Safe synchronous path planning for a fleet of fixed-wing constant speed aircraft

Maël Feurgard, Gautier Hattenberger, Nicolas Durand[1,*] and Simon Lacroix[2,†]
Fédération ENAC ISAE-SUPAERO ONERA, Université de Toulouse, France[1]
RIS, Laboratoire d'Analyse et d'Architecture des Systèmes, Université de Toulouse, CNRS, Toulouse, France[2]

Abstract

Path planning for multiple unmanned aerial vehicles is a difficult task, and even more for a fleet of fixed-wing aircraft. One specific case is the transition to, or between, formation flight patterns, which requires synchronous arrivals while ensuring minimal separation, and ideally maintaining cruise speed. We present a centralized method to solve this problem based on enumerating different Dubins paths. Given a travel time for the fleet, it builds a set of possible paths for each aircraft. Then, it checks in parallel separation between each path pair. This yields coefficients for an Integer Linear Programming problem determining if a fleet-wide conflict-free solution exists. This process is repeated for different travel times sampled with increasing resolution until the user-defined accuracy is met. The method is benchmarked with Monte-Carlo simulations considering up to 20 aircraft simultaneously, achieving an 95% success rate for an average 8 seconds computation time. A showcase integration is done using three simulated aircraft within the PaparazziUAV simulator.

1 Introduction

Unmanned Aerial Vehicles (UAVs) are becoming increasingly affordable at relatively low prices, thereby encouraging the deployment of multiple autonomous aircraft simultaneously. In particular, a fleet of fixed-wing aircraft is useful for long missions, requiring significant payload or airspeed ; aerial cartography [1], airborne particles sensing [2] and forest monitoring [3] are examples of such tasks. They all require solving a common problem: cooperative path planning.

Cooperative path planning consists in moving multiple agents in a shared space to achieve a common goal. Several constraints can be considered, the most common being collision avoidance, synchronous arrival to destination, maintaining formation and feasible dynamics. These problems have been significantly studied for aerial vehicles [4, 5, 6]. We are particularly interested in collision-free 2D movements with synchronous arrivals, since such movements correspond to getting into a planar formation or switching between formations. Different strategies exist to achieve such transitions: discretize the airspace using a grid[7]or sets of lines [8] first, then generate safe flight paths and finally smooth them; use leader-follower or swarm schemes to handle collision avoidance dynamically, and vary speed for synchronization [9, 10, 11]; or directly generate valid paths.

We focus on this last category as it does not rely on a discretization parameter and provide an explicit path to be followed for each agent. This ensures that any solution is deadlock-free, but it usually requires more computing power and there is no guarantee that a solution will be found. Paths can be generated along several models. Some can consider the acceleration limitations of the aircraft, and will use clothoids [12], Bézier splines [13, 14] or Pythagorean Hodograph splines [15, 16, 17]. Others use the Dubins model [18], as it simplifies trajectory generation into a geometry problem with circles and lines [19, 20] at the cost of introducing acceleration discontinuities.

Among the full path generation methods, only a limited amount of aircraft are considered and performances are poorly studied. In [14], several Monte Carlo simulations are done, measuring accuracy and runtime of the planner, testing from 2 to 7 aircraft. But this article does not manage collisions. The other articles show only sample cases, without timing computations and the number of aircraft does not exceed 5. This is especially limited compared to the leader follower techniques, such as in [9] where 3 groups of 7 aircraft were deployed in real conditions.

We present a new method based on *Dubins paths* to solve the fixed-wing fleet path planning problem with the constraints of constant airspeed, collision avoidance and synchronous arrival in 2D. The main difference with [19, 20] is that instead of looking for one specific combination of curves in a continuous domain satisfying both synchronization and collision avoidance, we build sets of synchronous curves for each aircraft in which we look for a collision-free combination. This allows us to parallelize computations and push the benchmark from 3 to 20 aircraft simultaneously.

2 Problem formulation

We consider the problem of path planning for a fleet of fixed-wing aircraft flying at constant air speed and altitude, using simplified dynamics in an obstacle-free area.

*Email adresses: [name].[surname]@enac.fr
†Deceased, https://sites.laas.fr/Simon_Lacroix/

Let the aircraft be numbered from 1 to N, and represented using a 2D position $[x_k, y_k] \in \mathbb{R}^2$ and a 2D orientation $\theta_k \in \mathbb{S} := (-\pi, \pi]$ with respect to the ground. We combine these values in a state vector $\mathbf{p}_k = [x_k, y_k, \theta_k]$. For the sake of simplicity, we assume all aircraft have the same characteristics, namely the same cruise air speed V, minimal turn radius $\rho_{\min}$, and minimal separation distance δ.

This yields the following dynamics for each aircraft $k \in [1, N]$, the so-called *Dubins vehicle* [18]:

$$\frac{\mathrm{d}\mathbf{p}_k}{\mathrm{d}t} = \begin{pmatrix} \dot{x_k} \\ \dot{y_k} \\ \dot{\theta_k} \end{pmatrix} := \begin{pmatrix} V\cos\theta_k \\ V\sin\theta_k \\ u_k \end{pmatrix} \tag{1}$$

The sole control parameter is the change in orientation $u_k \in [{}^{-V}/_{\rho_{\min}}, {}^{V}/_{\rho_{\min}}]$. Note that the vehicle is always aligned with its speed vector. As such, for a differentiable trajectory $\gamma : \mathbb{R} \to \mathbb{R}^2$, we may abuse the notation $\mathbf{p}_k(t) = \gamma(t)$ to state that $\gamma(t) = [x_k(t), y_k(t)]$ and $\gamma'(t)$ is aligned with $[\cos\theta_k(t), \sin\theta_k(t)]$, i.e. the aircraft follows γ both in position and orientation.

We require synchronicity: all aircraft reach their destination at the same time. This constraint stems from formation flights, and can be adapted to sequencing for instance. We develop this aspect in section 4.

Let us formulate our optimization problem. Given for each aircraft k an initial state $\mathbf{S}_k$, a final state $\mathbf{E}_k$ both in $\mathbb{R}^2 \times \mathbb{S}$, and a set of possible twice differentiable paths $\Gamma \subset \mathcal{D}^2(\mathbb{R}^+, \mathbb{R}^2)$, we seek to minimize the time taken by the fleet to reach its final configuration while satisfying the constraints. Formally:

$$\min_{\substack{\tau \in \mathbb{R}^+ \\ \gamma_k \in \Gamma, 1 \le k \le N}} \quad \tau \qquad \text{(Flight time)} \tag{2}$$

Such that we achieve:

1. Correct synchronous trajectory:

$$\forall\, k \in [1, N] \quad \gamma_k(0) = \mathbf{S}_k \ \wedge\ \gamma_k(\tau) = \mathbf{E}_k$$

2. Collision avoidance:

$$\forall\, k, s,\ k \neq s \quad \forall\, t \in [0, \tau]\ \ \|\gamma_k(t) - \gamma_s(t)\| > \delta$$

3. Constant speed:

$$\forall\, k \quad \forall\, t \in [0, \tau]\ \ \|\gamma_k'(t)\| = V$$

4. Bounded acceleration:

$$\forall\, k \quad \forall\, t \in [0, \tau]\ \ \|\gamma_k''(t)\| \le \frac{V^2}{\rho_{\min}}$$

The core starting point for solving this problem is to define the set of possible paths Γ. This set is often made of splines [13, 15, 16, 17] as they provide a continuous family with a convenient representation for integrating spatial and dynamic constraints in a solver. But they heavily rely on a good initial guess and continuous optimization. This can be tempered by the use of global heuristics like Particle Swarm Optimization (PSO), at a significant computational cost.

The method presented here is based on a finite set of solutions. It has the advantage of exploring significantly different paths, which may then be used as a better starting point for locally optimized methods. We now present the elements making our set of solutions: the *Dubins paths*.

3 Dubins paths

Dubins showed in [18] that the shortest path between any two states of a vehicle described in (1) is made of straight lines and circle arcs of radius $\rho_{\min}$. We denote the possible components S, L, R, respectively for Straight, Left and Right [turn]. The optimal solution is among the following six possibilities: RSR, LSL ; RSL, LSR and LRL, RLR.

We may still need more variety in the choice of paths in order to avoid conflicts. We add the single-turn paths SLS and SRS. They have a limited scope of application, but can still extend the number of possibilities. With the six basic Dubins paths and these two, we define the set of *basic paths* γ_0, which has cardinality 8. For any type $\gamma \in \gamma_0$, a path is exactly defined by starting and ending poses $\mathbf{S}, \mathbf{E} \in \mathbb{R}^2 \times \mathbb{S}$ and a turn radius $\rho > 0$, and we denote $\gamma[\mathbf{S}, \mathbf{E}, \rho] \in \mathcal{D}^2(\mathbb{R}, \mathbb{R}^2)$ the corresponding function, when it exists. They are all showcased in Figure 1.

By construction, all of these solutions are guaranteed to satisfy the Constant speed and Bounded acceleration constraints. Two main questions remain: how to detect conflicts and how to tune the paths to achieve synchronicity ?

3.1 *Finding a fitting solution*

Dubins paths provide a direct solution to the individual path planning problem, but offer little tuning mechanisms for achieving a specific path length. This question has been studied, and several methods devised: artificially increase the turning radius [19], use circles with different radii [21], in-

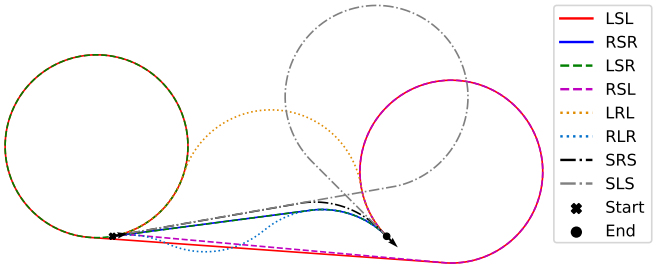

Figure 1: Examples of the six Dubins paths plus two extras (SRS, SLS)

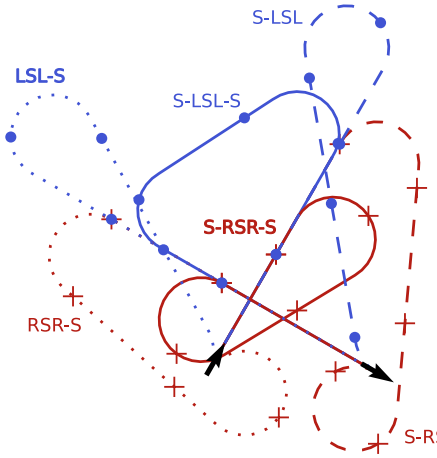

Figure 2: Start, end and both extended paths: RSR and LSL

troduce straight extensions [22] or deform straight parts [23]. We use two strategies: increasing the turn radius and adding straight segments at the start or end of the path.

More formally, fitting length $\ell > 0$ by tweaking radius ρ of a path from $\mathbf{S} = [x_S, y_S, \theta_S]$ to $\mathbf{E} = [x_E, y_E, \theta_E]$ corresponds to the following optimization problem:

$$\min_{\rho > \rho_{\min}} (\text{len}(\gamma[\mathbf{S}, \mathbf{E}, \rho]) - \ell)^2 \tag{3}$$

Similarly, using length extensions distributed with ratio $r \in [0, 1]$ along the initial direction $\mathbf{V_S} = [\cos\theta_S, \sin\theta_S, 0]$ and final one $\mathbf{V_E} = [\cos\theta_E, \sin\theta_E, 0]$ yields the problem:

$$\min_{l \geq 0} (\text{len}(\gamma[\mathbf{S} + lr\mathbf{V_S}, \mathbf{E} - l(1-r)\mathbf{V_E}, \rho_{\min}]) + l - \ell)^2 \tag{4}$$

Note that the second method introduces new types of paths: Dubins extended by straight lines. The ratio parameter allows many variations, but for the sake of simplicity we will only keep three: start extended ($r = 1$), noted S-*AAA*; end extended ($r = 0$) *AAA*-S and both extended ($r = 1/2$) S-*AAA*-S, where in all cases *AAA* represents one of the three letters combination characterizing an element of γ_0. A showcase of some Dubins extended paths is given in Figure 2. Let Γ define the set containing the *basic paths* γ_0 length fitted using the aforementioned methods; its cardinal is $8 + 3 \cdot 8$.

Solving problems (3) and (4) is manageable since they are unidimensional and quite monotonous, in spite of some discontinuities (see [14, Figure 4] for path length as a function of the turn radius). We use Brent's method [24, Chapter 4] to efficiently find solutions.

3.2 *Conflicts detection*

Dubins path are made of straight lines and circle arcs. Thus the question of detecting a conflict between two paths of Γ can be broken down into finding conflicts between their components. Three subcases emerge: line-line, line-circle and circle-circle. Both 2D lines and circles can be conveniently represented using complex functions.

For two parametric functions $f_1, f_2 : [0, \tau] \to \mathbb{C}$ representing two trajectories, we define two types of separations with respect to some minimal distance $\delta > 0$:

- *Temporal separation* : two points following prescribed trajectories must stay separated by at least δ.

$$\min_{t \in [0,\tau]} |f_1(t) - f_2(t)| > \delta \tag{5}$$

- *Spatial separation* : two shapes must be separated by at least δ.

$$\min_{(t_1,t_2) \in [0,\tau]^2} |f_1(t_1) - f_2(t_2)| > \delta \tag{6}$$

One can easily verify that if (6) is satisfied, then so is (5).

Practically, we are interested in *temporal separation*. It is easy to compute in the line-line case, as it becomes a second degree polynomial equation. But there are no such simplifications for the others ; we require a global solver, the interval method [25] for instance. To limit the use of such computationally expensive procedures, we can perform pre-filtering by computing the *spatial separation*, which is straightforward given the geometric nature of the paths considered.

We have presented in this section the set of paths Γ used for solving problem (2). We have shown how to fit them to specific length and how to detect conflicts between them. Let us now put this together into a fleet path planner.

4 FLEET PATH PLANNING

4.1 *Presentation of the algorithm*

Our procedure for planning a path for the fleet respecting the constraints given in (2) uses combinatorial optimization. We consider two main factors regarding complexity: the number of different possible paths $|\Gamma|$ and the number of aircraft N. The former can be increased or reduced by modifying the chosen individual path planning methods, and the latter expresses the scale and difficulty of the problem.

Three primitives are required to describe our method, whose underlying principles have been given in section 3:

- $\tau_k, \gamma_k \longleftarrow \text{ShortestDubins}(\mathbf{S}_k, \mathbf{E}_k, V, \rho_{\min})$: Given starting and ending states $\mathbf{S}_k, \mathbf{E}_k$, the aircraft speed V and its minimal turn radius $\rho_{\min}$, compute the shortest path γ_k belonging to Γ and give the travel time τ_k.

  Its complexity scales linearly with $|\Gamma|$, since each path must be considered to find the shortest.

- $L_k \longleftarrow \text{FitDubins}(\mathbf{S}_k, \mathbf{E}_k, V, \rho_{\min}, \tau)$: Given starting and ending states $\mathbf{S}_k, \mathbf{E}_k$, the aircraft speed V, its minimal turn radius $\rho_{\min}$ and a travel time τ, output a set of paths L_k taken from Γ with turning radii greater or equal than $\rho_{\min}$ and having a travel time of τ. This set may be empty if there is no way to go from $\mathbf{S}_k$ to $\mathbf{E}_k$ given the constraints and possible paths.

  Its complexity also scales linearly with $|\Gamma|$, since each path must be considered.

- True|False $\longleftarrow$ AreSeparated$(\delta, \gamma_1, \ldots, \gamma_N)$: Given a minimal separation distance δ and a list of paths $\gamma_1, \ldots, \gamma_N$, check if there is a loss of separation between any two pairs.

  We can safely assume that there is an underlying IsPairSeparated$(\delta, \gamma_a, \gamma_b)$ subroutine checking the separation between two given paths, solving *spatial* (6) then possibly *temporal* (5) separations. AreSeparated requires $\frac{N(N-1)}{2} = \mathcal{O}\left(N^2\right)$ calls to the subroutine (since separation is a symmetric notion).

We can now describe the general principles, also summarized in pseudocode algorithm 1. It starts by computing for each aircraft its best individual solution, ignoring collisions, by using the ShortestDubins primitive. We want synchronous arrivals, i.e. same travel time for everyone. A first good guess is the maximum of the minimum individual times, which we denote $\tau_{\min}$. Using a user-defined parameter R, the maximum possible time is set to $R \cdot \tau_{\min}$. We use these two values to initialize a priority queue Q containing the flight times considered.

The rest of algorithm 1 executes the main loop until a stop condition is met: For every untested flight time τ in Q, compute the possible paths L_k for each aircraft $k \in [1, N]$ using FitDubins, and look through all combinations $(\gamma_1, \ldots \gamma_N) \in L_1 \times \cdots \times L_N$; if one satisfies AreSeparated with the desired separation δ, break the loop, otherwise move to the next value in Q.

If a solution is found, remove all elements in Q with flight time greater than the one of the solution, since they cannot be optimal. Then, Q is populated by inserting b values regularly spaced between every two successive $\tau_i, \tau_{i+1} \in Q$ (if $b = 1$, this is equivalent to successive dichotomies). To limit sampling to a reasonable resolution, a minimum width $w > 0$ is specified by the user, such that if $\tau_{i+1} - \tau_i \leq w$, no samples are added between these two values.

The number of iterations is counted as the number of different flight times τ tested. Once above a given threshold, the program stops. The second stop condition is the maximum total execution time, and is evaluated before testing a new τ. The last stop condition is a progress criterion: the program stops if no new τ are added to Q when re-sampling.

4.2 Algorithm analysis

The different stop conditions ensure that the program terminates in a timely manner, but does not guarantee that a solution is found. Furthermore, there is no guarantee at large to find a solution, even with unlimited computations, as it may not exist in the Γ set chosen. Still, we can evaluate the complexity of testing a flight duration τ, which is the core of this algorithm. Computing the set of possible paths for each aircraft contributes $N \cdot \mathcal{O}(|\Gamma|)$. Then, there are at most $|\Gamma|$ paths per aircraft, meaning a total of $|\Gamma|^N$ combinations. Each combination has to be tested for conflicts, yielding a

Algorithm 1: Dubins based path planning algorithm

Data: Initial and final states $(\mathbf{S}_k, \mathbf{E}_k)_{1 \leq k \leq N}$; flight parameters $V, \rho_{\min}, \delta$; optimization parameters R, w, b
Result: Duration τ and Dubins trajectories $(\gamma_k)_{1 \leq k \leq N}$

```
for 1 ≤ k ≤ N do    // Initial solutions
    τ_k, γ_k ⟵ ShortestDubins(S_k, E_k, V, ρ_min)
end
τ_min ⟵ max_{1≤k≤N}(τ_k)
// Ascending priority queue
Q ⟵ [τ_min; R · τ_min]
Best ⟵ ∅
repeat
    foreach untested τ ∈ Q do
        // Compute all solutions for a given length
        for 1 ≤ k ≤ N do
            L_k ⟵ FitDubins(S_k, E_k, V, ρ_min, τ)
        end
        // If one L_k is empty, this loop is skipped
        foreach Combination (γ_1, ..., γ_N) of L_1 × ··· × L_N do
            if AreSeparated(δ, γ_1, ..., γ_N) then
                Best ⟵ τ, (γ_1, ..., γ_N)
                break twice
            end
        end
        Mark τ as tested
    end
    if Best is not empty then
        Remove from Q all τ >Best[0]
    end
    for Each successive pair τ_i, τ_{i+1} ∈ Q do
        Δτ ⟵ τ_{i+1} − τ_i
        if Δτ > w then
            for 1 ≤ j ≤ b do Add τ_i + j Δτ/(b+1) to Q
        end
    end
until Stop condition
return Best
```

cost per iteration of $\mathcal{O}\left(N^2 |\Gamma|^N\right)$. This is *polynomial* with respect to the number of different paths $|\Gamma|$ and *exponential* with respect to the number of aircraft N.

Here are two ways we used in our implementation to speed up computations. Since conflicts involve aircraft two by two, it is possible to compute for each of the $\frac{N(N-1)}{2}$ unordered aircraft pairs the $|\Gamma|^2$ paths pairs, and check for conflict. This provides $N^2 |\Gamma|^2$ binary coefficients $c_{a,b,g,h}$ denoting whether there is a conflict between aircraft a and b using respectively paths g and h. We can formulate a *0-1 Integer Linear Program* (ILP) with variables $x_{a,g}$ defining if aircraft a uses path g. There are two types of constraints: avoiding conflicts

$$\forall a \neq b \in [1, N] \; \forall g, h \in \Gamma \; c_{a,b,g,h} \cdot (x_{a,g} + x_{b,h}) \leq 1$$

and choosing a unique path for each aircraft

$$\forall a \in [1, N] \; \sum_{g \in \Gamma} x_{a,g} = 1$$

This is a well-known NP-complete problem, and an efficient existing solver can be used, for instance the HiGHS solver [26].

Another important point is that this algorithm is easily parallelizable. Indeed, each aircraft k can independently compute its set of possible paths L_k, and each conflict check is independent of the others. Hence, if we consider a centralized multiprocessor unit, we may use at most $\frac{|\Gamma|^2 N(N-1)}{2}$ cores to compute the aforementioned $c_{a,b,g,h}$.

4.3 Algorithm extensions

The capabilities of algorithm 1 can be extended by a clever use of its formulation. The first one is that it is possible to take into account a constant uniform wind W. Since the flight time τ is set before planning the paths, it is possible to update the ending point by integrating the wind deviation. This amounts to replace $\mathbf{E}_k$ by $\mathbf{E}_k - W\tau$, thereby anticipating the wind induced drift. Note that since the same linear change of referential is applied to every aircraft, conflict detection does not need to be changed. Regarding the initial guess for $\tau_{\min}$, there exist methods to compute it in presence of uniform wind, for instance [27].

The second modification consists in incorporating temporal separation at arrival. Assume that aircraft $k \in [2, N]$ must reach their destination with a time difference Δt_k with respect to the previous aircraft $k-1$. This is implemented by replacing τ in the call of FitDubins by $\tau + \sum_{j=1}^{k-1} \Delta t_j$. One use case of such modification is landing sequencing.

5 Simulations

The method is tested in simulation with 3 types of scenarios: transition between formations, random state to formation and random state to another random state, respectively referred to as Formation, RNG to Formation and Full RNG in

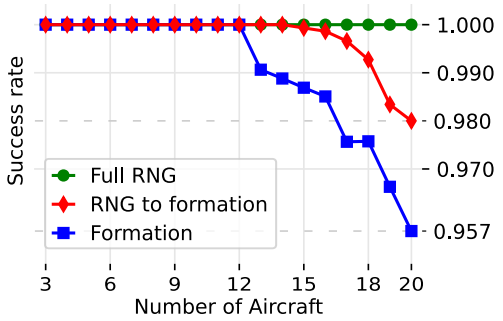

Figure 3: Success rate as function of aircraft number and type of test case

Figure 3. A total of 22 formations were considered, ranging from circular and linear to rectangular and their variants.

We based our simulations on a fixed-wing drone of 1 m wingspan (equivalent to a Parrot Disco). The airspeed is set to 15 m/s, minimal turn radius to 40 m and minimal separation to 80 m. When using random positions, sampling was done in a 1×1 km area with 240 m of distance requested. When using positions in formation, this distance between reference positions is set to 120 m. The distance between initial and final formations is of 1 km. No wind was considered in these tests. Regarding the algorithm parameters, we set $R = 10$, $b = 2$ and $w = \max(0.1\,\text{s}, R \cdot \tau_{\min} \cdot 10^{-4})$. The stop conditions values are 300 iterations maximum and a 60 s timeout. The solver has been benchmarked on a 12th Gen Intel(R) Core(TM) i7-1255U, using 12 threads.

We ran tests with fleets of 3 to 20 aircraft, with 5230 cases each: 300 fully random, 530 formation transitions and 4400 random to formation. The proportion of successful cases per aircraft number and problem type is reported in Figure 3. Success rate starts to decrease for more than 12 aircraft. Two main reasons can be given for failures. First, formation cases use tighter margins at endpoints and do not optimize aircraft reordering between formations. Instead, they use an arbitrary order, which may be unnecessarily difficult. Second, randomly selected initial points may hinder synchronicity, especially when the arrivals are tightly ordered. Nevertheless, for 12 aircraft and less, no failures were reported, and the rate stays above 95% when testing with 20 aircraft.

Computation times are reported in Figure 4. We measured the main thread time since parallelism has been exploited in our implementation. For 9 aircraft and below, 99% of cases are solved in less than 1 s. For 20 aircraft, 90% of cases were completed in less than 13 s. A detailed analysis reveals that the 60 s timeout is reached only for some cases with 17 aircraft or more. The number of iterations is never a limiting factor; the maximal value measured is 87 iterations.

The most difficult cases are the transitions between formations, then random to formations and finally fully random; the computation times with 20 aircraft average to 5065 ms, 3877 ms and 417 ms respectively. It can be explained by the

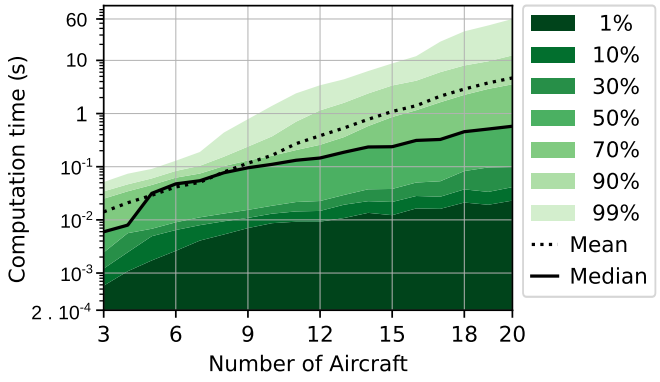

Figure 4: Computation time log distribution as function of aircraft number

lack of geometric separation between trajectories with formations, as it incurs the use of more time-consuming algorithms for detecting potential conflicts.

Preliminary integration of the method has been tested using the 6 degrees of freedom simulator from PaparazziUAV [28]. A centralized planner gather data from three simulated aircraft sharing the same characteristics: nominal airspeed set to $15\,\mathrm{m/s}$, minimal turn radius tuned to $40\,\mathrm{m}$ and minimal separation to $30\,\mathrm{m}$. This last parameter is significantly smaller compared to our initial tests in order to keep the planned trajectories in a reasonable airspace. Dynamic replanning is set up with the following rules, by decreasing order of priority: a new plan can be established at most once every $15\,\mathrm{s}$, the current plan is considered invalid if a deviation of $10\,\mathrm{m}$ is measured for one aircraft, a new plan can be proposed if it reduces flight time by 10%, the current plan ends at $8\,\mathrm{s}$ from arrival. Tracking is achieved using the predefined line and circle primitives of the autopilot. Each aircraft adjusts its own airspeed to try reducing tracking error. It is based on the time-to-arrival error, with adjustments within $\pm 1\,\mathrm{m/s}$ of its cruise speed.

One of the simulated flights is showcased in Figure 5. The objective is to achieve four straights in chevron formation with a spacing of $36\,\mathrm{m}$. The highlighted section contains the transition with the largest error. This error for aircraft 3 can be attributed to cutting the start of its turn and failing to slow down efficiently. This deviation triggered at $161\,\mathrm{s}$ a rescheduling, which at first increased the tracking error due to the sudden change in command, followed by a decrease as control stabilized. More generally, these simulations show a satisfying planning strategy, especially considering the naive tracking implemented. Still, the tracking error is not negligible and some margins have to be considered between the separation requested to the solver and the actual minimal tolerable separation.

6 Conclusions and Future Works

In this article we showed a method to solve synchronous path planning for a fleet of constant speed Dubins vehicles, and benchmarked it from 3 to 20 aircraft. On average, even up to 20 aircraft, the computation time is $8\,\mathrm{s}$ for a success rate superior to 95%. The method enforces synchronicity by setting the travel time first, then creating a finite set of paths per aircraft satisfying the constraints, and finally seeking a conflict-free combination, which is a solution to the fleet path planning problem. Simulated flights within the Paparazzi-UAV system show correct tracking given aircraft dynamics, but highlight the possible improvements in plan execution.

This algorithm has several limitations, the first one being that there are no guarantees to find a solution. This may be mitigated by adding possible paths and reducing margins, but these come with computational and safety costs. Secondly, computational cost scales exponentially with the number of aircraft, limiting scalability. This method was not made to replace swarm-like tactics used to handle large amounts of UAVs, but the time to find a solution can be an obstacle depending on the available hardware and expected reactivity. Lastly, the paths presented here are based on Dubins model. If it simplifies greatly path planning, it does not match an actual aircraft model, since lateral acceleration is discontinuous. Given how important separation between aircraft is, we wish that path following is as precise as possible.

There are multiple ways of improving this method. The most promising direction is to locally post-process the generated paths using B-Splines [14] or Pythagorean Hodograph splines [15, 17] in order to fix discontinuities, and then improve tracking accuracy and avoid collisions using appropriate guidance techniques, such as Guiding Vector Fields [29]. One could also improve computation time by studying how conflicts between paths evolve as they are continuously deformed to fit a different length. Lastly, other extensions are possible, such as integrating vertical movement [30].

Our C++ implementation of the solver can be found at https://github.com/enacuavlab/DubinsFleetPlanner.

References

[1] Esther Salamí, Cristina Barrado, and Enric Pastor. Uav flight experiments applied to the remote sensing of vegetated areas. *Remote Sensing*, 6(11):11051–11081, November 2014. ISSN: 2072-4292. DOI: 10.3390/rs61111051.

[2] Felipe Gonzalez, Marcos P.G. Castro, Pritesh Narayan, Rod Walker, and Les Zeller. Development of an autonomous unmanned aerial system to collect time-stamped samples from the atmosphere and localize potential pathogen sources. *Journal of Field Robotics*, 28(6):961–976, October 2011. ISSN: 1556-4967. DOI: 10.1002/rob.20417.

[3] Lucio R. Salinas, Georgios Tzoumas, Lenka Pitonakova, and Sabine Hauert. Digital twin technology for wildfire monitoring using uav swarms. In *2023 International Conference on Unmanned Aircraft Systems*

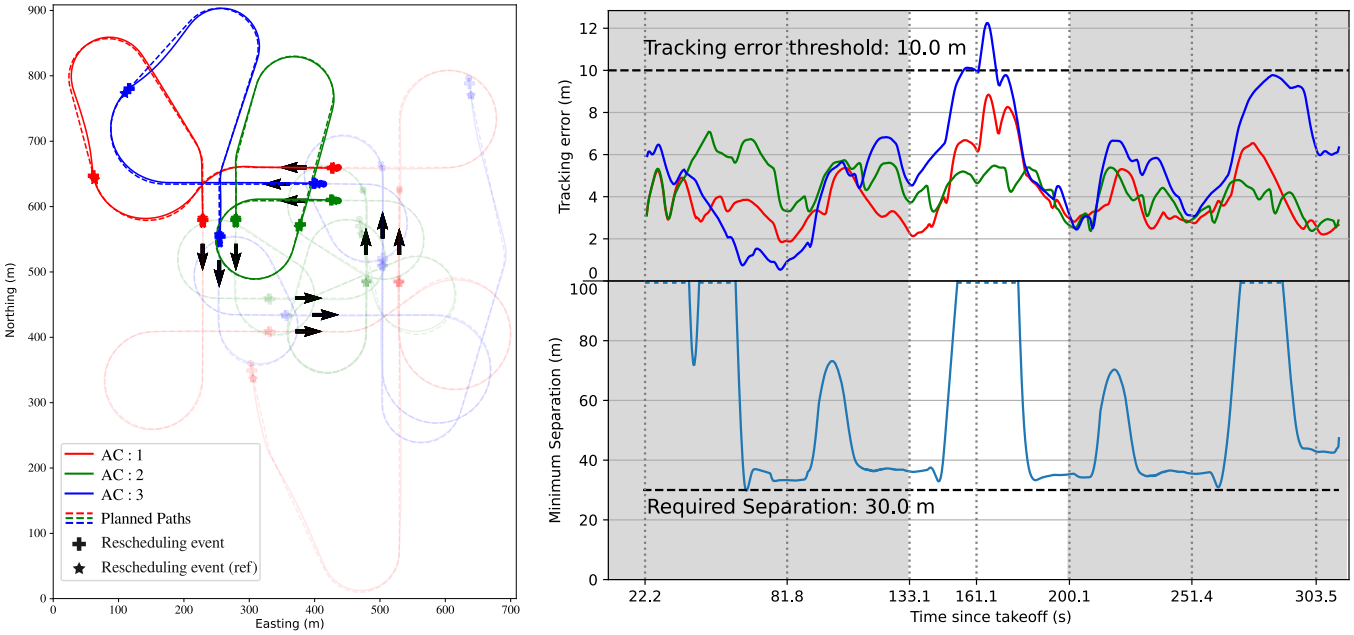

Figure 5: Flight of three identical aircraft simulated using PaparazziUAV. Separation measured up to 100 m

(ICUAS), pages 586–593. IEEE, June 2023. DOI: 10.1109/icuas57906.2023.10155819.

[4] Hao Zhang, Bin Xin, Li-hua Dou, Jie Chen, and Kaoru Hirota. A review of cooperative path planning of an unmanned aerial vehicle group. *Frontiers of Information Technology & Electronic Engineering*, 21(12):1671–1694, December 2020. ISSN: 2095-9230. DOI: 10.1631/fitee.2000228.

[5] Soon-Jo Chung, Aditya Avinash Paranjape, Philip Dames, Shaojie Shen, and Vijay Kumar. A survey on aerial swarm robotics. *IEEE Transactions on Robotics*, 34(4):837–855, 4, August 2018. ISSN: 1941-0468. DOI: 10.1109/tro.2018.2857475.

[6] Yuanchang Liu and Richard Bucknall. A survey of formation control and motion planning of multiple unmanned vehicles. *Robotica*, 36(7):1019–1047, March 2018. ISSN: 1469-8668. DOI: 10.1017/s0263574718000218.

[7] Heng Su, Hongquan Yun, Jingwen He, Feng Zhang, and Yue Wang. Multi-aircraft path planning method based on cooperative search a-star algorithm. *2019 IEEE International Conference on Unmanned Systems (ICUS)*, 2019. DOI: 10.1109/icus48101.2019.8995994.

[8] Luitpold Babel. Coordinated target assignment and uav path planning with timing constraints. *Journal of Intelligent & Robotic Systems*, 94(3–4):857–869, July 2018. ISSN: 1573-0409. DOI: 10.1007/s10846-018-0910-9.

[9] Zhihong Liu, Xiangke Wang, Lincheng Shen, Shulong Zhao, Yirui Cong, Jie Li, Dong Yin, Shengde Jia, and Xiaojia Xiang. Mission-oriented miniature fixed-wing uav swarms: a multilayered and distributed architecture. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 52(3):1588–1602, March 2022. ISSN: 2168-2232. DOI: 10.1109/tsmc.2020.3033935.

[10] Yeongho Song, Seunghan Lim, Hyunsam Myung, Hokeun Lee, Junho Jeong, Heungsik Lim, and Hyondong Oh. Distributed swarm system with hybrid-flocking control for small fixed-wing uavs: algorithms and flight experiments. *Expert Systems with Applications*, 229:120457, November 2023. ISSN: 0957-4174. DOI: 10.1016/j.eswa.2023.120457.

[11] Varun Madabushi, Yocheved Kopel, Adam Polevoy, and Joseph Moore. Dense fixed-wing swarming using receding-horizon nmpc. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pages 8656–8662. IEEE, May 2025. DOI: 10.1109/icra55743.2025.11127916.

[12] Madhavan Shanmugavel, Antonios Tsourdos, Rafal Zbikowski, and Brian White. Path planning of multiple uavs with clothoid curves in two dimensions. *IFAC Proceedings Volumes*, 40(7):461–466, 2007. ISSN: 1474-6670. DOI: 10.3182/20070625-5-fr-2916.00079.

[13] Wang Yu, Wang Shuo, Wang Rui, and Tan Min. Generation of temporal–spatial bezier curve for simultaneous arrival of multiple unmanned vehicles. *Informa-*

tion Sciences, 418–419:34–45, December 2017. ISSN: 0020-0255. DOI: 10.1016/j.ins.2017.07.031.

[14] Landon Shumway and Randal W. Beard. Time-synchronized b-spline path planning for multi-agent uav systems with fixed speed profiles. In *2025 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 272–278. IEEE, May 2025. DOI: 10.1109/icuas65942.2025.11007834.

[15] Madhavan Shanmugavel, Antonios Tsourdos, Rafal Zbikowski, and Brian White. 3d path planning for multiple uavs using pythagorean hodograph curves. In *AIAA Guidance, Navigation and Control Conference and Exhibit*. American Institute of Aeronautics and Astronautics, June 2007. DOI: 10.2514/6.2007-6455.

[16] Venanzio Cichella, Ronald Choe, Bilal S. Mehdi, Enric Xargay, Naira Hovakimyan, Anna C. Trujillo, and Isaac Kaminer. Trajectory generation and collision avoidance for safe operation of cooperating uavs. In *AIAA Guidance, Navigation, and Control Conference*. American Institute of Aeronautics and Astronautics, January 2014. DOI: 10.2514/6.2014-0972.

[17] Ronald Choe, Javier Puig-Navarro, Venanzio Cichella, Enric Xargay, and Naira Hovakimyan. Cooperative trajectory generation using pythagorean hodograph bézier curves. *Journal of Guidance, Control, and Dynamics*, 39(8):1744–1763, August 2016. ISSN: 1533-3884. DOI: 10.2514/1.g001531.

[18] Lester E. Dubins. On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents. *American Journal of Mathematics*, 79(3):497, July 1957. ISSN: 0002-9327. DOI: 10.2307/2372560. URL: http://www.jstor.org/stable/2372560 (visited on 07/02/2025).

[19] Madhavan Shanmugavel, Antonios Tsourdos, Rafał Żbikowski, and Brian White. Path planning of multiple uavs using dubins sets, 2005. DOI: 10.2514/6.2005-5827.

[20] Qing-yang Chen, Ya-fei Lu, Gao-wei Jia, Yue Li, Bing-jie Zhu, and Jun-can Lin. Path planning for uavs formation reconfiguration based on dubins trajectory. *Journal of Central South University*, 25(11):2664–2676, November 2018. ISSN: 2227-5223. DOI: 10.1007/s11771-018-3944-z.

[21] Aditya K. Rao, Kushal P. Singh, and Twinkle Tripathy. Curvature bounded trajectories of desired lengths for a dubins vehicle. *Automatica*, 167:111749, September 2024. ISSN: 0005-1098. DOI: 10.1016/j.automatica.2024.111749.

[22] Zheng Chen, Kun Wang, and Heng Shi. Elongation of curvature-bounded path. *Automatica*, 151:110936, May 2023. ISSN: 0005-1098. DOI: 10.1016/j.automatica.2023.110936.

[23] Weiran Yao, Naiming Qi, Jun Zhao, and Neng Wan. Bounded curvature path planning with expected length for dubins vehicle entering target manifold. *Robotics and Autonomous Systems*, 97:217–229, November 2017. ISSN: 0921-8890. DOI: 10.1016/j.robot.2017.09.003. URL: https://www.sciencedirect.com/science/article/pii/S092188901730132X.

[24] Richard Peirce Brent. *Algorithms for minimization without derivatives*. Prentice-Hall, 1973.

[25] E. R. Hansen. Global optimization using interval analysis: the one-dimensional case. *Journal of Optimization Theory and Applications*, 29(3):331–344, 1979. ISSN: 1573-2878. DOI: 10.1007/BF00933139.

[26] Q. Huangfu and J. A. J. Hall. Parallelizing the dual revised simplex method. *Mathematical Programming Computation*, 10(1):119–142, 2018. ISSN: 1867-2957. DOI: 10.1007/s12532-017-0130-5. URL: https://doi.org/10.1007/s12532-017-0130-5.

[27] Fanchen Wu and Zheng Chen. Minimum-time paths for dubins airplane in steady wind. *Journal of Guidance, Control, and Dynamics*:1–13, August 2025. ISSN: 1533-3884. DOI: 10.2514/1.g008932.

[28] Gautier Hattenberger, Murat Bronz, and Michel Gorraz. Using the Paparazzi UAV System for Scientific Research. In *IMAV 2014, International Micro Air Vehicle Conference and Competition 2014*, pp 247–252, Delft, Netherlands, August 2014. DOI: 10.4233/uuid:b38fbdb7-e6bd-440d-93be-f7dd1457be60. URL: https://enac.hal.science/hal-01059642.

[29] Weijia Yao, Héctor Garcia de Marina, Zhiyong Sun, and Ming Cao. Guiding vector fields for the distributed motion coordination of mobile robots. *IEEE Transactions on Robotics*, 39(2):1119–1135, April 2023. ISSN: 1941-0468. DOI: 10.1109/tro.2022.3224257.

[30] Mark Owen, Randal W. Beard, and Timothy W. McLain. *Implementing dubins airplane paths on fixed-wing uavs**. In *Handbook of Unmanned Aerial Vehicles*. Kimon P. Valavanis and George J. Vachtsevanos, editors. Springer Netherlands, Dordrecht, August 2014, pages 1677–1701. ISBN: 978-90-481-9707-1. DOI: 10.1007/978-90-481-9707-1_120.