	probability of no detection
$P_{i,j,k}$	probability that target i is located in cell j at time instant k
$P_{i,j}$	probability that target i is located in cell j
$P_{j,k}$	probability that a cell j at time instant k contains at least one target
r	instantaneous detection rate
r_{O_i}	radius of the i^{th} circular obstacle
t	general instant of time
t_0	initial time
t_f	final or mission time
t_{div}	time used to evaluate at a precise time instant a BP with the <i>de Casteljau</i> algorithm
V	UAV velocity
V_{max}^t	maximum target velocity
V_{max}	maximum UAV velocity
V_{min}	minimum UAV velocity
x_1	first spatial dimension
x_1^u	UAV position along first dimension
x_2	second spatial dimension
x_2^u	UAV position along second dimension
x_j	position of the cell j
$X_{i,k}$	random variable defining position of target i at time instant k
$Y_{i,j,k}$	random variable of measurement of target i , in cell j , at time instant k
$y_{i,j,k}$	measurement of target i , in cell j , at time instant k
$B_{i,N}$	Bernstein Polynomial basis of order N
C_N	Bernstein Polynomial of order N

Acknowledgements

I would like to express my special thanks to **Hélène Piet-Lahanier** for her support during my research internship at ONERA. I also thank **Michel Kieffer** for his precise and timely contributions. My gratitude goes to my internal supervisor **Davide Invernizzi** for his availability and attentiveness despite the distance.

Un grand merci à mes professeurs de Supaero préférées, **Caroline Berard** et **Caroline Chanel**. Merci pour la passion que vous mettez dans vos domaines et dans votre enseignement. Pour vous, les étudiants sont des compagnons de chemin dont vous vous occupez. Merci parce que ce travail de fin d'études a été possible grâce à l'attention que vous m'avez portée.

Ringrazio **Paola Sala** della segreteria del Polimi perchè ha avuto la disponibilità di risolvere i miei mille problemi burocratici in modo rapido ed efficace.

Ci tengo a terminare NON ringraziando la burocrazia e tutte le persone che incarnano in senso negativo questo termine. Tutte le persone che si limitano a fare al minimo il loro lavoro senza aiutare le persone che hanno davanti. Persone che non fanno un minimo sforzo davanti ad una situazione che esce dal normale rimandando continuamente ad altri.

Per tutti gli altri ringraziamenti vi lascio alle pagine successive.

