Emergency Trajectory Structure for UAVs

Maëva Ongale-Obeyi, Damien Goubinat, Daniel Delahaye, Pierre-Loïc Garoche

aerospace

MDPI

Article

Emergency Trajectory Structure for UAVs

Maëva Ongale-Obeyi [1,*], Damien Goubinat [2,†], Daniel Delahaye [3,†] and Pierre-Loïc Garoche [3,†]

[1] Thales Avionics France, 31100 Toulouse, France
[2] Thales Canada Inc., Montréal, QC H4S 2C2, Canada; damien.goubinat@thalesgroup.com
[3] Ecole Nationale de l'Aviation Civile, Université de Toulouse, 31400 Toulouse, France; delahaye@recherche.enac.fr (D.D.); pierre-loic.garoche@enac.fr (P.-L.G.)
* Correspondence: maeva.ongaleobeyi@fr.thalesgroup.com; Tel.: +33-6-52-38-39-40
† These authors contributed equally to this work.

Abstract: The study of the design of emergency trajectories of air vehicles is one of the key elements in improving airspace safety for air vehicles. The aim is to lighten pilots' workload, offering quick and effective solutions. However, almost all flight optimizers proposed in the literature still need to be completed when it comes to resolving emergency contexts, which presents a significant disadvantage to the advancement of scientific research. This resolution is based on the following problems: (a) finding paths free of obstacles, (b) ensuring their flight capacity, and finally, (c) calculating trajectories optimizing several criteria with a calculation time constraint (a few minutes). This document analyzes the safety landing problem and proposes an architecture that effectively reduces complexity and ensures solvability within a reasonable computational time. This architectural framework is designed to be adaptable, allowing for testing several algorithms to provide a quick overview of their strengths and weaknesses in this context. The primary aim of these tests is to benchmark the computational time of the overall architecture, ensuring that this adaptable framework is fully capable of handling the problem's complexity. It is important to note that the algorithms chosen address only a simplified version of the problem. The initial results are promising in terms of time response and the potential to enhance the representativeness and complexity of the problem. The next phase of our research will focus on striking the right balance between complexity, representativity, and computational time, aiming to impact emergency response significantly.

Keywords: emergency; autonomous decision support framework; multiple trajectory planning

Academic Editor: Gokhan Inalhan

Received: 6 November 2024
Revised: 25 December 2024
Accepted: 26 December 2024
Published: 31 December 2024

Citation: Ongale-Obeyi, M.; Goubinat, D.; Delahaye, D.; Garoche, P.-L. Emergency Trajectory Structure for UAVs. *Aerospace* **2025**, *12*, 21. https://doi.org/10.3390/aerospace12010021

1. Introduction

Aerial vehicle safety is one of the most widely studied topics in aviation. Indeed, the use of aerial vehicles within various disciplines makes them the key players in many sectors, and thus, their system safety is always a priority. Before technical development reaches the point where we can realistically question whether it is necessary to have a pilot on board, we must first focus on a question such as "How can we reduce the crew load as much as possible in all circumstances?". This question is one of many that must be resolved before allowing fully automatic flights. Our contributions to this research could significantly impact the safety of aerial vehicles, inspiring and motivating further improvement in the field.

Among the circumstances that can overload the crew's work are emergencies, which are the topic of this article. In response to an emergency during flight, the pilot's first objective is to stabilize and regain control of the aircraft. Then, they focus on the quickest

and safest way to land the plane, with the assistance of the Flight Management System, flight manuals, and air traffic controllers. This decision is often difficult because it depends on many factors, such as the new maneuvering envelope, the available fuel, the distance to landing fields, and terrain avoidance.

Several projects and working groups have already addressed this issue from different points of view. For instance, the study [1] focused on the case of emergency landings for damaged aircraft, specifically for obstacle avoidance. Another study, which used an Unmanned Aerial Vehicle (UAV) model, dealt with an obstacle avoidance problem, mainly in managing the landing sites during an emergency landing [2]. A guidance aid in case of systems failure was addressed in another study, where the generation of smooth trajectories facilitates manual landing within this damaged flight envelope.

One lesson from these previous works [3] is that the problem is far too complex to tackle all at once, especially in our case when we try to minimize computational time. The context of the situation justifies our need to have this control over time computation; in an emergency, in the tactical phase, the algorithm's reactivity is a must, unlike many studies performed to meet an objective for the strategic phase. Indeed, we are either working on avoidance problems, as in [4], or solving problems related to the search for a landing site [5], or even trajectory smoothing problems, as we can find solutions in [6]. Also, we notice that, in a general way, the literature presents us with only a partial solution to the problem rather than a complete resolution. By "resolution system", we mean a comprehensive approach that addresses all aspects of the emergency landing problem. Certification is one of the main reasons for partial solutions. With the need for embedded algorithms, the number of usable systems and the ability to solve complex problems are limited. Thus, by repealing this certification limit, we aim for a more comprehensive treatment of the plan: Our ambition is first to focus on the resolution system before focusing on the certification question.

1.1. Literature Review

The field of autonomous driving has witnessed the application of various decision-support architectures. Rule-based [7,8] and knowledge-based [9] methods achieved success in controlled environments. However, these methods are not without their challenges, demanding a comprehensive understanding of the traffic environment and significant time and memory resources for computations. This underscores the need for a more efficient and effective solution.

Given the intricacy of the task, current methods often concentrate on a single aspect of the problem, be it the multiple of the trajectories [10], the smoothing of the trajectory [11], the computation of an optimal route [12], the calculation of a dynamic optimal trajectory [13], etc. The cornerstone of these planning-based methods is trajectory planning, i.e., generating a trajectory for a given scenario [14,15]. However, their applicability must still be completed in a highly interactive environment, such as in emergencies. This research, therefore, focuses on the planning scheme of multiple optimal trajectories, from one start point to one or more landing points, in an emergency scenario, aiming to fill this gap.

The complexity of the problem at hand necessitates a multi-faceted approach, ruling out a single-pass or single-program solution. As demonstrated in [16,17], we have tackled this issue using decomposition. This research, therefore, introduces a crucial multi-layer architecture that effectively addresses all aspects of the problem, underscoring its significance in the field of autonomous driving and artificial intelligence.

1.2. Problem Description

Our research centers on a specific problem: computing one or more trajectories between a starting point and a well-defined landing point while avoiding obstacles in a 3D

cluttered environment. To tackle this problem, we employ tools from avoidance problems, path-smoothing tools, and optimal control theory to find the best algorithm combination for each aspect of our problem.

The mathematical formulation of the safety trajectories is a novel approach intricately linked to the avoidance analysis of dynamic systems. This research sets out to categorize the problem into three distinct areas: first, the avoidance problem in a 3D environment; second, the exploration of smoothing tools for paths to create flyable paths without abrupt variations in the input parameters; and finally, a comprehensive survey of the situation and the dynamic context in which the vehicle operates. This study unveils the potential to meet all the following criteria:

1. Compute one or more paths in a 3D cluttered environment, with different obstacles of different density;
2. Compute one or more smooth paths in a 3D cluttered environment;
3. Compute one or more continuous and dynamic trajectories in a 3D cluttered environment.

1.3. Organization

This paper is organized as follows: First, a background of existing methods is presented, where relevant algorithms are highlighted. Second, we present how such methods have been tested and associated to resolve our problem.

2. Survey of Existing Methods

As previously said, the problem of emergency trajectories is too complex to be tackled in one pass. Therefore, we split the problem to manage all aspects more effectively. To accomplish this decomposition, we use the characteristics requested by our trajectories:

1. First, we want to compute an obstacle-free path connecting two points in a known clustered 3D environment. This problem is a classic case widely studied in avoidance problem studies; an overall survey can be found in [18].
2. Second, in case of a problem with the trajectory tracking tools, like Airborne Collision Avoidance Systems (ACAS) or Traffic Alert and Collision Avoidance Systems (TCAS), it is the pilot's responsibility to ensure the continuation of the flight. Thus, producing a path as smooth as possible is necessary to avoid sudden changes in speed or angles. This issue is related to a smoothing path problem: the aim is to design a flyable path over or close to the objective points while satisfying constraints such as maximum curvature and G2 continuity (Two curves having a common point and tangent vectors lying along the same direction and having the same curvature (which is, the same rate of change of the direction) are said to have curvature continuity. The directions at the joint seem to change with the same "speed"). This study has been addressed using approaches like nonlinear optimization algorithms in [19].
3. Finally, we consider the dynamic model. Indeed, considering the aircraft's dynamics and the state of the flight controls, this multi-dimensional approach can be obtained using optimal control theories, which may provide the entire set of optimal parameters (including the attitude angles, velocities, fuel, etc.) for a trajectory between two points as a function of time and subject to the particular restrictions involved. The study of these methods, described in [20], implies a complete analysis in time and space. Hence, treating this last problem allows us to convert our paths into trajectories.

This paper focuses on a particular emergency; indeed, we place ourselves in a static context and only consider static obstacles. Thus, the terrain database will entirely define our obstacle database, such as potential non-fly zones, terrain relief, etc.

2.1. Path Planning

Motion planning for an automated robot has been widely discussed over the past 50 years. Two main categories of approaches are usually distinguished: deliberative and reactive processes (see Figure 1).

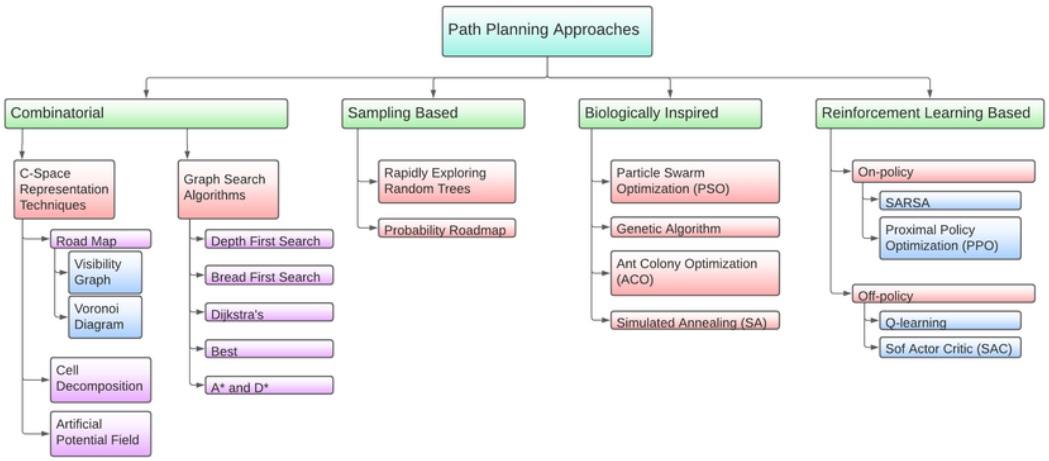

Figure 1. Classification of path-planning algorithms [21].

- The principle of "deliberative approaches" is to determine a complete movement of the robot between an initial position and a final position from a model of the environment in which the system evolves;
- "Reactive approaches", on the other hand, only compute the movement to be applied to the next time step from the sensor data retrieved by the robotic system at each instant. A representation of the environment is thus constructed as the movement progresses: navigation is, therefore, possible in an uncertain environment, as well as in a dynamic environment.

The importance of having a complete journey, therefore, means that deliberative approaches are best suited to our problem. We meticulously compared the most used deliberative algorithms in the field, both in their initial forms and modified forms, to ensure we found the best answer to the problem.

1. A* algorithm: A* is the most widely used algorithm in robotics. The A* algorithm is a path search algorithm in a graph between an initial node and an ending node. It uses a heuristic on each node to estimate the best path and then visits the nodes based on this heuristic evaluation. It is one of the so-called "admissible" algorithms because its purpose always guarantees to find the shortest path. This algorithm was first proposed by Peter E. Hart, Nils John Nilsson, and Bertram Raphael in 1968 [22]. Each node of a cost function to evaluate a path can be expressed as follows:

$$f(n) = g(n) + h(n) \quad (1)$$

where f is the cost of the node, g is the distance separating the current node from the initial node, and h is a lower bound of the distance separating the current node from the final node.
The algorithm evaluates each node to find the best resultant $f(n)$ corresponding to the shortest path. The properties of the $A\star$ algorithm make it the most used in mobile robotics. Reference [23] shows it is optimal in its computational times and faster than any other shortest-path algorithm using the same heuristic. In addition, this method has many variants and can solve many situations.

We chose to study this algorithm in modeling the avoidance problem because, in addition to being one of the best-known, it has refinement properties linked to path building. $A\star$ is part of the graph search methods. Thus, the refinement of the grid depends on the user. From this point of view, we find a strong advantage in the study of obstacle avoidance in having a refined grid around the obstacle and looser and more spaced points with free space.

2. PRM algorithm: The Probabilistic Roadmap Method (PRM) [24] is a discrete version of a continuous (c-space) that contains far fewer states than the original space. It is generated by randomly sampling the most significant c-space and connecting these points in a roadmap.
   PRMs are unique among other cell decomposition planners, those who try to solve the planning problem by decomposing the whole search space into multiple cells. Path planning using a PRM has two phases:
   (a) A construction phase of the roadmap: random points of the free space are chosen and added to a list of feasible points. The mapping algorithm tries to connect those feasible points if the associated links do not intersect with the obstacles. This list of links is the roadmap;
   (b) A path query phase: In the query phase, when the robot needs to plan a path between two points, the algorithm uses the roadmap created in the first phase.

   Due to its random search, the PRM is not an algorithm subject to the curse of dimensionality. It also makes it an excellent candidate for solving the avoidance problem.
3. RRT algorithm: Another possible algorithm is a sampling-based method named Rapidly Exploring Random Tree (RRT). Popularized by Karaman and Sertac [25], RRT is a random sampling of the configuration space; it starts at the starting location and randomly grows a tree to span space. The main objective is to favor the extension of the tree to areas that still need to be filled. The planner pushes the search tree from previously constructed vertices.
   The principle is as follows: from an initial point q_{init}, the neighborhood of such a point is explored by arbitrarily choosing a new unobstructed configuration qrand at each iteration. The second step determines the q_{near} node closest to q_{rand} in the existing tree. The next step extends the tree from q_{near} in the direction of q_{rand} by a length ϵ. Finally, if this extension succeeds, the newly created point q_{new} will be added to the tree. This process is repeated until the destination q_{goal} is reached. This principle is illustrated in the Figure 2.

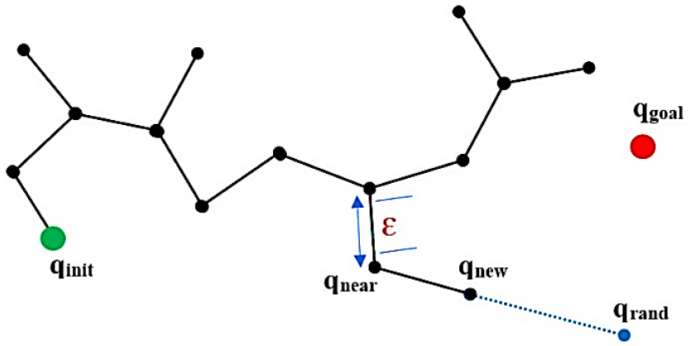

Figure 2. The construction process of RRT [25]. Input: Initial points q_{init}, q_{goal}. Incremental distance ϵ. Output: RRT graph G.

The RRT exploration method contains a free space-spanning tree construction and query phases. Its performance comes from the fact that it does not require a precomputation stage. A suitable property of this approach is that the growth is strongly

biased towards unexplored areas of free space, so rapid exploration occurs. This construction allows, in particular, the processing of large dimensions; thus, the search remains steady in areas of immense dimensions.

2.2. Path Smoothing

Then, we need to be able to provide a smooth path. Furthermore, we need to process the route computed by our path-planning algorithm.

Smoothing paths for a non-holonomic mobile [26] robot is the subject of many works. The first approach consists of generating paths of straight-line segments tangentially connecting circular arcs of maximum curvature. These paths are the shortest in configurations where the robot only moves forward (as demonstrated by Dubins in 1957 [27]). However, several studies [28,29] on the control of mobile robots have highlighted the importance of the continuity of curvature to obtain precise tracking paths (although the mentioned paths do not verify this property). Therefore, the proposed solutions introduce interpolation and approximation techniques to geometrically model paths for mobile robots. Among these primitive paths are cubic splines, the first methods to be developed. Bezier curves were then introduced, adopting a more flexible design. The evolution continued with B-Splines, well defined here [30]; the generalization of Bezier curves, described in [31]; and Non-Uniform Rational Basis Splines, also called NURBS, in the 1980s. Next, we present an overview of the path-smoothing techniques used in robotic navigation.

1. Dubins Splines: [27] showed that an optimal path between two points is a piecewise curve C_1 and C_2 without obstacles. This curve is formed by circular arcs connected tangentially by segments of lines and can be written in the form CSC or CCC with C as a circular arc of minimal radius, i.e., a radius equal to 1 and S as a straight line segment [32]. It should also be noted that each arc of a segment or circle can be degenerated, that is, of zero length.
   This path can be specified more precisely by indicating its direction of travel. Indeed, we denote by R (respectively L) a circular arc traversed in the direct or clockwise direction (respectively, counterclockwise). Thus, there are at most six Dubins paths between two points:
   - The CCC type containing the two paths of LR_uL and RL_uR types where u represents the length of the intermediate arc, satisfying $\pi \leq u$.
   - The CSC type containing the four paths of the LSL, LSR, RSL, and RSR types.
2. Cubic Splines: The automotive industry developed spline interpolation at General Motors around 1950 [30]. Cubic spline interpolation consists of finding for each interval $[x_k, x_{k+1}]$ a polynomial $S_k(x)$ of the third degree so that the resulting interpolation function of $[x_0, x_n]$, as well as its first and second derivative, are continuous:

$$S_k(x) = a_k x^3 + b_k x^2 + c_k^x + d_k, x \in [x_k, x_{k+1}] \tag{2}$$

   Spline functions offer flexibility (i.e., no break in the curvature radius) at the interpolation function's level. However, all the points are linked, and changing the location of one of them affects the whole system of equations.
3. Non-Uniform Rational B-Splines, also called NURBS [33], are defined by the following equation:

$$C(u) = \frac{\sum_{i=0}^{n} N_{i,p}(u) w_i P_i}{\sum_{i=0}^{n} N_{i,p}(u) w_i} \; a \leq u \leq b \tag{3}$$

   where the $\{P_i\}$ are the control points (forming a control polygon) and the $\{w_i\}$ are their associated weights. The $\{N_i, p(u)\}$ are the B-spline basis functions of degree p defined on the nodal vector U. The nodal vector is a sequence of parameter values

that determine where and how the control points will affect the shape of the curve. It divides the parameter space into intervals, and each time the parameter enters a new nodal interval, a unique control point becomes active, while an older one becomes inactive. This ensures the local influence of the control points, providing a clear understanding of their role in shaping the curve. This vector is built with an increasing sequence of nodes ($u_i \leq u_{i+1}$) between 0 and 1. These consecutive nodes can have identical values: this is defined by a node diversity, which allows it to accentuate a point influence on the curve.

Thus, to ensure the interpolation of the two ends of the control polygon, it is necessary to fix the diversity of the first and the last node of U to the value $p+1$. Note also that the degree p, the number of control points $n+1$, and the number of nodes $m+1$ of the nodal vector are related by $m = n + p + 1$.

As for the weight parameter, each w_i plays a crucial role in determining the impact of the point P_i on the curve. Increasing weights associated with points pulls the curve towards these points, providing a strong sense of control and predictability; conversely, when weights are decreased, the curve moves away.

If $w_i = 1$ for any point, we find ourselves in the case of B-splines. Otherwise, for values different from 1, higher or lower importance is attributed to the corresponding control point:

- $w_i \leq 1$: gives a curve farer to P_i;
- $w_i \geq 1$: gives a curve closer to P_i;
- $w_i = 0$: the point P_i no longer has any influence;
- $w_i \to \infty$: the curve passes through P_i.

NURBS are B-splines with weighted control points, offering precision that allows the curve to be precisely snapped to one or more points, as desired. They maintain the degree of independence and the property of local modification that characterizes B-splines. Incorporating control point weights significantly enhances flexibility and allows NURBS to dynamically synthesize various curve shapes by adjusting control points, nodes, and weights.

2.3. Time Parametrization

Now that we can successfully compute a path without obstacles, we must introduce the dynamic context. We want to join the points of the route planning by a continuous trajectory that simulates the vehicle's movement.

This paper uses the Point Mass Model for general flight vehicle dynamics. Consider an inertial frame attached to the Earth. The state of the system is defined by the following:

- $(x, y, z) \in \mathbb{R}^3$ is the position of a vehicle in the inertial reference frame;
- $V \in [V_{\min}, V_{\max}]$ is the True Air Speed;
- $\phi \in [-\pi, \pi]$ is the heading angle;
- $\gamma \in [-\pi, \pi]$ is the flight pass angle.

The computation of continuous trajectory is based on the equations of motion, which are defined by the following system, called 5D Dubins model [34]:

$$\begin{cases} \dot{x} &= V sin\phi cos\gamma \\ \dot{y} &= V cos\phi cos\gamma \\ \dot{z} &= V sin\gamma \\ \dot{\phi} &= u_1 \\ \dot{\gamma} &= u_2 \end{cases} \tag{4}$$

where $V(t) = \|v(t)\|, v(t) = (\dot{x}, \dot{y}, \dot{z}) \in \mathbb{R}^3$ is the velocity at time t, ϕ is the heading angle and γ the flight-path angle. Here, the control $u = (u_1, u_2)$ is subject to the following constraint:

$$\|u_1\| \leq u_{\max}, \|u_2\| \leq u_{\max}, \tag{5}$$

where the typical value of the maximal control u_{max} is around 1.

Based on this system of equations, we aim to determine a trajectory that minimizes a specific criterion while satisfying the constraints. The resolution of this type of problem relies on principles of the theory of optimal control.

Definition 1. *In optimal control, we distinguish two vector variables: a control variable, u, to indicate the decisions made, and a state variable, x, to indicate the system's state over time. The standard control model* (P_c) *for a control system is composed of two main components: first, the dynamic controlled system (to which constraints are often added), and second, the functional control system (defining the optimization criterion):*

$$P_c \equiv \begin{cases} \min\limits_{u,x} J(u(t), x(t)), \text{ (corresponding to the cost to be minimized)} \\ \text{Under constraints} \\ \dot{x}(t) = f(u(t), x(t), t), \text{ (controlled dynamic system: state equations)} \\ x(t_0) = x^{(t_0)}, x\left(t_f\right) \in C_f, \text{ (initial and terminal conditions, on the states)} \\ g(u(t), x(t), t) \leq 0, \text{ (conditions on control and/or state)} \end{cases} \tag{6}$$

where:

- $J : \mathbb{R} \times \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}$ *is* $\mathcal{C}^1$.
- $f : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$ *is* $\mathcal{C}^1$.
- $g : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$ *is* $\mathcal{C}^1$.

The following statement is the most usual Pontryagin Maximum Principle (PMP), well developped in [35], valuable for general nonlinear optimal control problems (6), with control constraints but without state constraint.

Theorem 1. *If the trajectory* $x(t)$*, associated to the optimal control* $u(t)$ *on* $[t_0, tf]$*, is optimal, then it is the projection of an extremal* $(x(t), p(t), p^0, u(t))$ *(called extremal lift), where* $p^0 \neq 0$ *and* $p(t) : [t_0, t_f] \to \mathbb{R}^n$ *is the adjoint vector, with* $(p(t), p^0) \neq (0, 0)$*, such, that*

$$\begin{cases} \dot{x}(t) &= \frac{\partial H}{\partial p}(x(t), p(t), p^0, u(t), t) \\ \dot{p}(t) &= \frac{\partial H}{\partial x}(x(t), p(t), p^0, u(t), t) \end{cases} \tag{7}$$

almost everywhere on the time interval $[t_0, t_f]$*, where* $H(x, p, p^0, u, t) = \langle p, f(x, u, t)\rangle + p^0 f^0(x, u, t)$ *is the Hamiltonian,*

$$H(x(t), p(t), p^0, u(t), t) = \max_{v \in \mathcal{U}} H(x(t), p(t), p^0, v(t), t) \tag{8}$$

almost everywhere on the time interval $[t_0, t_f]$*. Moreover, if the final time* t_f *to reach the target* M_1 *is not fixed, then the following condition must be met at the final time* t_f

$$\max_{v \in \mathcal{U}} H(x(t), p(t), p^0, v(t), t) = -p^0 \frac{\partial g}{\partial t}(x(tf), tf). \tag{9}$$

Additionally, if M_0 and M_1 (or just one of them) are submanifolds of $\mathbb{R}^n$ locally around $x(t_0) \in M_0$ and $x(t_f) \in M_1$, then the adjoint vector can be built to satisfy the transversality conditions at both extremities (or just one of them):

$$p(t_0) \perp T_{x(t_0)} M_0 \;\; p(t_f) - p^0 \frac{\partial g}{\partial t}(x(t_f), t_f) \perp T_{x(t_f)} M_1, \tag{10}$$

where $T_x M_i$ denotes the tangent space to M_i at the point x_i.

Over the past few decades, many numerical methods in optimal control have emerged [36]. While it is impossible to list them all, we will focus on the local deterministic methods, a subset that presents a unique set of challenges:

- Indirect methods [37]: these methods use the conditions of the PMP to determine the controls u as a function of the states x and the adjoint states p and to reduce to the resolution of an algebra-differential system. This last system will be transformed into a nonlinear problem of finite dimension to determine the initial assistant states $p(t_0)$, making it possible to obtain the final state $x(t_f)$.
- Direct methods [38]: by appealing to a total or partial "discretization" of the optimal control problem (in the sense that discretization allows the transformation of the continuous problem into a discrete problem with a finite number of variables), these methods refer partly to the original problem as a nonlinear programming problem.

However, the literature has allowed us to identify the advantages and drawbacks of these methods, which we summarize in Table 1 below [20].

Table 1. Characteristics of the direct and indirect methods.

Direct Methods	Indirect Methods
Simple implementation, without prior knowledge	A priori knowledge of the structure of the optimal trajectory
Insensitive to choice of initial condition	Susceptible in the choice of the initial condition
Ease of taking state constraints into account	Theoretical difficulty of taking state constraints into account
Globally optimal closed-loop controls	Locally optimal open-loop control
Low and medium numerical precision	Very high numerical precision
Effective in low dimension	Effective in any dimension
Problem of local minima	Small domain of convergence
Heavy in memory	Parallelizable computation

Stepping back from the theory, we find ourselves leaning towards direct methods. To illustrate this, we will use a concrete example provided by [39], as depicted in the following figure (Figure 3), to compare the methods.

1. Single shooting method: The main idea begins meticulously discretizing the control u and the denoted state x. This transforms the initial optimal control problem into a finite-dimensional, potentially significant, nonlinear programming problem (NLP). The single shot involves approximating the trajectory using a single integration. In other words, it is like firing a cannonball, checking where the ball has landed, adjusting the initial velocity, and repeating. This process is repeated until the error between the target and the place where the ball landed, called a defect, denoted ϵ, is reduced to zero ($\epsilon \to 0$). The optimization algorithm is designed to find the initial speed that achieves this precision. This behavior is illustrated in Figure 4.

2. Multiple shooting method: This method is designed for efficiency. It starts by discretizing a partial problem in control and state, defining a parameterization of the control on each time interval $[t_l, t_{l+1}]$, and choosing a numerical integrator for the resolution of the ODEs. The optimal control problem is then discretized into an NLP problem. Multiple shooting works by dividing the initial problem into sub-problems and solving them in parallel. In the end, continuity constraints between the segments are added, ensuring a smooth and efficient solution.This behavior is illustrated in Figure 5.
3. Direct collocation method: This method is robust and reliable. It starts by assuming a specific representation of the state, particularly a "polynomial". Then, it ensures that the dynamic equations are followed in a finite number of instants under the collocation conditions. These intermediate instants in $[t_l, t_{l+1}]$, called "quadrature instants", subdivide $[t_l, t_{l+1}]$ into $k-1$ periods and are used by the digital dynamics integration scheme. Piecewise polynomial state trajectories are obtained, which converge towards the dynamics of the controlled system at the collocation points. The discretization of the problem in optimal control is then carried out, and an NLP problem is generated. This behavior is illustrated in Figure 6.

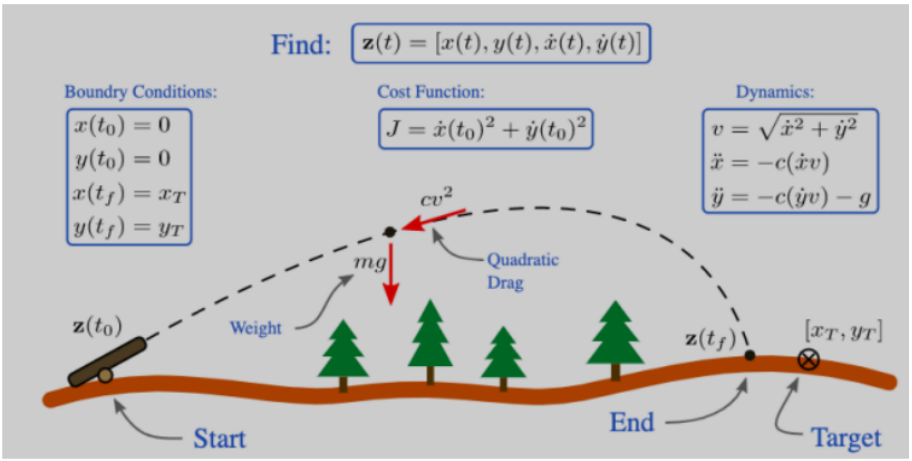

Figure 3. OCP problem [39]. This situation can be modeled as a boundary-value problem. The goal is to find the trajectory, assuming that (1) the cannonball is a point mass, (2) air friction is modeled using quadratic drag, and (3) the amount of powder is proportional to the initial speed squared.

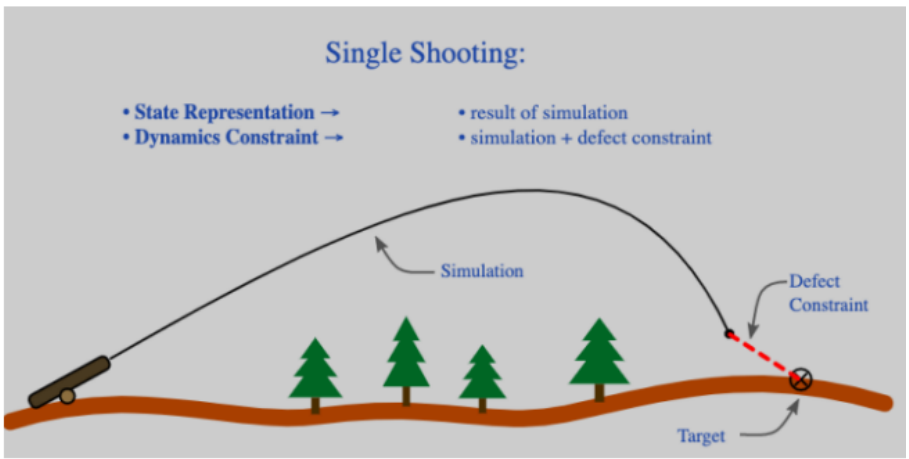

Figure 4. Single shooting scheme [39]. It approximates the trajectory using a single simulation: fire the cannon, check where the ball landed, adjust the initial speed, and repeat.

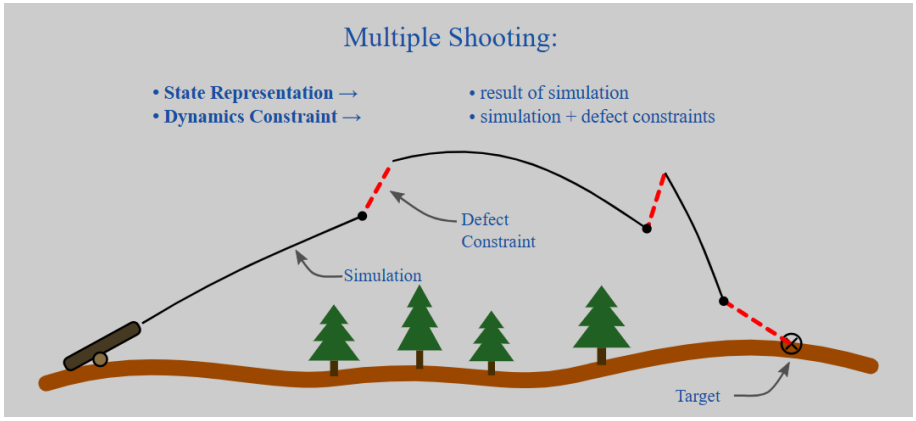

Figure 5. Multiple shooting sheme [39]. It works by breaking the problem into smaller ones and solving them parallelly. In the end, each segment is constrained to connect to the previous one.

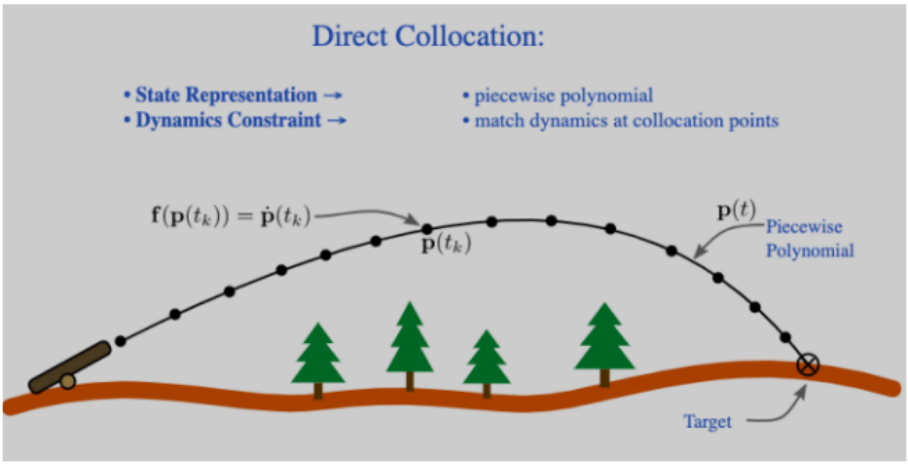

Figure 6. Collocation shooting scheme [39]. The trajectory is approximated using a piecewise polynomial. Physics is satisfied by requiring that the dynamics match the derivative of the polynomial at each collocation point.

3. Experiences and Results

3.1. Experiences

Now that we have presented and described the relevant design algorithms, we present a scenario to select the fastest algorithms to implement our trajectory computation problem. Thus, by considering the various sub-problems previously stated, we outline the algorithm most likely to answer this sub-problem. The goal is to end up with the most efficient combination in terms of time for computing emergency trajectories.

3.1.1. Path Planning Problem

To support a validation of the 3D trajectory planning of aerial vehicles, we analyzed several algorithms to extract the three that are the most significant to our problem, the quickest in time computation.

it is crucial to note that this section conducts a comprehensive theoretical comparison of those algorithms, with their characteristics succinctly summarized in Table 2. Subsequently, they undergo a rigorous evaluation based on the computed results.

As mentioned earlier, the primary focus is on time computation. Given our objective to handle emergencies, the computational time of the program takes precedence over all other characteristics, including the path length, particularly in our case where the avoidance problem is the initial step for our solution.

For our tests, we take the cluttered 3D environment below, dedicated to UAVs for which the number of obstacles (in blue on the Figure 7) has been fixed at 10, 15, and 20 (Table 3).

Table 2. Comparison of different path planning method properties [40].

Method	Type	Time Complexity	S/D Environment	Structure	Real-Time
A⋆	Node/Grill-Based Algorithms	$\Theta(mlogn) \leq T \leq \Theta(n^2)$	Static	Grill	Online
PRM	Sampling-Based Algorithm	$\Theta(nlogn) \leq T \leq \Theta(n^2)$	Dynamic	Roadmap	Online
RRT	Sampling-Based Algorithm	$\Theta(nlogn) \leq T \leq \Theta(n^2)$	Static and Dynamic	Tree	Online

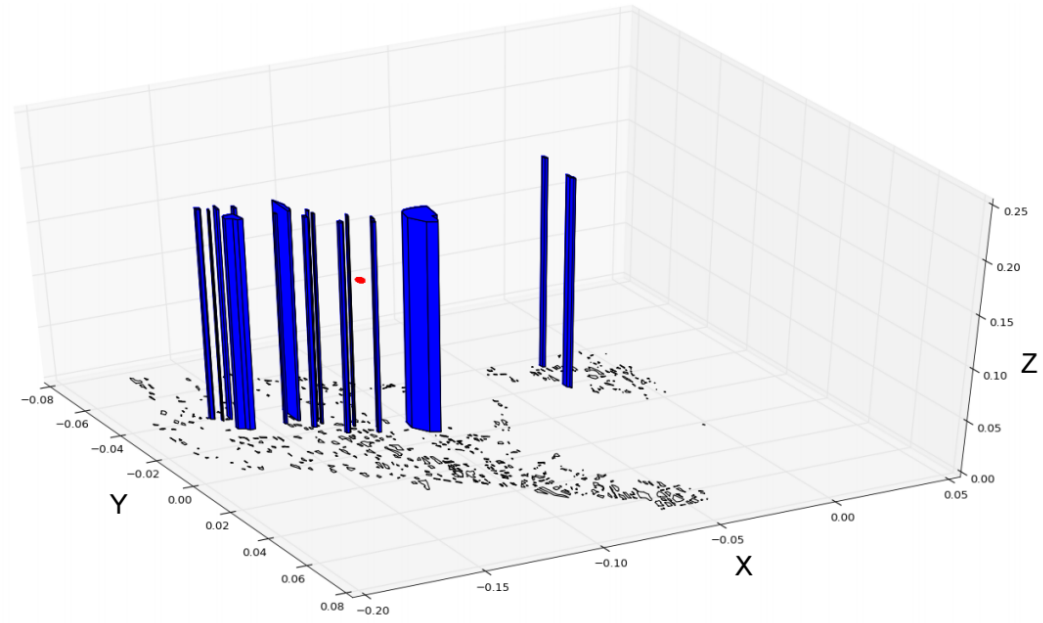

Figure 7. UAV environment. The starting positions are marked with a red point, and the destination positions are black shapes. The blue columns represent the no-fly zones, obstacles that must be avoided.The X, Y, and Z axes represent the longitude, latitude, and altitude coordinates measured in degrees for the lateral and kilometers for the vertical, respectively.

Table 3. Comparison of different 3D path planning methods depending on the number of obstacles. In the table, the bold text highlights the best results.

Obstacles	Methods	Computation Time (s)
10	A⋆	3.12
	PRM	1.50
	RRT	**1.05**
15	A⋆	4.03
	PRM	2.23
	RRT	**1.30**
20	A⋆	6.15
	PRM	4.51
	RRT	**3.30**

When it comes to results, the RRT stands out as the top performer, thanks to its unmatched speed. In the realm of path smoothing, we lean on this algorithm for the initial phase of problem resolution, which involves generating a feasible path. The RRT's rapidity

and efficiency make it a solid choice for this crucial first step, providing you with a sense of being well informed and confident in its capabilities.

3.1.2. Path Smoothing

Due to the importance of generating smooth trajectories with continuous curvatures and controlling and tracking non-holonomic mobile robots, several solutions have been proposed in the literature to compute feasible solutions.

In this comprehensive review, we have meticulously outlined the proposed methods for resolving this problem, starting from the initial solutions that introduced only non-smooth curves (ignoring the continuity of curvature) to the various splines used to model the curves' paths (cubic splines and NURBS). Our thorough exploration of NURBS curves, highlighting their significant advantages over other techniques, leads us to the conclusion that NURBS is the most suitable method for solving our smoothing problem. We have also compared the various methods in the same environment as before. For the resolution of the problem, two alternatives were offered:

1. The combination with the RRT: The path-planning algorithm allows for different expansion methods, notably smoothing techniques. This choice increases the computational time but allows for the simultaneous processing of both the avoidance and smoothing problems (Table 4).

Table 4. Comparison of different smooth path planning methods depending on the number of obstacles.

Obstacles	Methods	Time Computation (s)
10	RRT + Dubins	8.32
	RRT + Cubic Splines	5.32
	RRT + NURBS	**2.00**
15	RRT + Dubins	10.76
	RRT + Cubic Splines	9.34
	RRT + NURBS	**3.12**
20	RRT + Dubins	20.0
	RRT + Cubic Splines	12.43
	RRT + NURBS	**4.97**

2. The post-path-planning phase uses smoothing methods after computing the reference path. This choice forces us to add an obstacle checker, which ensures the path remains obstacle-free after smoothing. The obstacle checker scans the environment and alerts the system if any obstacles are detected, preventing potential collisions (Table 5).

Table 5. Comparison of different smoothing planning methods depending on the number of obstacles.

Obstacles	Methods	Time Computation (s)
10	Dubins	0.11
	Cubic Splines	0.024
	NURBS	**0.019**
15	Dubins	0.20
	Cubic Splines	0.027
	NURBS	**0.025**
20	Dubins	0.37
	Cubic Splines	0.030
	NURBS	**0.028**

Regardless of the option we consider, it is clear that the method involving NURBS consistently produces the best computational times. This not only confirms its practicality but also provides insurance about its efficiency. In terms of the two options, they are not directly comparable, and the choice ultimately comes down to the preference for the structure of our architecture. In our case, we have chosen to go with the post-path-planning approach phase.This approach, being independent from the path planning algorithm, empowers us to try different combinations of path planning and smoothing algorithms without running both of them, as is the case with the smoothing algorithms included in the RRT.

3.1.3. Time Parametrization

A drawback of direct-shooting methods is the dependence on the initial point. This can lead to numerical instabilities during the numerical integration process. For instance, even minor variations in the initial conditions x0 can result in significant changes in the terminal conditions. In light of this, the collocation method emerges as the most reliable solution for our problem, a conclusion we will validate empirically. The reliability of the collocation method gives us confidence in its use. To compare the three methods, we initiate the resolution of the following OCP:

$$(P_c) = \begin{cases} \min\limits_{u,x} \int t_0 = 0_f^t \|u(t)\| dt \\ \text{Under the constraints} \\ \dot{x} = f(u(t), x(t), t) \\ x(t_0) = A, x(t_f) = B, \end{cases} \tag{11}$$

where f is the function presented in (4), which represents the objective function of our optimization problem, with different starting points A and ending points B (Table 6).

Table 6. Comparison of different direct optimal control methods depending on the scenario.

Environments	Methods	Time Computation (s)
$A = [5, 11, 8, \frac{\pi}{6}, \frac{\pi}{3}, \pi], B = [10, 20, 30, \frac{\pi}{6}, \frac{\pi}{3}, 5]$	Single Shooting	4.5
	Multiple Shooting	4
	Collocation	**2.7**
$A = [10, 20, 30, \frac{\pi}{6}, \frac{\pi}{3}, 5], B = [5, 11, 8, \frac{\pi}{6}, \frac{\pi}{3}, \pi]$	Single Shooting	10
	Multiple Shooting	5.1
	Collocation	**2.9**
$A = [100, 50, 20, \frac{\pi}{4}, \frac{\pi}{2}, \pi], B = [10, 20, 30, 0.9, \pi]$	Single Shooting	174
	Multiple Shooting	63
	Collocation	**5.7**

As always, the computation time is this survey's primary criterion for comparisons. Whether from an empirical or theoretical standpoint, we can confidently assert that collocation is the most efficient method for solving OCP through direct methods.

3.1.4. Synthesis

The results demonstrate that we can determine the best algorithm for each subproblem. Thus, the combination of the three algorithms allows us to compute a trajectory validating the following three criteria:

1. To be obstacle-free, which is what the RRT path-planning algorithm checks;
2. To be smooth, which is the result of the path smoothing part via NURBS;

3. To be a continuous trajectory responding to the dynamic context, which is processed through the time parametrization solver.

Our structure, which includes the combination of the three algorithms and the trajectory calculation process, is designed to efficiently compute trajectories. This quick computation of trajectories is a key advantage of our approach, effectively addressing our problem.

3.2. Results

We use a numerical approach to support our remarks and to present our analysis from an industrial perspective. The objective is to compute an emergency trajectory for a drone, a task that is of significant importance in the field of drone technology. The starting point is clearly defined, and there are multiple landing zones.

This use case involves the search for one or more emergency trajectories in the context of a drone failure in mid-flight. With knowledge of the environment, our solver must compute trajectories, respecting the criteria we have previously defined. The considered airspace is reported in Figure 6 above.

First, three paths are generated in the previously defined environment. Starting from the same starting point, they connect different landing sites chosen randomly by the algorithm. The complexity of the computation is linked to the position of the initial point. Indeed, we need to compute several paths leading to different landing sites. In searching n paths, we begin by running n times a simple RRT algorithm. Its random behavior might allow us to find different paths, in our case, three. We must compute them simultaneously to always end up with distinctly different paths.

The process remains similar to the standard RRT. Each tree explores the search space from the same q_{start}. A random point q_{rand} is chosen for each tree, and then the point closest to q_{rand} in the tree, q_{near}, is selected. For each pair $\{q_{near}, q_{rand}\}$, the point q_{new} is computed (this point is at a distance δ from q_{near} in the direction of q_{rand}). If q_{new} does not belong to the obstacle space, it is added as a new path point to the corresponding tree. The process is repeated until each tree is linked to one of the endpoints (see Figure 8). The steps of the proposed approach for multiple paths are shown below in Algorithm 1.

Algorithm 1 Multiple RRT pseudocode

Generate k Path($q_{start}, q_{goal,k}, maxiter, obstacle, N$)
for k = 1 to N **do**
  $Tree_k \leftarrow q_{start}$
  while $q_{goal,k}$ not in $Tree_k$ or $i = maxiter$ **do**
    $q_{rand} \leftarrow RandomNode()$
    $q_{near} \leftarrow NearestNode(Tree_k, q_{rand})$
    $q_{new} \leftarrow NewNode(q_{rand}, q_{near})$
    if no_collision($q_{new}, obstacle$) and no_ intersection(q_{new}, Matrix_Tree) **then**
      $q_{nearnode} \leftarrow NearNodes(q_{new})$
      $q_{new} \leftarrow SelectParent(q_{nearnode})$
      if q_{new} **then**
        $Tree_k$.addnode(q_{new})
        $i \leftarrow i + 1$
      end if
    end if
  end while
  Matrix_Tree.addtree($Tree_k$)
end for

Then, we smooth them out using the second algorithm (see Figure 9). The NURBS method above belongs to the class of approximation methods. Therefore, the newly computed curves do not necessarily interpolate with the previous obstacle-free waypoints, thus presenting a risk of obtaining paths that intercept obstacles. To avoid this issue, we are forced to use a trajectory checker, which works as follows:

- Initially, each waypoint has its weight initialized to $w_i = 1$.
- For each point of the calculated smooth paths, we check if there is an interception with an obstacle: if there is an interception, the NURBS curve will be recalculated with an update of the weights. As previously discussed, the greater the weight of a waypoint, the more critical the point.

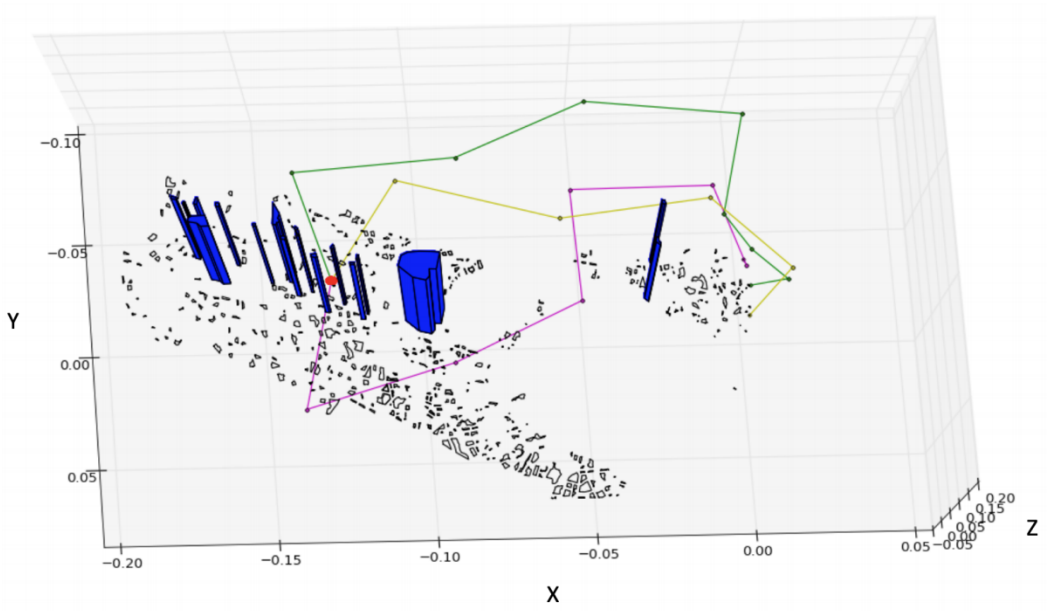

Figure 8. Computation of the flight paths. Input: environment presented in Figure 7. Output: list of waypoints for the three computed flight paths.

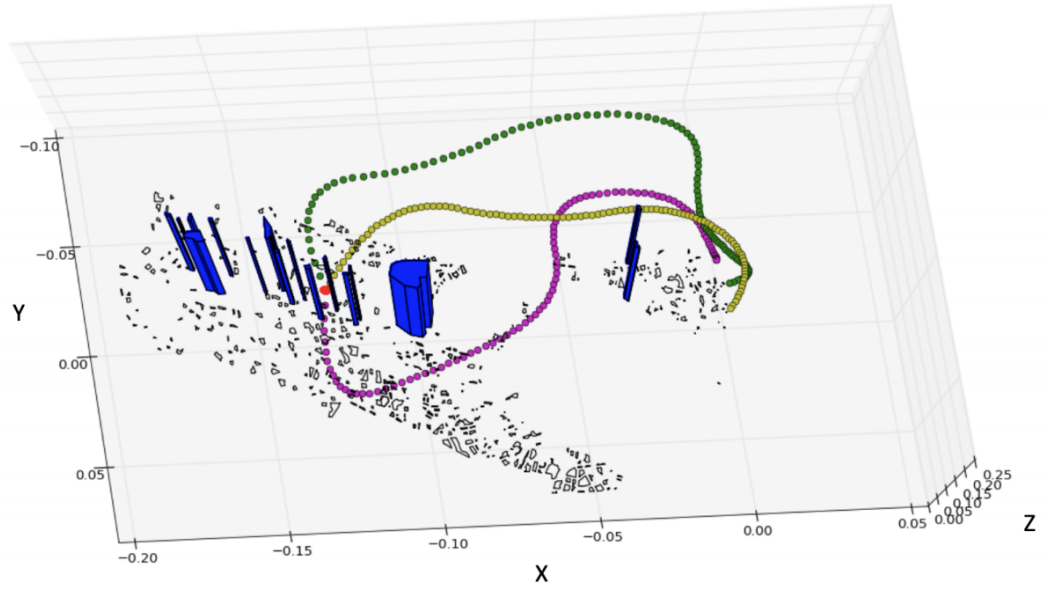

Figure 9. Computation of the smooth flight paths. Input: list of waypoints for the three computed flight paths present in Figure 8. Output: list of waypoints for the three computed smooth flight paths.

Finally, now that the flight plans are well established, we use them as a reference to calculate the continuous trajectories. Considering the UAV's dynamics, previously defined in Equation (4), we can calculate three 5D Dubins trajectories avoiding obstacles (see Figure 10). The color lines in Figures 8–10 highlight the three different paths.

Therefore, we consider the optimal control problem:

$$J = min\ t_{a,f}, \forall a \in 1, \ldots, N_a, \tag{12}$$

subject to the Point Mass Model, where $a \in \{1, \ldots, N_a\}$ is the path index, N_a is the number of paths, and t is time. The function f represents our dynamics, detailed in (4) and (5).

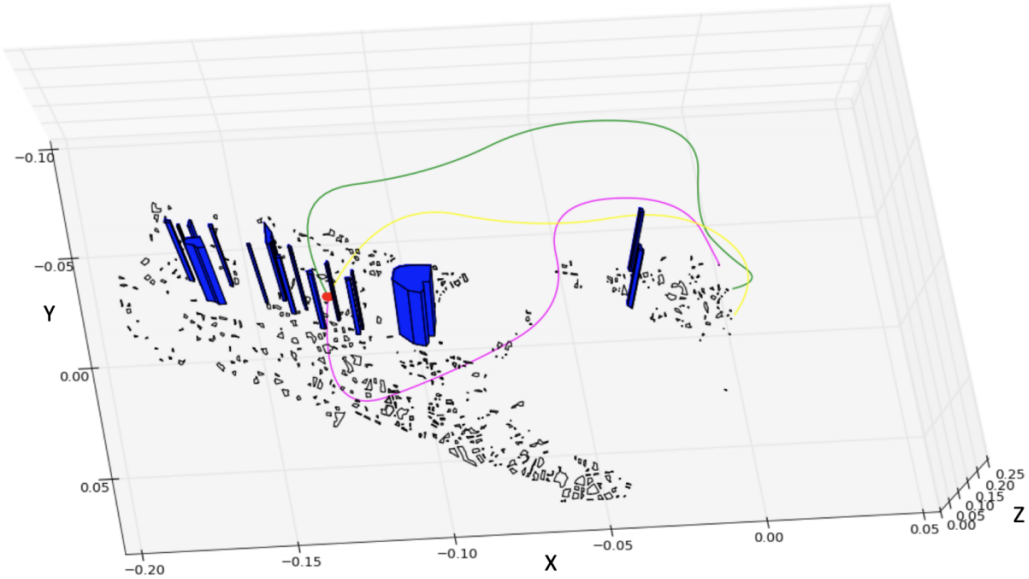

Figure 10. Computation of the dynamic trajectories. Input: list of waypoints for the three computed flight paths present in Figure 9. Output: list of waypoints for the three computed dynamic trajectories.

These examples validate the effectiveness of our method for creating emergency trajectories. Each step, including the computation time, is meticulously detailed in the following Table 7:

Table 7. Time computations(s) of each step, based on the number of computed trajectories.

Number of Computed Trajectories	Paths Planning (s)	Paths Smoothing (s)	Continuous Trajectories (s)
3	1.2	0.06	15
4	1.5	0.2	20
5	12.3	1.8	25
6	13.8	2.1	30
7	25.2	7.5	35
8	40.7	10.4	40
9	51.2	11.2	45
10	65.4	13.6	50

4. Conclusions

This article introduces a novel approach for producing multiple emergency trajectories. By creating a comprehensive system that incorporates both path-planning and trajectory-planning algorithms (see Figure 11 below), we enable the planner to swiftly compute several trajectories towards different known sites while considering the environment and

the UAV's motion constraints. This unique method combines the RRT and NURBS to obtain smooth paths in a 3D cluttered environment. The path-planning part generates numerous reference paths used in the trajectory-planning section. Based on a collocation method, this part is dedicated to trajectory planning. The approach uses a motion equation system based on 5D Dubin's vehicle dynamics to output the expected trajectories.

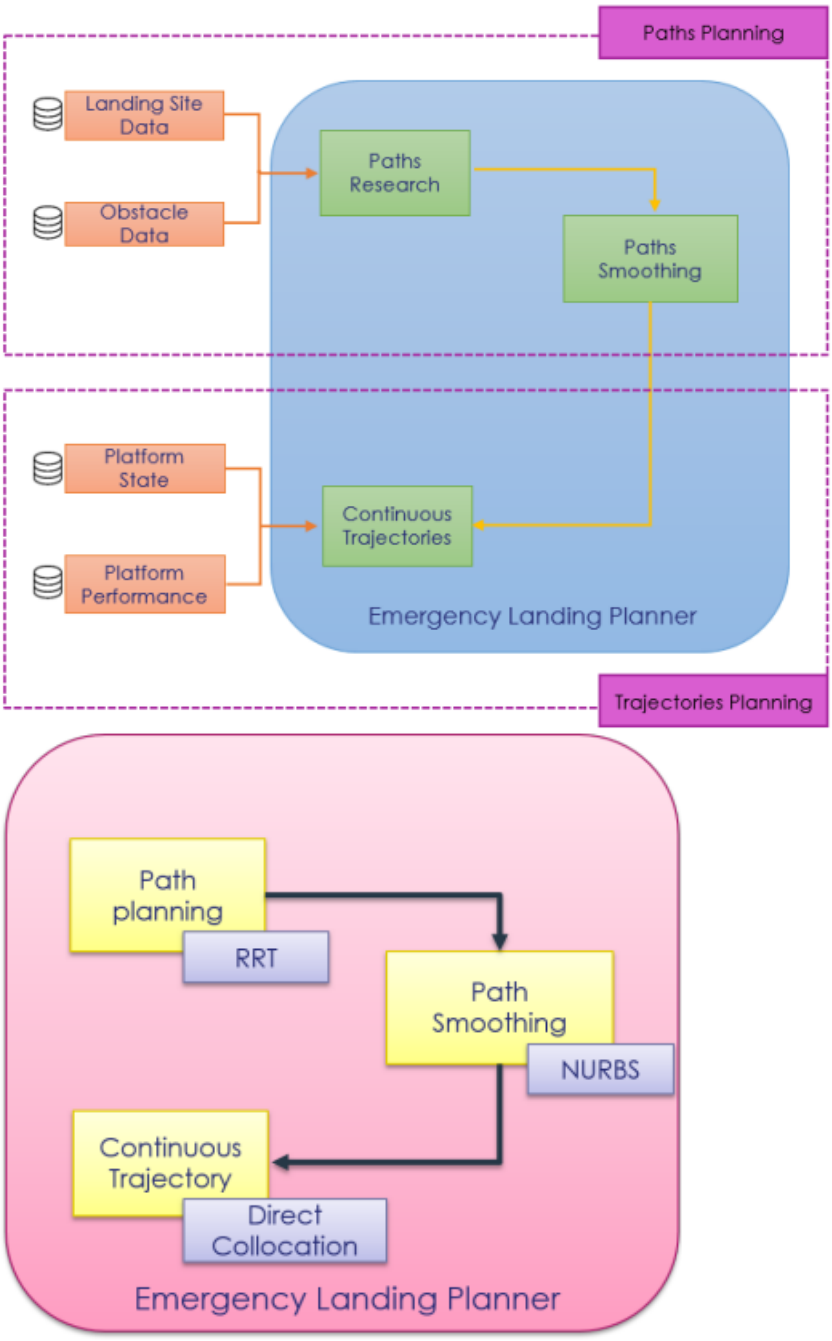

Figure 11. Emergency landing planner architecture.

Looking ahead, our future work holds several extensions. We plan to implement multi-trajectories in a 3D dynamic environment, incorporating wind data and non-constant speed in the dynamics. The approach could also be expanded to large-scale multi-path systems by appropriately parallelizing the computation to reduce computational time. This opens up avenues for further research. Lastly, the challenge of real-time analysis presents a significant opportunity for improvement, sparking excitement and motivation for potential future advancements.

Author Contributions: Writing—original draft, M.O.-O.; Writing—review, M.O.-O., D.G., D.D. and P.-L.G.; Writing—editing, M.O.-O. and D.G. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The original contributions presented in this study are included in the article. Further inquiries can be directed to the corresponding author.

Conflicts of Interest: Author Maëva Ongale-Obeyi was employed by the company Thales Avionics France. Author Damien Goubinat was employed by the company Thales Canada Inc. The remaining authors declare that the re-search was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

References

1. Tomlin, C.J.; Mitchell, I.; Bayen, A.M.; Oishi, M. Computational techniques for the verification of hybrid systems. *Proc. IEEE* **2003**, *91*, 986–1001. [CrossRef]
2. Alam, M.S.; Oluoch, J. A survey of safe landing zone detection techniques for autonomous unmanned aerial vehicles (UAVs). *Expert Syst. Appl.* **2021**, *179*, 115091. [CrossRef]
3. Yu, J.; LaValle, S.M. Optimal multi-robot path planning on graphs: Structure and computational complexity. *arXiv* **2015**, arXiv:1507.03289.
4. Zaytoon, J. Hybrid Dynamic Systems: Overview and discussion on verification methods. In *Informatics in Control, Automation and Robotics II*; Springer: Dordrecht, The Netherlands, 2007; pp. 17–26.
5. Girard, A. Reachability of uncertain linear systems using zonotopes. In *International Workshop on Hybrid Systems: Computation and Control*; Springer: Berlin/Heidelberg, Germany, 2005; pp. 291–305.
6. Botchkarev, O.; Tripakis, S. Verification of hybrid systems with linear differential inclusions using ellipsoidal approximations. In *International Workshop on Hybrid Systems: Computation and Control*; Springer: Berlin/Heidelberg, Germany, 2000; pp. 73–88.
7. Ortlieb, M.; Adolf, F.M. Rule-based path planning for unmanned aerial vehicles in non-segregated air space over congested areas. In Proceedings of the 2020 AIAA/IEEE 39th Digital Avionics Systems Conference (DASC), San Antonio, TX, USA, 11–15 October 2020; pp. 1–9.
8. Khachumov, M. The problems of multi-point route planning and rule-based trajectory tracking for an autonomous UAV under wind loads. In Proceedings of the 2018 IEEE 15th International Workshop on Advanced Motion Control (AMC), Tokyo, Japan, 9–11 March 2018; pp. 204–208.
9. Wang, G.; Ge, S.S. General fight rule-based trajectory planning for pairwise collision avoidance in a known environment. *Int. J. Control. Autom. Syst.* **2014**, *12*, 813–822. [CrossRef]
10. Rösmann, C.; Hoffmann, F.; Bertram, T. Planning of multiple robot trajectories in distinctive topologies. In Proceedings of the 2015 European Conference on Mobile Robots (ECMR), Lincoln, UK, 2–4 September 2015; pp. 1–6.
11. Gasparetto, A.; Zanotto, V. A new method for smooth trajectory planning of robot manipulators. *Mech. Mach. Theory* **2007**, *42*, 455–471. [CrossRef]
12. Legrand, K.; Puechmorel, S.; Delahaye, D.; Zhu, Y. Robust aircraft optimal trajectory in the presence of wind. *IEEE Aerosp. Electron. Syst. Mag.* **2018**, *3*, 30–38. [CrossRef]
13. Kalmár-Nagy, T.; D'Andrea, R.; Ganguly, P. Near-optimal dynamic trajectory generation and control of an omnidirectional vehicle. *Robot. Auton. Syst.* **2004**, *46*, 47–64. [CrossRef]
14. Werling, M.; Ziegler, J.; Kammel, S.; Thrun, S. Optimal trajectory generation for dynamic street scenarios in a frenet frame. In Proceedings of the 2010 IEEE International Conference on Robotics and Automation, Anchorage, Alaska, 3–8 May 2010; pp. 987–993.
15. Damerow, F.; Eggert, J. Risk-aversive behavior planning under multiple situations with uncertainty. In Proceedings of the 2015 IEEE 18th International Conference on Intelligent Transportation Systems, Gran Canaria, Spain, 15–18 September 2015; pp. 656–663.
16. Zhang, W.; Kamgarpour, M.; Sun, D.; Tomlin, C.J. A hierarchical flight planning framework for air traffic management. *Proc. IEEE* **2011**, *100*, 179–194. [CrossRef]
17. Samaranayake, P.; Kiridena, S. Aircraft maintenance planning and scheduling: An integrated framework. *J. Qual. Maint. Eng.* **2012**, *18*, 432–453. [CrossRef]
18. Yang, L.; Qi, J.; Song, D.; Xiao, J.; Han, J.; Xia, Y. Survey of robot 3D path planning algorithms. *J. Control. Sci. Eng.* **2016**, 7426913. [CrossRef]

19. Chen, X. Smoothing methods for complementarity problems and their applications: A survey. *J. Oper. Res. Soc. Jpn.* **2000**, *43*, 32–47. [CrossRef]
20. Trélat, E. *Contrôle Optimal: Théorie & Applications*; Vuibert: Paris, France, 2005.
21. Sánchez-Ibáñez, J.R.; Pérez-del-Pulgar, C.J.; García-Cerezo, A. Path planning for autonomous mobile robots: A review. *Sensors* **2021**, *21*, 7898. [CrossRef] [PubMed]
22. Hart, P.E.; Nilsson, N.J.; Raphael, B. A formal basis for the heuristic determination of minimum cost paths. *IEEE Trans. Syst. Sci. Cybern.* **1968**, *4*, 100–107. [CrossRef]
23. Souissi, O.; Benatitallah, R.; Duvivier, D.; Artiba, A.; Belanger, N.; Feyzeau, P. Path planning: A 2013 survey. In Proceedings of the 2013 International Conference on Industrial Engineering and Systems Management (IESM), Agdal, Morocco, 28–30 October 2013; pp. 1–8.
24. Bohlin, R.; Kavraki, L.E. Path planning using lazy PRM. In Proceedings of the 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings, San Francisco, CA, USA, 24–28 April 2000; Volume 1, pp. 521–528.
25. Karaman, S.; Walter, M.R.; Perez, A.; Frazzoli, E.; Teller, S. Anytime motion planning using the RRT. In Proceedings of the 2011 IEEE International Conference on Robotics and Automation, Shanghai, China, 9–13 May 2011; pp. 1478–1483.
26. Noreen, I.; Khan, A.; Habib, Z. A review of path smoothness approaches for non-holonomic mobile robots. In *2018 Computing Conference*; Springer International Publishing: Cham, Switzerland, 2018; pp. 346–358.
27. Dubins, L.E. On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents. *Am. J. Math.* **1957**, *79*, 497–516. [CrossRef]
28. Khan, A.; Noreen, I.; Habib, Z. On Complete Coverage Path Planning Algorithms for Non-holonomic Mobile Robots: Survey and Challenges. *J. Inf. Sci. Eng.* **2017**, *33*, 101–121.
29. Gu, W.; Cai, S.; Hu, Y.; Zhang, H.; Chen, H. Trajectory planning and tracking control of a ground mobile robot: A reconstruction approach towards space vehicle. *ISA Trans.* **2019**, *87*, 116–128. [CrossRef]
30. De Boor, C. On calculating with B-splines. *J. Approx. Theory* **1972**, *6*, 50–62. [CrossRef]
31. Han, X.A.; Ma, Y.; Huang, X. A novel generalization of Bézier curve and surface. *J. Comput. Appl. Math.* **2008**, *217*, 180–193. [CrossRef]
32. Nguyen, A.D.; Tran, N.H.; Nguyen, T.T.; Nguyen, A.T.; Tran, T.P. A Hybrid Multi-waypoints Path Planning System for Robots with Minimum Turning Radius Constraint Using GA-B-Spline and Dubins Interpolation. In *International Conference on Advanced Mechanical Engineering, Automation and Sustainable Development*; Springer International Publishing: Cham, Switzerland, 2021; pp. 906–917.
33. Piegl, L.; Tiller, W. *The NURBS Book*; Springer Science & Business Media: New York, NY, USA, 2012.
34. Shanmugavel, M. *Path Planning of Multiple Autonomous Vehicles*; Cranfield University: Silsoe, UK, 2007.
35. Trélat, E. Optimal control and applications to aerospace: Some results and challenges. *J. Optim. Theory Appl.* **2012**, *157*, 713–758. [CrossRef]
36. Faulwasser, T. *Optimization-Based Solutions to Constrained Trajectory-Tracking and Path-Following Problems*; Shaker: Aachen, Germany, 2013.
37. Wang, X. *Solving Optimal Control Problems with MATLAB: Indirect Methods*; ISE Dept., NCSU: Raleigh, NC, USA, 2009; p. 27695.
38. Sager, S.; Bock, H.G.; Reinelt, G. Direct methods with maximal lower bound for mixed-integer optimal control problems. *Math. Program.* **2009**, *118*, 109–149. [CrossRef]
39. Kelly, M. Cannon Example. Available online: http://www.matthewpeterkelly.com/tutorials/trajectoryOptimization/canon.html (accessed on 6 November 2017).
40. Karaman, S.; Frazzoli, E. Sampling-based algorithms for optimal motion planning. *Int. J. Robot. Res.* **2011**, *30*, 846–894. [CrossRef]