Computing optimal trajectories for light soaring aircraft using Fast Marching Tree Star

Lucas Bonin, Andreas Guitart, Daniel Delahaye, Xavier Prats, Eric Feron

Computing optimal trajectories for light soaring aircraft using Fast Marching Tree Star

Lucas Bonin[1*], Daniel Delahaye[1], Andreas Guitart[1], Xavier Prats[2] and Eric Feron[3]

[1*]OPTIM, Ecole Nationale de L'Aviation Civile, Toulouse, Occitanie, France.
[2]Department of Physics - Aerospace division, Universitat Politècnica de Catalunya, Castelldefels, Spain.
[3]RISC, King Abdullah University of Science and Technology, Saudi Arabia.

*Corresponding author(s). E-mail(s): bonin.lucas.74@gmail.com;
Contributing authors: delahaye@recherche.enac.fr;
andreas.guitart@enac.fr; xavier.prats@upc.edu;
eric.feron@kaust.edu.sa;

Abstract

In the last couple of years, performances of light soaring aircraft (para gliders, hang gliders, or light sailplanes) have increased significantly, allowing pilots to fly great distances using only the convective energy of the atmosphere. This activity, called "Cross-Country flying" or "soaring", requires topological and aerological knowledge and a lot of pre-flight preparation in order to make the right decisions, thus maximizing the flying distance or minimizing the flying time in a race. To optimize these flights, a pre-tactical decision support tool has been developed. This tool results from the adaptation of the Fast Marching Tree Star algorithm. It is extended in such a way it can deal with differential constraints associated with light soaring aircraft, operating in a convective atmosphere, in a field of wind, and close to reliefs. The method has been tested and validated on a real competition light soaring aircraft.

Keywords: Motion planning, trajectory Design, soaring aircraft, anisotropy

1 Introduction

For the last few years, ultralight gliders have evolved significantly making them more efficient and accessible. Current performances allow pilots to fly large distances without engines, using only the convective energy of the atmosphere. This activity, called "Cross-Country (XC) Flying", is mainly based on flight trajectories that join several thermals (i.e., columns of air with significant upwind components). Spiral-like trajectories are flown inside thermals in order to take enough altitude and then glide towards the next thermal area, repeating this process in order to travel long distances (Fig. 1).

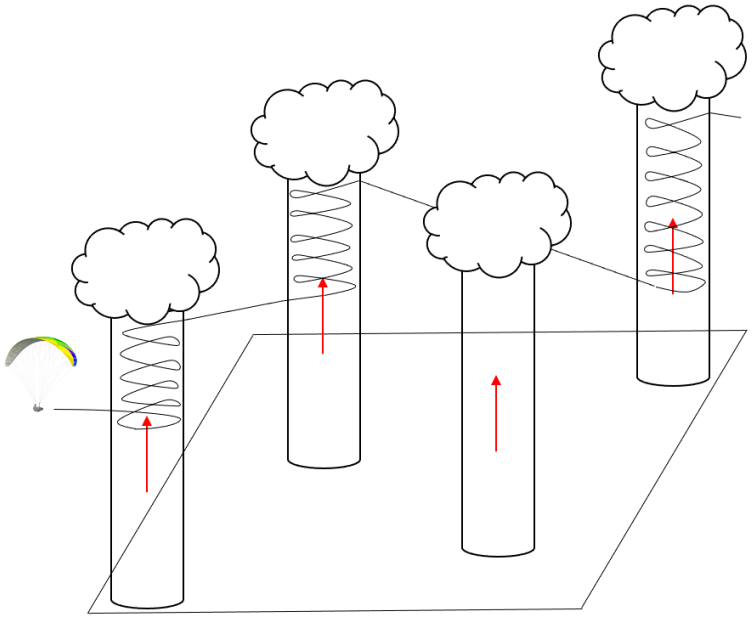

Fig. 1: Cross-Country (XC) flying

Year after year, more and more XC competitions take place. All competitors are equipped with cutting-edge technology and it has become very challenging to win. At present, there are two ways to compete in XC:

- to fly the longest distance (Open distance).
- to fly the first to finish (Race to goal).

When competing in Race to goal, pilots are supposed to fly all the way through specified control points represented by virtual cylinders in the air. Both competitions require very specific flying skills, along with an accurate knowledge of meteorological conditions.

In order to give the pilot an idea of what could be the optimal path to follow for such a competition, a pre-tactical decision support tool is proposed

in this paper. The tool is based on a modified version of the Fast Marching Tree Star (FMT*) algorithm. This sampling-based method takes into account the wind forecast of the competition day and a probability map of thermals [1]. However, the model presented does not take into account the flight dynamics of the ultralight glider and considers weather conditions as constant over time.

The main contribution of this paper is to model the path planning problem for XC flying as an optimal kinodynamic motion planning problem. The second contribution of this paper is to provide an extension of the FMT* algorithm to deal with high anisotropic environments and nonlinear differential motion constraints. Usually, the drawback of adding differential motion equations in a sampling-based path planning algorithm is the need for a two-point boundary value problem (TPBV) solver that computes a near optimal control input to steer the vehicles between specified initial and final states. As a way to get around, the TPBV has been reduced to either a quadratic or non-linear equation. This paper is organized as follows: Section 2 presents a state of the art of path planning algorithm and trajectory optimization for soaring flight. Section 3 details the problem that is solved. Section 4 describes the proposed algorithm. Finally, simulation results on a competition paraglider are presented in Section 5.

2 Previous Related Works

The main objective of motion planning problems is to find a sequence of valid configurations that drives an object (i.e., vehicle) from an origin to a destination point while avoiding collision with obstacles and minimizing a given cost function (time, distance,...). Several approaches and methods have been developed to address this problem. The most used ones are outlined below.

2.1 Discretization of the space

One approach is to discretize the research space in the form of a graph or a tree and use this structure to find a feasible path.

2.1.1 Deterministic algorithms

The first category of discreet methods are deterministic algorithms, where the network is fixed and therefore two same sets of initial conditions generate the same solution. The solution is limited because it depends on the graph. A larger problem or a continuous research space can lead to a high number of nodes and thus, an important computational time. Dijkstra's algorithm, introduced by Edsger Dijkstra in 1959 [2], is a classical way to find the shortest path in a graph with positive edge values. For a given starting node, the algorithm will compute the minimum cost path to any node of the graph. Using a priority queue, the time complexity is $O(a + n.log(n))$ with a the number of edges and n the number of nodes. The A* algorithm is an informed path search algorithm [3], often used when planning the path of a robot. It is close to Dijkstra's

method, where a heuristic function h is added and steers the path toward the goal. The cost function is defined as follows:

$$f(n) = g(n) + h(n) \tag{1}$$

where $g(n)$ the cost of the path from the start and $h(n)$, an estimation of the cost from the node n to the goal. At each iteration, the node n that minimizes f is added, in such a way it extends the path all the way to the goal. Since the heuristic function must be admissible in order to ensure its optimality, this is not an easy task for a complex problem.

2.1.2 Probabilistic algorithms

The other category of discreet algorithms is the probabilistic one. Instead of using an explicit representation of the space, a set of points is randomly sampled in the free space. Then, a research tree is built to explore the space and to find a feasible trajectory. Compared to deterministic methods, these algorithms are less costly from a computational time point of view and more adapted to high dimensional space. Sampling-based methods are a family of methods that includes multi-query algorithms, such as *Probabilistic Road Map (PRM)*; and single query methods, such as *Rapidly Exploring Random Tree (RRT)* [4]. It also includes their asymptotically optimal equivalents PRM* and RRT*. A state-of-the-art sampling-based algorithm is the Fast Marching Tree (FMT) algorithm [5]. It is asymptotically optimal and converges significantly faster than its counterpart path planning algorithms, justifying its use in the article. In the PRM, a random graph is computed in order to represent the research space, and then, a classical path searching method is applied to this graph. In the RRT and FMT algorithms, the rationale is different: the sampling is used to build a research tree iteratively, from the start to the goal. An extension of the sampling-based algorithms RRT* and FMT* have been developed in [6] and in [7] to handle systems described by differential constraints. These methods have been applied to some aeronautic problems such as obstacle avoidance trajectory design in [8], or more recently to emergency trajectory generation in [9, 10].

2.2 Optimal control

Another approach to answering the motion planning problem is to solve the associated optimal control problem (OCP). The goal is to compute, for any time, the optimal input for a system in order to minimize an objective function, while taking into account the dynamics of the system by considering, as constraints of the optimization, a set of differential equations. Methods to solve OCP are based on either direct or indirect methods, among which, level-set methods, collocation, and multiple shooting. One way to solve the OCP is to transcript it into a non linear problem and solve it using standard non linear programming (NLP) solvers [11]. Another way to solve OCP is to make

an analogy with the propagation of a wave. Since a wave tends to propagate along the path whose traveling time is shortest. By simulating the propagation of a wave, an optimal path can be determined, based on a criterion that has to be optimized.

Fast Marching The *Fast Marching* algorithm presented in [12] is part of more general method called level set algorithm. It determines the minimum time to reach any point of the space from a starting point. This method can be applied in the case of an isotropic environment. It can also be used in a low anisotropic environment.

Ordered Upwind When the propagation of the front depends on the position and on the direction of motion, the space is not considered isotropic. It is the case under a wind field if the wind speed has the same order of magnitude as the vehicle's speed. The *Ordered Upwind* algorithm has been developed in [13] to overcome this problem. However, it is more costly compared to the Fast Marching algorithm.

2.3 Trajectory optimization for soaring aircraft

One of the first research on the optimization of XC flying was made by Paul MacCready in 1954 [14]. He introduced the speed to fly as the optimal airspeed to adopt between two thermals making some assumptions on the next thermal vertical speed. Very popular for sailplanes, this method is less common in paragliding since pilots do not typically have an airspeed indicator. Also, this method does not allow an overall optimization of the path, since it considers each leg (from thermal to thermal) independently. Since the introduction of the MacCready speed to fly, several ways have been studied to optimize the trajectory of soaring vehicles in the atmosphere:

- Integrate the dynamic equation of the glider from a starting point with a set of control [15].
- Consider the Hamilton-Jacobi-Bellman equation for the optimal speed to fly [16].
- Solve the optimal control problem of minimizing the altitude loss of a glider maneuvering in still air[17].

These above approaches deal with path planning and some of them make assumptions on the wind field that surround the aircraft. However, the possibility to optimize the trajectory without assuming the characteristics of the aerological environment have been studied in [18] or in [19]. The method used in [18] provides autonomous soaring for a small glider using Q-learning, a Reinforcement Learning algorithm. However, since the paraglider dynamic is complex, the Q-learning algorithm is not adapted. In [19], the glider uses a visual sensor to detect clouds and deduce navigation and guidance.

Following this state of the art, FMT* seems very promising to solve the studied problem. Indeed, it is asymptotically optimal and its time complexity is low. Moreover, it can be adaptable to the problem of soaring flight trajectory optimization.

3 Problem Setup

This section formulates the problem to be solved and details the different models that are considered.

3.1 Atmospheric model

The air vehicle evolves in a convective atmosphere and in a wind field which are supposed constant over time for a given 3D location. The two are set as one 3D vector field $W : \mathbb{R}^3 \to \mathbb{R}^3$, the updraft being modeled by the vertical wind speed as follows:

$$(x, y, z) \mapsto (W_x(x, y, z), W_y(x, y, z), W_z(x, y)) \tag{2}$$

The longitudinal components W_x, W_y are computed using a weather forecast. The vertical component W_z can be determined in two ways. The first one is to use a weather forecast with a high resolution mesh grid, like the one generated with WRF (Weather Research and Forecasting tool), which directly gives the vertical velocity. The other way is to use a thermal map [1] (See Fig. 2), where the position of the center of thermals are indicated and noted $(x_i, y_i), \forall i \in N$. To compute W_z from this thermal map, each thermal is modeled as 2D Gaussian independent of the z coordinate and the maximum altitude of a thermal is limited by the cloud ceiling noted z_{max}. Thus, for N thermals the vertical component is written as:

$$\forall (x, y) \in \mathbb{R}^2, W_z(x, y) = \begin{cases} A \sum_{i=1}^{N} exp(-(\frac{(x-x_i)^2}{2\sigma^2} + \frac{(y-y_i)^2}{2\sigma^2})) & \text{if } z \leq z_{max} \\ 0 & \text{if } z > z_{max} \end{cases} \tag{3}$$

where σ is the order of magnitude of the radius of a thermal and A is the maximum velocity inside thermals (see Fig. 3).

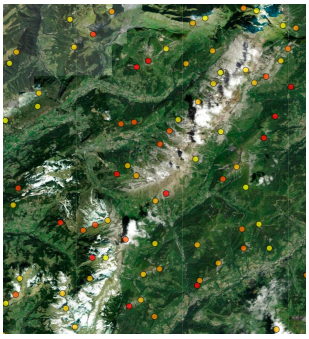

Fig. 2: Thermal map, dots represent position of thermals

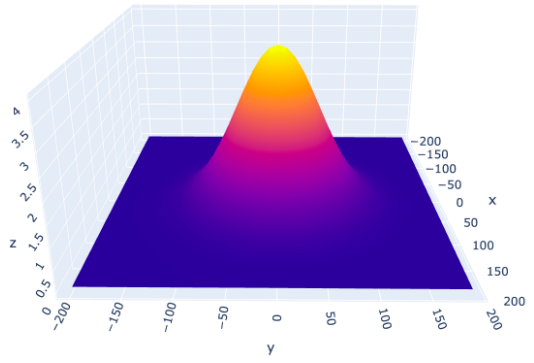

Fig. 3: Gaussian model of a thermal

3.2 Terrain

The glider evolves over the ground and should avoid collision with it. Thus, let us define the terrain as a smooth surface $z_t : \mathbb{R}^2 \to \mathbb{R}$ that is computed by real data points (see Fig. 4).

3.3 Soaring aircraft kinematics

During the flight, the pilot can adjust the following control variables (see Fig. 5 and Fig. 6):

- the heading $\psi \in [0, 2\pi[$;
- the airspeed $V_a \in \mathbb{R}^3$, in order to modify the glide path angle $\gamma \in [-\pi, \pi]$.

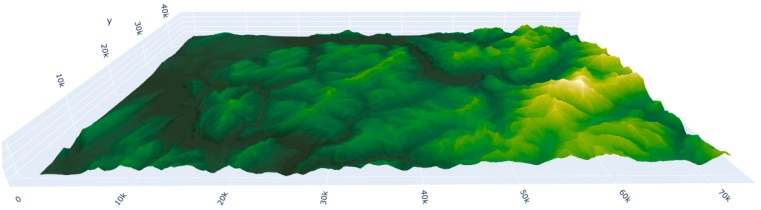

Fig. 4: Terrain surface

Let v_h and v_v be the horizontal airspeed and the vertical airspeed of the glider such that:

$$V_a = (v_h \cos\psi, v_h \sin\psi, v_v) \tag{4}$$

The airspeed, the ground speed, and the wind field are linked through the following differential equations which describe the motion of the aircraft in space:

$$\begin{cases} \dot{x}(t) = v_h(t)\cos\psi(t) + W_x(x,y,z) \\ \dot{y}(t) = v_h(t)\sin\psi(t) + W_y(x,y,z) \\ \dot{z}(t) = v_z(t) + W_z(x,y) \end{cases} \tag{5}$$

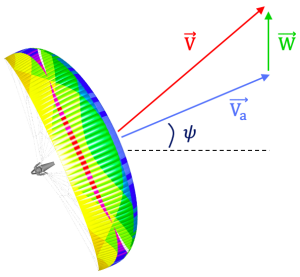

Fig. 5: Heading control

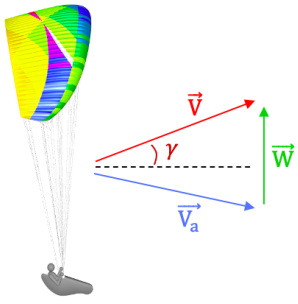

Fig. 6: Glide path angle control

where $(x(t), y(t), z(t))$ is the position at t of the aircraft in a Cartesian frame. The real kinematic behavior of a light glider is complex and depends on its nature (paraglider, hang glider, or sailplane). The model described here is simplified and the following assumptions are made:

Assumption 1. *Apparent mass or inertia effect are not considered.* This means that the vehicle is always in a steady-state condition.

Assumption 2. *The airspeed of the glider is described by its speed polar.* This means that at any time, the vertical speed v_v and the horizontal speed v_h are linked by the polar curve (see Fig. 7). This mean that $v_v = p(v_h)$, where the function p can be considered as a quadratic function ($\forall x \in \mathbb{R}^+, p(x) = k_2x^2 + k_1x + k_0$) or a linear one ($\forall x \in \mathbb{R}^+, p(x) = l_1x + l_0$) and $v_{h_{min}} \leq v_h \leq v_{h_{max}}$. In still air, the glider is always in descent (the flight path angle is always

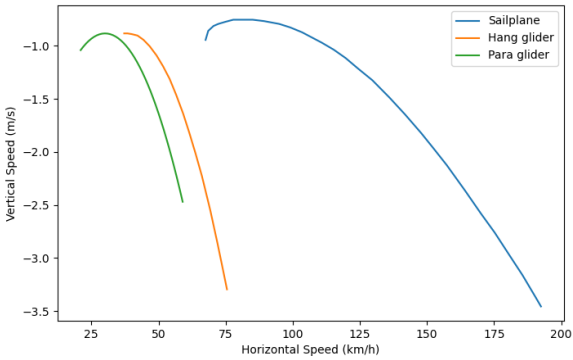

Fig. 7: Speed polar of soaring aircraft

negative). The local wind field due to a thermal will affect the ground glide path angle, allowing the glider to gain altitude in the case of sufficient upwind.

Assumptions 3. *The speed polar is independent of the turn rate (may be true for a shallow turn).*

3.4 Optimal motion planning problem

Finding the minimum time trajectory between two points in XC competitions can be expressed as an optimization problem as follows:

- **Optimality Criterion:** Minimize the flying time from the starting point (x_0, y_0, z_0) to the goal point (x_f, y_f, z_f);
- **State Variables:** Position of the aircraft at the time instant t : $(x(t), y(t), z(t))$;
- **Control Variables :** At time t, the horizontal speed $v_h(t)$ and the heading $\psi(t)$. $\forall t > 0, v_h(t) \in [v_{h_{min}}, v_{h_{max}}], \psi(t) \in [0, 2\pi]$;
- **Constraints:** Respect Eq. (5), the speed polar, and avoid collision with the terrain (i.e., the height of the glider is always strictly positive).

Mathematically, this problem can be written as:

$$\begin{array}{ll} \min_{\psi(.),v_h(.)} & \int_0^T \mathrm{d}t = T \\ \text{subject to:} & \dot{x}(t) = v_h(t)\cos\psi(t) + W_x(x(t),y(t),z(t)) \\ & \dot{y}(t) = v_h(t)\sin\psi(t) + W_y(x(t),y(t),z(t)) \\ & \dot{z}(t) = p(v_h(t)) + W_z(x(t),y(t)) \\ & v_{h_{min}} \leq v_h(t) \leq v_{h_{max}} \\ & z_t(x(t),y(t)) < z(t) \\ & (x(0),y(0),z(0)) = (x_0,y_0,z_0) \\ & (x(T),y(T),z(T)) = (x_f,y_f,z_f) \end{array} \tag{6}$$

This optimization problem can be expressed in the general form of an optimal kinodynamic motion planning problem as the one presented in [6]. Let the state space $\chi \subset \mathbb{R}^3$ and the control space $U \subset \mathbb{R}^2$ be compact sets such that:

$$\begin{cases} \forall X(t) \in \chi & X(t) = (x(t),y(t),z(t)) \\ \forall u(t) \in U & u(t) = (v_h(t),\psi(t)) \end{cases} \tag{7}$$

Let the subset $\chi_{obs} \subset \chi$ be the *obstacle region* such that χ/χ_{obs} is an open set and $\chi_{free} = \chi/\chi_{obs}$ define the *obstacle-free region* such that:

$$\forall X(t) \in \chi_{free} \iff z_t(x(t),y(t)) < z(t) \tag{8}$$

Let $\chi_{goal} \subset \chi$ be the *goal region* and the cost functional:

$$\begin{array}{rccc} g: & \chi & \to & \mathbb{R}^+ \\ & X(t) & \mapsto & g(X(t)) = 1 \end{array} \tag{9}$$

Finally, let $\mathbb{X}$ and $\mathbb{U}$ be the sets of all essentially bounded measurable functions defined $\forall T \in \mathbb{R}^{+*}$ from $[0,T]$ to χ and U, respectively. Then, the problem writes as follows [6]: *Given the state space χ, the goal region χ_{goal}, the obstacle-free region χ_{obs}, an initial state $(x_0,y_0,z_0) \in \chi_{free}$ and a differential equation that describe the system dynamics. Find a control $u \in \mathbb{U}$ with domain $[0,T]$ for $T \in \mathbb{R}^{+*}$ such that the trajectory $X \in \mathbb{X}$:*

- verifies $\dot{X} = f(X(t),u(t)), \forall t \in [0,T]$;
- avoids obstacles, $X(t) \in \chi_{free}, \forall t \in [0,T]$;
- reaches the goal region, $X(T) \in \chi_{goal}$;
- minimizes the cost function $J(X) = \int_0^T g(X(t))\mathrm{d}t$.

4 Problem Resolution

The algorithm used to solve the problem presented above is an adaptation of the sampling-based algorithm *FMT** [5]. The process of this algorithm is detailed below.

4.1 FMT* Algorithm

The environment is at first (randomly) sampled with N points or nodes. Then, FMT* explores each node to compute the optimal path from the origin to the goal (see Figs. 8 and 9).

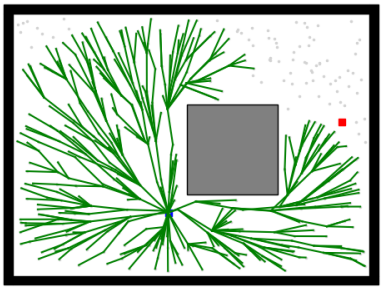

Fig. 8: Tree constructed by FMT* is exploring the space

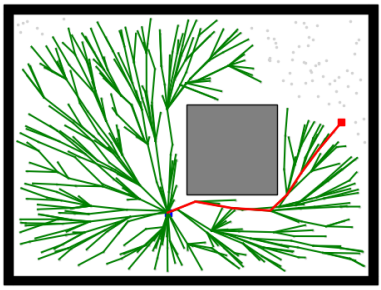

Fig. 9: Once the goal is found, the optimal path is retrieved

For this, at each iteration, nodes are split into 3 sets:

- $V_{unvisited}$: nodes that have never been visited by the FMT* (in green in the following figures);
- V_{open}: nodes that have already been visited but for which the cost value is not definitively computed (in orange in the following figures);
- V_{closed}: nodes that have already been visited but for which the cost value is definitively computed (in red in the following figures).

The local optimization process works as follows (see Fig. 10) :

1. The lowest-cost open node z is selected and all its unvisited neighbors are considered. x_{init} is the starting point of the trajectory, and the root node of the tree.
2. For each unvisited neighbor x of z, all x open neighbors are considered and an edge between x and one of these neighbors is added to make this connection locally-optimal (without considering any constraint).
3. If this connection does not violate constraints or is collision-free, it is added to the tree and x is removed from the set $V_{unvisited}$ and added to V_{open}.
4. Once all z unvisited neighbors have been visited, z is put in V_{closed} and the iteration is done.

FMT* (for which the pseudocode is given in the Appendix) is extended in order to take into account the high anisotropic space. In the classical FMT*, new tree nodes are added in the set named $V_{open,new}$, and this set is added to a global set V_{open} afterward. These steps do not allow radical changes in the direction of propagation. A way to get around this is to add new tree nodes directly in the V_{open} set.

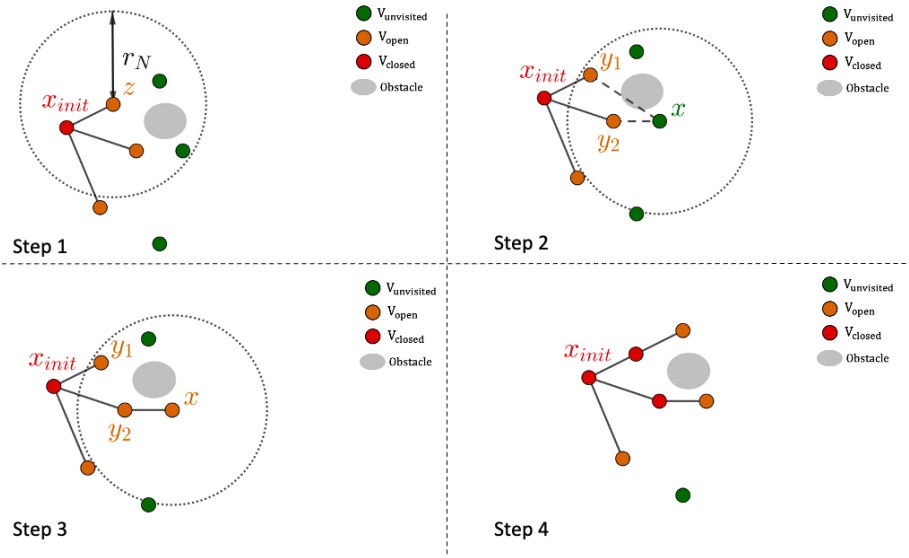

Fig. 10: One iteration of the FMT* algorithm

4.2 Implementation details

4.2.1 Non Uniform Sampling Strategy

During a flight, the behavior of the glider changes. When it comes to gliding between thermals, the flight regime is constant and the trajectory is composed of long segments. On the other hand, during the climb, the aircraft makes narrow spirals. To account for this, thermal areas are sampled with more density than the free space as illustrated in Fig. 11.

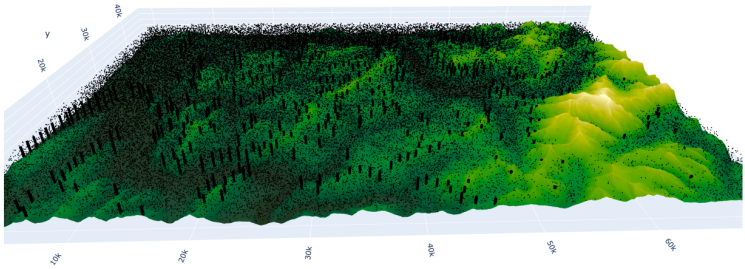

Fig. 11: Non-uniform sampling of the 3D space

To ensure asymptotic optimality, the neighbour radius r_n is defined as follows [5]:

$$r_n = 2(1+\eta)\left(\frac{1}{dl}\right)^{\frac{1}{d}}\left(\frac{\mu(\chi_{free})}{\zeta_d}\right)^{\frac{1}{d}}\left(\frac{log(n)}{n}\right)^{\frac{1}{d}} \tag{10}$$

where:

- l is the strictly positive upper bound of the probability density function supported over χ_{free}.
- $\eta > 0$ is a tuning parameter that depends on the nature of the problem. In this case, the kinematic constraint is binding, the covered neighbors are few compared to the classical case. Thus, η must compensate for this added constraint.
- μ is the Lebesgue measure and ζ_d is the volume of the unit ball in d-dimension.

4.2.2 Trajectory and Cost computation

In the FMT* algorithm, for two samples $(Z_1, Z_2) \in \chi^2_{free}$ the function $Cost(Z_1, Z_2)$ returns the cost of the optimal trajectory that starts from Z_1 and reaches Z_2, when such a trajectory exists. Computing such an optimal trajectory, for a line integral cost, for every connection, can be computationally challenging. Indeed, it means solving a two-point boundary value problem for each connection. Thus, to compute the cost between two states Z_1 and Z_2, the optimal path is approximated with a cost-weighted, **straight line connection** which respects the differential constraint. Along the straight line $\overline{Z_1 Z_2}$, the control is supposed constant and thus the solution for the control u is piecewise-constant.

$$Z_2 = Z_1 + \int_{t_1}^{t_2} \dot{X}(t).dt = Z_1 + \int_{t_1}^{t_2} f(X(t), \overline{u}).dt \tag{11}$$

The integral is computed using a repeated midpoint rule. The interval $[t_1, t_2]$ is partitioned into m equally spaced points and $h = t_2 - t_1$. Eq. (11) can be rewritten as follows:

$$Z_2 = Z_1 + \frac{h}{m}\sum_{i=1}^{m} f(Z_1 + \frac{i(Z_2 - Z_1)}{m+1}, \overline{u}) \tag{12}$$

The straight line connection $\overline{Z_1 Z_2}$ is feasible with respect to the differential equation if and only if $\exists\ h > 0$ and $\exists\ \overline{u} \in U$ verifying (Eq. (12)). Thus,

computing the cost h is equivalent to solving the following system of equations:

$$\begin{cases} x_2 = x_1 + h\overline{v_h}\cos\overline{\psi} + \frac{h}{m}\sum_{i=1}^{m} W_x(x_{mi}, y_{mi}, z_{mi}) \\ y_2 = y_1 + h\overline{v_h}\sin\overline{\psi} + \frac{h}{m}\sum_{i=1}^{m} W_y(x_{mi}, y_{mi}, z_{mi}) \\ z_2 = z_1 + hp(\overline{v_h}) + \frac{h}{m}\sum_{i=1}^{m} W_z(x_{mi}, y_{mi}, z_{mi}) \end{cases} \tag{13}$$

with p the *speed polar function* and:

$$\forall i \in [1, m], \begin{cases} x_{mi} = x_1 + \frac{i(x_2 - x_1)}{m+1} \\ y_{mi} = y_1 + \frac{i(y_2 - y_1)}{m+1} \\ z_{mi} = z_1 + \frac{i(z_2 - z_1)}{m+1} \end{cases} \tag{14}$$

For the sake of simplicity Eq. (13) is written :

$$\begin{cases} \mathrm{d}x = h(\overline{v_h}\cos\overline{\psi} + w_{xm}) \\ \mathrm{d}y = h(\overline{v_h}\sin\overline{\psi} + w_{ym}) \\ \mathrm{d}z = h(p(\overline{v_h}) + w_{zm}) \end{cases} \tag{15}$$

4.2.3 Reachable set

Exact estimation of the reachable set is also a computationally intensive task. However, the reachable set can be estimated. For a set V and a state Z, $Reachable(V, Z)$ returns the subset $\Gamma \subset V$ such that $\forall Z' \in \Gamma, \exists h > 0, \exists \overline{v_h} \in [v_{h_{min}}, v_{h_{max}}], \exists \overline{\psi} \in [0, 2\pi[$ solution of *Eq.* (15).

4.2.4 Cost functional with linear polar

In the case of linear speed polar $p(x) = l_1 x + l_0$, solve Eq. (15) is equivalent to solve the quadratic equation :

$$ah^2 + bh + c = 0 \tag{16}$$

with:

$$\begin{cases} a = l_1^2({w_{xm}}^2 + {w_{ym}}^2) - (l_0 + w_{zm})^2 \\ b = 2(\mathrm{d}z(l_0 + w_{zm}) - l_1^2(\mathrm{d}x w_{xm} + \mathrm{d}y w_{ym})) \\ c = l_1^2(\mathrm{d}x^2 + \mathrm{d}y^2) - \mathrm{d}z^2 \end{cases} \tag{17}$$

Thus, for two states Z_1, Z_2, $Cost(Z_1, Z_2)$ return the solution $h > 0$ of (Eq. (16)) which verifies:

$$\overline{v_h} = \frac{1}{l_1}\left(\frac{dz}{h} - (l_0 + w_{zm})\right) \in [v_{h_{min}}, v_{h_{max}}]$$

4.2.5 Cost functional with quadratic polar

In the case of quadratic speed polar $p(x) = k_2x^2 + k_1x + k_0$, the system (Eq. (15)) can be reduced to the following equation:

$$(\frac{\mathrm{d}x}{\mathrm{d}z}(p(\overline{v_h}) + w_{zm}) - w_{xm})^2 + (\frac{\mathrm{d}y}{\mathrm{d}z}(p(\overline{v_h}) + w_{zm}) - w_{ym})^2 = \overline{v_h}^2 \tag{18}$$

which can be solved by using the bisection method. Thus, for two states Z_1, Z_2, $Cost(Z_1, Z_2)$ returns the solution $\overline{v_h} \in [v_{h_{min}}, v_{h_{max}}]$ of (Eq. (18)) which gives the minimum $h > 0$.

4.2.6 Illustration of the modified FMT*

In order to highlight the modifications on the FMT*, the steps are illustrated (see Fig. 12) and are explained in this section.

Let us detail one iteration on a vertical example, the obstacle represented in grey can be assimilated as a mountain. The algorithm works in the same way in 3D.

1. The first step is the same as the classical FMT* (see Section 4.1)
2. For each unvisited neighbor x of z, all x open neighbors are considered and all connections between x and one of these neighbors are checked.
3. The retain connection is the one that: is feasible from a kinematic point of view (see Section 4.2.2), minimizes the flying time from the start, and is collision-free.
4. Once all z unvisited neighbors have been visited, z is put in V_{closed} and the iteration is done.

This process is continued until the goal is reached. Once it is done, the optimal trajectory can be recovered (bold trajectory in Fig. 12)

5 Results and Discussion

The algorithm presented above was implemented using Python and ran on a computer with a 1,8 GHz processor, 8 Gb of RAM, and a MacOS operating system. For the following experiments, the linear polar function representing a competition paraglider was chosen:

$$\begin{cases} v_v = -0.2104v_h + 1.0247 \\ v_{h_{max}} = 16m/s \\ v_{h_{min}} = 8m/s \end{cases} \tag{19}$$

Since the optimal solution of the real 3D problem is not known the algorithm is first validated on 2D scenarios, both on the longitudinal and the vertical axis. Then a real problem is solved and the results of the algorithm are compared with the execution of a real flight.

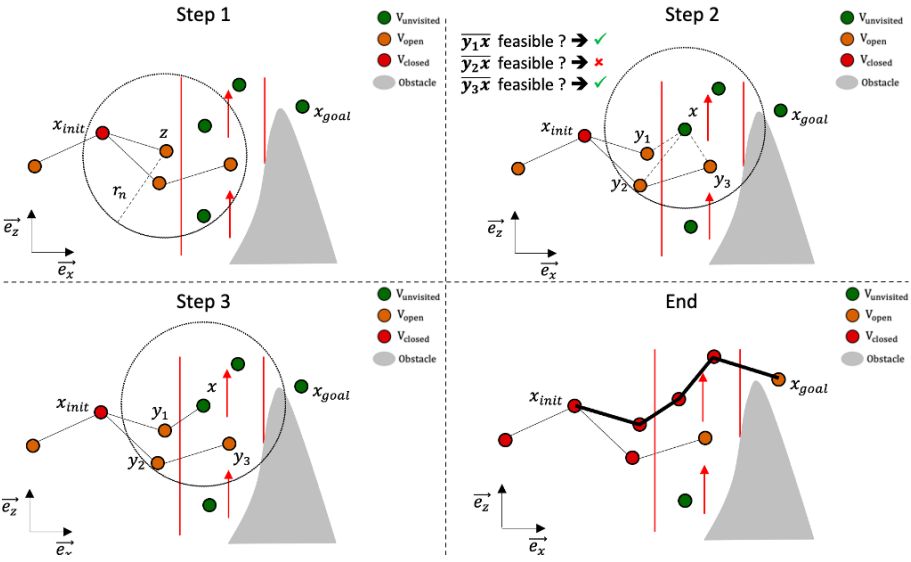

Fig. 12: Modified FMT*: One iteration and final trajectory

5.1 Longitudinal Validation: Zermelo's Navigation problem

The following problem is to plan a 2D trajectory in the horizontal plane from a starting point $A = (20000, -5000)$ to a final point $B = (0, 0)$ while minimizing the traveling time. In order to allow the glider to move horizontally, the upwind $W_z(x, y) = \overline{w_z}$ is set constant equal to 1.5 m/s. As $\mathrm{d}z = 0$, v_h is constant:

$$\overline{v_h} = -\frac{l_0 + w_{zm}}{l_1} = 12\ m/s$$

The horizontal wind is set as follows:

$$\begin{cases} W_x(x, y) = W_x(y) = -\frac{\overline{v_h} y}{h} \\ W_y(x, y) = 0 \end{cases} \tag{20}$$

with $h = 5000m$. This problem is known as Zermelo's navigation problem and in this particular case ($W_x(x, y) = f(y), W_y(x, y) = 0$), it admits an analytical solution. The solution is obtained by using Pontryagin's maximum principle. The optimum time solution for this problem is **1751.289 s**. Algorithm 1 presented in Section 4 was run with different uniform samplings and results were compared with the optimal (analytical) trajectory (see Fig. 13). As the number of samples increases, the computed trajectory with FMT* converges toward the optimum solution (see Fig. 14).

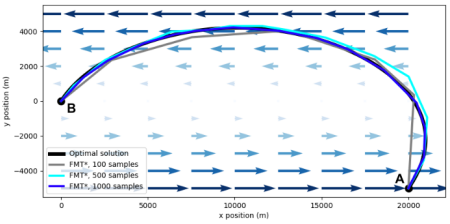

Fig. 13: Comparison of the computed trajectories with FMT* and the optimal trajectory, from A to B.

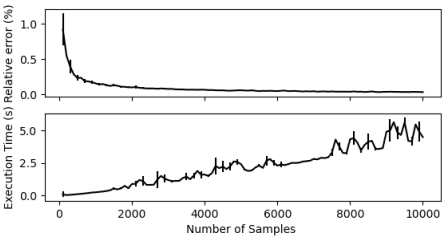

Fig. 14: Evolution of the solution with the number of samples.

5.2 Vertical Validation

Typical trajectories in the vertical plane in XC flying are represented in Fig. 16. The trajectory consists of a series of gliding phases and climbing phases within thermal areas. The structure of the minimum time trajectory depends highly on the position of thermal areas and their intensity. If thermals are weak, the pilot should fly at a low speed to decrease the sink rate and thus decrease the altitude loss (blue trajectory). On the contrary, with powerful thermals with high upwind speed, the pilot should fly at high speed because he will recover the loss of altitude quickly in the next updraft (black trajectory).

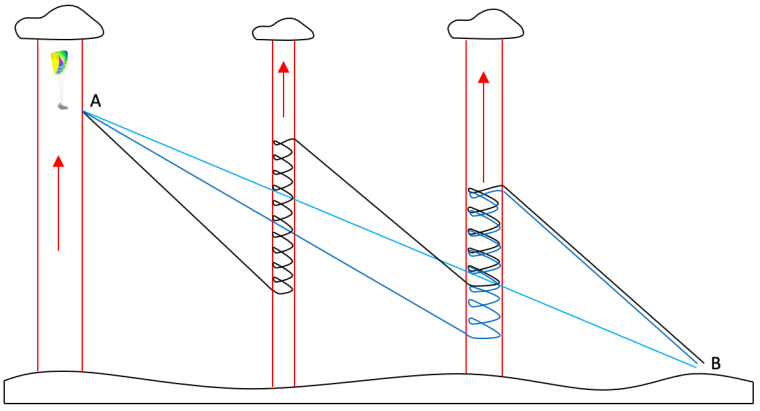

Fig. 15: Examples of vertical trajectories in XC flying from A to B

To verify that the algorithm approximates well the optimal trajectory on the vertical plane, the following scenario has been put in place. The problem is to plan the minimum time trajectory from a point $A = (0, 1000)$ to a point $B = (20000, 0)$ while going through different cylindrical thermal areas with a radius equal to 150m. The longitudinal wind is null. The positions of two thermals are [5000 m, 14000 m] with an upwind speed equal to 6 m/s. In order to access the optimal solution, the problem can be expressed as non linear

programming problem (NLP) where the decision variables are the altitude entrance and exit of each thermal and the speed v_h for each gliding segment. The speed v_h in thermal areas is constant and equal to the one that gives the minimum sink rate. Then, the optimal cost and trajectory are computed using a NLP solver. In this example, the optimal time is **1772.39 s**. To point out the importance of non uniform sampling in an anisotropic environment, the FMT* algorithm was run with different samplings. Thermal areas, noted $\chi_{thermal} \subset \chi_{free}$, are sampled with m samples and χ_{free} is sampled with n samples. Let us denote $\alpha = \frac{m}{n}$ the anisotropic factor of the sampling. Fig. 17 compares the computed trajectories with the optimal one.

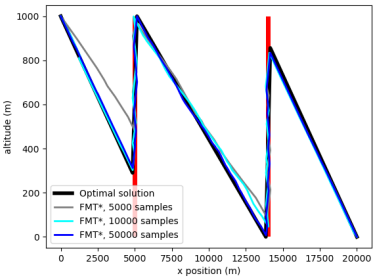

Fig. 16: Computed vertical trajectories

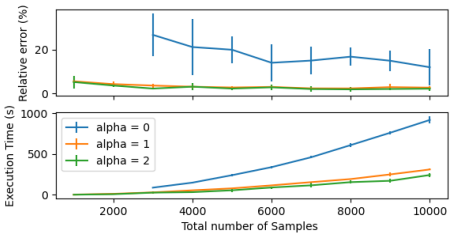

Fig. 17: Evolution of the solution with the number of samples.

When $\alpha = 0$, the sampling is uniform. In this case, a high number of samples and thus a high computational time is necessary to approach the optimal solution. When $\alpha > 0$, the sampling is non uniform, and the convergence is much faster. As pointed out above, the relative gap between the computed time and the optimal time is decreasing when the number of samples increases (see Fig. 17). Due to the fact that the environment is highly anisotropic, the number of samples has to be high to approximate well the solution. The previous two sections showed that the proposed method gives correct results on both axes, close to the optimal solution if the algorithm is set with a sufficient number of samples.

5.3 Comparison of real and computed trajectories

The following section deals with the real 3D scenario over the ground surface. Three different real trajectories (available at https://www.xcontest.org/world/en/) were compared to computed ones. The simulated trajectories show important gains in terms of flight time compared to the real ones (Table 1). However, computed trajectories rely on a theoretical thermal map and on wind forecast which can be far from reality. Comparison of trajectories produced by FMT* and real flown trajectories are shown in Figs. 18 to 20. Other figures can be found in Section 6.

Table 1: Cost comparison of computed trajectories with FMT* and real trajectories.

Id	Computed Solution (s)	Real trajectory (s)	Gain (%)
1	3345	4919	32
2	3524	5392	34.6
3	3266	3566	8.4

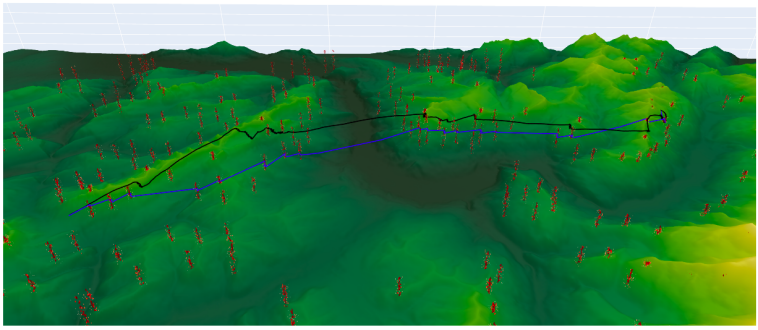

Fig. 18: Comparison between a computed (blue) and a flown trajectory (black) for the first scenario, red quivers represent thermals.

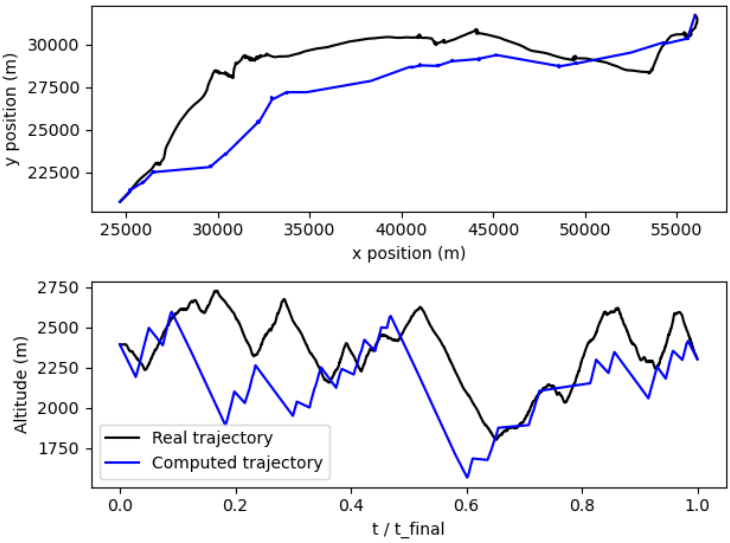

Fig. 19: Comparison of the longitudinal trajectories and the altitude profiles for the first scenario.

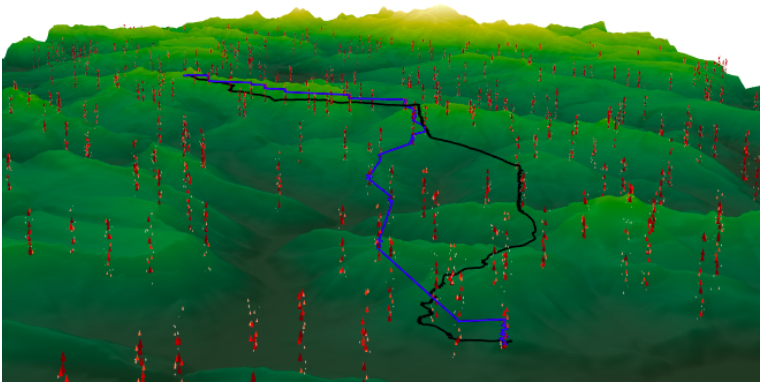

Fig. 20: Another comparison between a computed (blue) and a flown trajectory (black) for the third scenario, red quivers represent thermals.

6 Conclusion

In this paper, a new model of path planning for soaring aircraft has been presented as well as an anisotropic extension for the FMT* algorithm. Simulations point out that the method is well adapted to plan the path in a 3D wind field with complex kinematics motion while avoiding terrain collision. However, several points have to be improved. Because the quality of the solution depends highly on the accuracy of the probabilistic weather data (the wind and thermal map), a confidence index associated with each computed trajectory could be provided. Moreover, stochastic or robust approaches, as proposed in [20], could be an interesting approach to further explore. Any aircraft must respect the rules of the air and thus, avoid restricted airspace. These areas have to be added in the obstacle set. Finally, the weather (wind and thermals) depends on time, therefore, for trajectories longer than 2 hours, the time variability should also be taken into account. This study opens the way for the development of a future smartphone application to assist Cross-Country Flying competitors.

Appendix A

Pseudo code of the FMT* algorithm.

Algorithm 1 FMT* with anisotropic extension

$V \leftarrow \{x_{init}\} \cup SampleFree(n)$; $E \leftarrow \emptyset$
$V_{unvisited} \leftarrow V \backslash \{x_{init}\}$;$V_{open} \leftarrow \{x_{init}\}$;$V_{closed} \leftarrow \emptyset$
$z \leftarrow x_{init}$
while $z \notin \chi_{goal}$ **do**
    $N_z \leftarrow Near(V \backslash \{z\}, z, r)$
    $X_{near} \leftarrow N_z \cap V_{unvisited}$
    for $x \in X_{near}$ **do**
        $N_x \leftarrow Near(V \backslash \{x\}, x, r)$
        $\Gamma_x \leftarrow Reachable(V \backslash \{x\}, x)$
        $Y_{near} \leftarrow N_x \cap V_{open} \cap \Gamma_x$
        if $Y_{near} \neq \emptyset$ **then**
            $y_{min} \leftarrow argmin_{y \in Y_{near}} \{c(y) + Cost(y, x)\}$
            if $CollisionFree(y_{min}, x)$ **then**
                $E \leftarrow E \cup \{(y_{min}, x)\}$
                $V_{open} \leftarrow V_{open} \cup \{x\}$
                $V_{unvisited} \leftarrow V_{unvisited} \backslash \{x\}$
                $c(x) \leftarrow c(y_{min}) + Cost(y_{min}, x)$
            end if
        end if
    end for
    $V_{open} \leftarrow V_{open} \backslash \{z\}$
    $V_{closed} \leftarrow V_{closed} \cup \{z\}$
    $z \leftarrow argmin_{y \in V_{open}} \{c(y)\}$
end while
return $Path(z, T = (V_{open} \cup V_{closed}, E))$

Appendix B

More illustrations of the comparison between real and computed 3D trajectories.

References

[1] von Kanel, M.: Paraglidingnet: A sensor network for thermal research. Master's thesis, Swiss Federal Institute of Technology Zurich, vkaenemi@ee.ethz.ch (August 2010)

[2] Dijkstra, E.W., *et al.*: A note on two problems in connexion with graphs. Numerische mathematik **1**(1), 269–271 (1959)

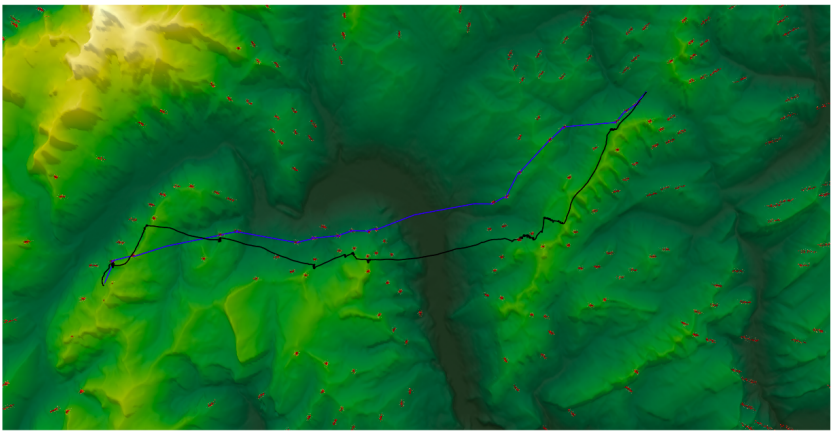

Fig. 21: Comparison between computed (blue) and a flown trajectory (black) for the 1st scenario, red quivers represent thermals.

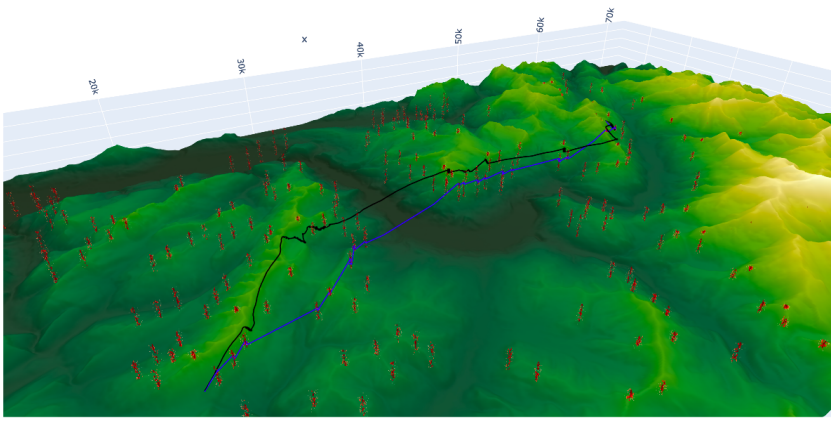

Fig. 22: Comparison between computed (blue) and a flown trajectory (black) for the 1st scenario, red quivers represent thermals.

[3] Hart, P.E., Nilsson, N.J., Raphael, B.: A formal basis for the heuristic determination of minimum cost paths. IEEE transactions on Systems Science and Cybernetics **4**(2), 100–107 (1968)

[4] Karaman, S., Frazzoli, E.: Sampling-based algorithms for optimal motion planning. The international journal of robotics research **30**(7), 846–894 (2011)

[5] Janson, L., Schmerling, E., Clark, A., Pavone, M.: Fast marching tree: A fast marching sampling-based method for optimal motion planning in many dimensions. The International journal of robotics research **34**(7),

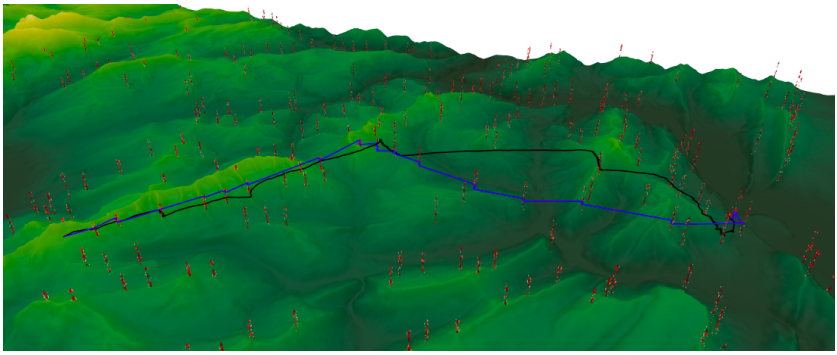

Fig. 23: Comparison between computed (blue) and a flown trajectory (black) for the 3rd scenario, red quivers represent thermals.

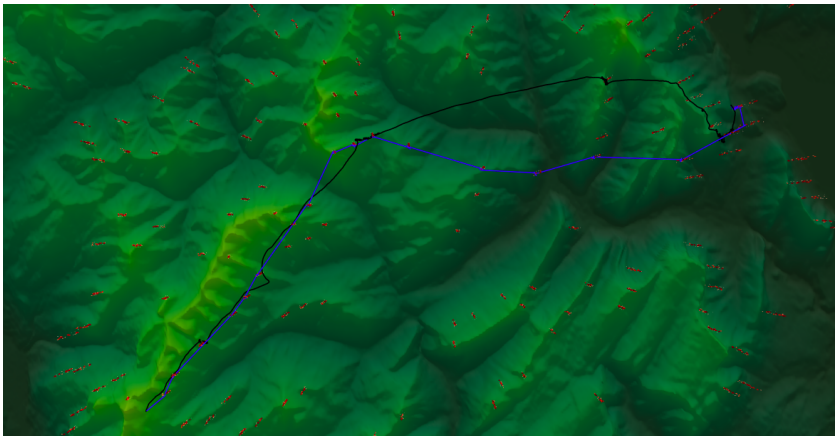

Fig. 24: Comparison between computed (blue) and a flown trajectory (black) for the 3rd scenario, red quivers represent thermals.

883–921 (2015)

[6] Karaman, S., Frazzoli, E.: Optimal kinodynamic motion planning using incremental sampling-based methods. In: 49th IEEE Conference on Decision and Control (CDC), pp. 7681–7687 (2010). IEEE

[7] Schmerling, E., Janson, L., Pavone, M.: Optimal sampling-based motion planning under differential constraints: the driftless case. In: 2015 IEEE International Conference on Robotics and Automation (ICRA), pp. 2368–2375 (2015). IEEE

[8] Pharpatara, P., Hérissé, B., Bestaoui, Y.: 3-D Trajectory planning of aerial vehicles using RRT*. IEEE Transactions on Control Systems Technology

25(3), 1116–1123 (2017)

[9] Guitart, A., Delahaye, D., Feron, E.: An accelerated dual fast marching tree applied to emergency geometric trajectory generation. Aerospace **9**(4) (2022)

[10] Sáez, R., Khaledian, H., Prats, X., Guitart, A., Delahaye, D., Feron, E.: A fast and flexible emergency trajectory generator - enhancing emergency geometric planning with aircraft dynamics. (2021)

[11] Betts, J.T.: Practical Methods for Optimal Control and Estimation Using Nonlinear Programming. SIAM

[12] Sethian, J.A.: A fast marching level set method for monotonically advancing fronts. Proceedings of the National Academy of Sciences **93**(4), 1591–1595 (1996)

[13] Sethian, J.A., Vladimirsky, A.: Ordered upwind methods for static hamilton–jacobi equations. Proceedings of the National Academy of Sciences **98**(20), 11069–11074 (2001)

[14] McCready, P.B.: An optimal airspeed selector. Soaring (1954)

[15] Lawrance, N.R.: Autonomous soaring flight for unmanned aerial vehicles (2011)

[16] Almgren, R., Tourin, A.: Optimal soaring via hamilton–jacobi–bellman equations. Optimal Control Applications and Methods **36**(4), 475–495 (2015)

[17] Wolek, A., Woolsey, C.: Optimal paths in still air for a sailplane with a quadratic glide polar. Technical Soaring **40**(2), 9–23

[18] Lecarpentier, E., Rapp, S., Melo, M., Rachelson, E.: Empirical evaluation of a q-learning algorithm for model-free autonomous soaring. arXiv preprint arXiv:1707.05668 (2017)

[19] Stolle, M.: Towards vision-based autonomous cross-country soaring for uavs. PhD thesis, UNIVERSITE DE TOULOUSE (2017)

[20] González-Arribas, D., Soler, M., Sanjurjo-Rivo, M.: Robust aircraft trajectory planning under wind uncertainty using optimal control. Journal of Guidance, Control, and Dynamics **41**(3), 673–688 (2018)